ATMIC NETWORKS
YOUR GUIDE TO GLOBAL CERTIFIATION

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

www.atmicnetworks .com

Warning: Keep connected with our support team for latest updates

Question: 1

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.
Reference: https://docs.mulesoft.com/api-manager/2.x/policy-mule3-provided-policies

Question: 2

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.
Reference: https://help.mulesoft.com/s/question/0D52T00004mXSTUSA4/how-should-a-company-measure-c4e-success

Question: 3

An organization is implementing a Quote of the Day API that caches today's quote.
What scenario can use the GoudHub Object Store via the Object Store connector to persist the cache's state?

A. When there are three CloudHub deployments of the API implementation to three separate CloudHub regions that must share the cache state
B. When there are two CloudHub deployments of the API implementation by two Anypoint Platform business groups to the same CloudHub region that must share the cache state
C. When there is one deployment of the API implementation to CloudHub and anottV deployment to a customer-hosted Mule runtime that must share the cache state
D. When there is one CloudHub deployment of the API implementation to three CloudHub workers that must share the cache state

Answer: D

Explanation:

Correct Answer: When there is one CloudHub deployment of the API implementation to three CloudHub workers that must share the cache state.
Key details in the scenario:

> > Use the CloudHub Object Store via the Object Store connector
Considering above details:
> > CloudHub Object Stores have one-to-one relationship with CloudHub Mule Applications.
> > We CANNOT use an application's CloudHub Object Store to be shared among multiple Mule applications running in different Regions or Business Groups or Customer-hosted Mule Runtimes by using Object Store connector.
> > If it is really necessary and very badly needed, then Anypoint Platform supports a way by allowing access to CloudHub Object Store of another application using Object Store REST API. But NOT using Object Store connector.
So, the only scenario where we can use the CloudHub Object Store via the Object Store connector to persist the cache's state is when there is one CloudHub deployment of the API implementation to multiple CloudHub workers that must share the cache state.

Question: 4

What condition requires using a CloudHub Dedicated Load Balancer?

A. When cross-region load balancing is required between separate deployments of the same Mule application
B. When custom DNS names are required for API implementations deployed to customer-hosted Mule runtimes
C. When API invocations across multiple CloudHub workers must be load balanced
D. When server-side load-balanced TLS mutual authentication is required between API implementations and API clients

Answer: D

Explanation:

Correct Answer: When server-side load-balanced TLS mutual authentication is required between API implementations and API clients

Fact/ Memory Tip: Although there are many benefits of CloudHub Dedicated Load balancer, TWO important things that should come to ones mind for considering it are:
> > Having URL endpoints with Custom DNS names on CloudHub deployed apps
> > Configuring custom certificates for both HTTPS and Two-way (Mutual) authentication.
Coming to the options provided for this question n :
> > We CANNOT use DLB to perform cross-region load balancing between separate deployments of the same Mule application.
> > We can have mapping rules to have more than one DLB URL pointing to same Mule app. But vicevera (More than one Mule app having same DLB URL) is NOT POSSIBLE
> > It is true that DLB helps to setup custom DNS names for Cloudhub deployed Mule apps but NOT true for apps deployed to Customer-hosted Mule Runtimes.
> > It is true to that we can load balance API invocations across multiple CloudHub workers using DLB but it is NOT A MUST. We can achieve the same (load balancing) using SLB (Shared Load Balancer) too. We DO NOT necessarily require DLB for achieve it.
So the only right option that fits the scenario and requires us to use DLB is when TLS mutual authentication is required between API implementations and API clients.
Reference: https://docs.mulesoft.com/runtime-manager/cloudhub-dedicated-load-balancer

Question: 5

What do the API invocation metrics provided by Anypoint Platform provide?

A. ROI metrics from APIs that can be directly shared with business users
B. Measurements of the effectiveness of the application network based on the level of reuse
C. Data on past API invocations to help identify anomalies and usage patterns across various APIs
D. Proactive identification of likely future policy violations that exceed a given threat threshold

Answer: C

Explanation:

Correct Answer: Data on past API invocations to help identify anomalies and usage patterns across various APIs
API Invocation metrics provided by Anypoint Platform:
> > Does NOT provide any Return Of Investment (ROI) related information. So the option suggesting it is OUT.
> > Does NOT provide any information w.r.t how APIs are reused, whether there is effective usage of APIs or not etc...
> > Does NOT prodive any prediction information as such to help us proactively identify any future policy violations.
So, the kind of data/information we can get from such metrics is on past API invocations to help identify anomalies and usage patterns across various APIs.
Reference:
https://usermanual.wiki/Document/APAAppNetstudentManual02may2018.991784750.pdf

Question: 6

What is true about the technology architecture of Anypoint VPCs?

A. The private IP address range of an Anypoint VPC is automatically chosen by CloudHub
B. Traffic between Mule applications deployed to an Anypoint VPC and on-premises systems can stay within a private network
C. Each CloudHub environment requires a separate Anypoint VPC
D. VPC peering can be used to link the underlying AWS VPC to an on-premises (non AWS) private network

Answer: B

Explanation:

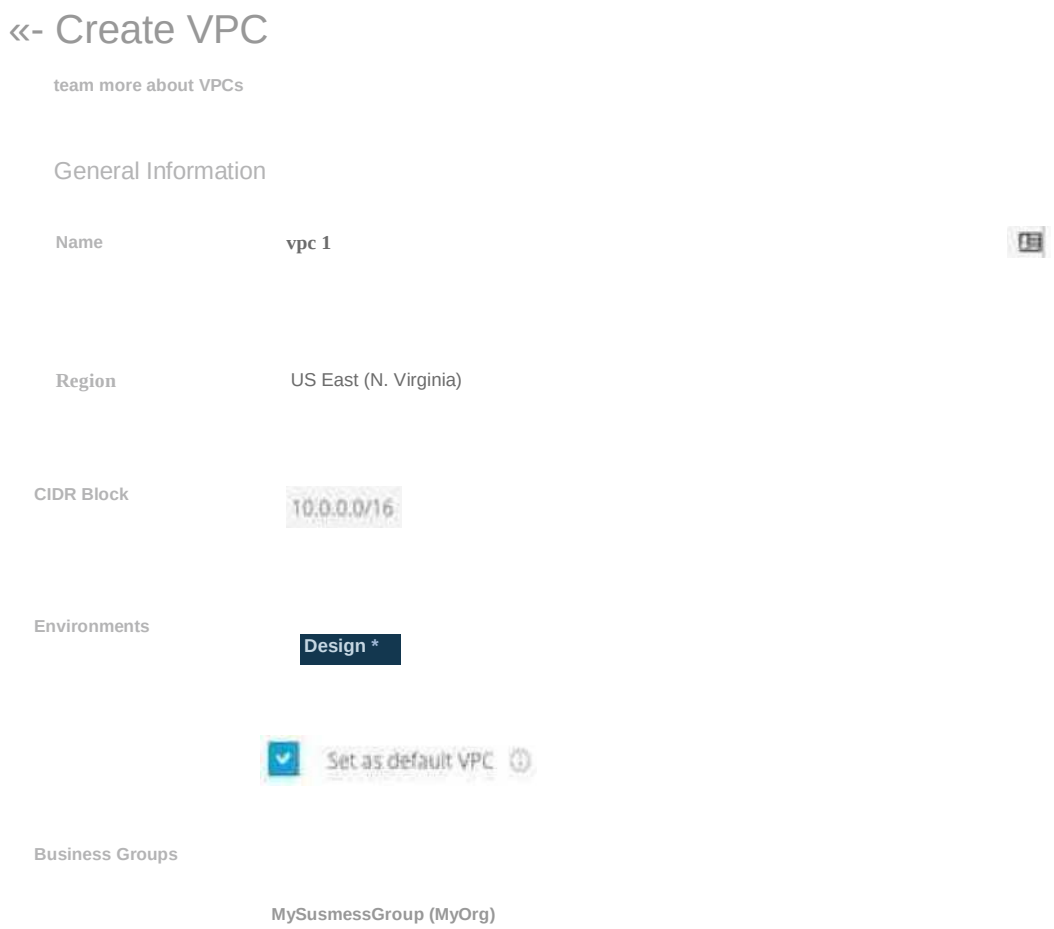
Correct Answer: Traffic between Mule applications deployed to an Anypoint VPC and on-premises systems can stay within a private network

E. The private IP address range of an Anypoint VPC is NOT automatically chosen by CloudHub. It is chosen by us at the time of creating VPC using thr CIDR blocks.

CIDR Block: The size of the Anypoint VPC in Classless Inter-Domain Routing (CIDR) notation.

For example, if you set it to 10.111.0.0/24, the Anypoint VPC is granted 256 IP addresses from 10.111.0.0 to 10.111.0.255.

Ideally, the CIDR Blocks you choose for the Anypoint VPC come from a private IP space, and should not overlap with any other Anypoint VPC's CIDR Blocks, or any CIDR Blocks in use in your corporate network.



that each CloudHub environment requires a separate Anypoint VPC. Once an Anypoint VPC is created, we can choose a same VPC by multiple environments. However, it is generally a best and recommended practice to always have seperate Anypoint VPCs for Non-Prod and Prod environments. >> We use Anypoint VPN to link the underlying AWS VPC to an on-premises (non AWS) private network. NOT VPC Peering.

Reference: https://docs.mulesoft.com/runtime-manager/vpn-about

Only true statement in the given choices is that the traffic between Mule applications deployed to an Anypoint VPC and on-premises systems can stay within a private network.

https://docs.mulesoft.com/runtime-manager/vpc-connectivity-methods-concept

Question: 7

An API implementation is deployed on a single worker on CloudHub and invoked by external API

clients (outside of CloudHub). How can an alert be set up that is guaranteed to trigger AS SOON AS that API implementation stops responding to API invocations?

A. Implement a heartbeat/health check within the API and invoke it from outside the Anypoint Platform and alert when the heartbeat does not respond

B. Configure a "worker not responding" alert in Anypoint Runtime Manager

C. Handle API invocation exceptions within the calling API client and raise an alert from that API client when the API Is unavailable

D. Create an alert for when the API receives no requests within a specified time period

Answer: B

Explanation:

Correct Answer: Configure a “Worker not responding” alert in Anypoint Runtime Manager.

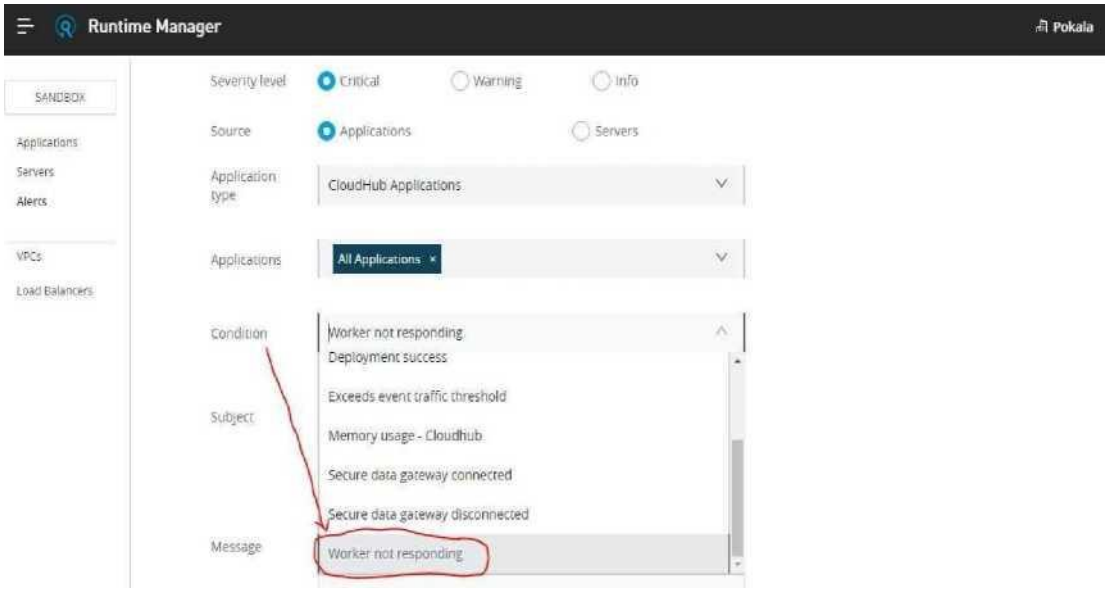
> > All the options eventually helps to generate the alert required when the application stops responding.

> > However, handling exceptions within calling API and then raising alert from API client is inappropriate and silly. There could be many API clients invoking the API implementation and it is not ideal to have this setup consistently in all of them. Not a realistic way to do.

> > Implementing a health check/ heartbeat with in the API and calling from outside to detmine the health sounds OK but needs extra setup for it and same time there are very good chances of generating false alarms when there are any intermittent network issues between external tool calling the health check API on API implementation. The API implementation itself may not have any issues but due to some other factors some false alarms may go out.

> > Creating an alert in API Manager when the API receives no requests within a specified time period would actually generate realistic alerts but even here some false alarms may go out when there are genuinely no requests from API clients.

The best and right way to achieve this requirement is to setup an alert on Runtime Manager with a condition "Worker not responding". This would generate an alert AS SOON AS the workers become unresponsive.



Bottom of Form

Top of Form

Question: 8

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

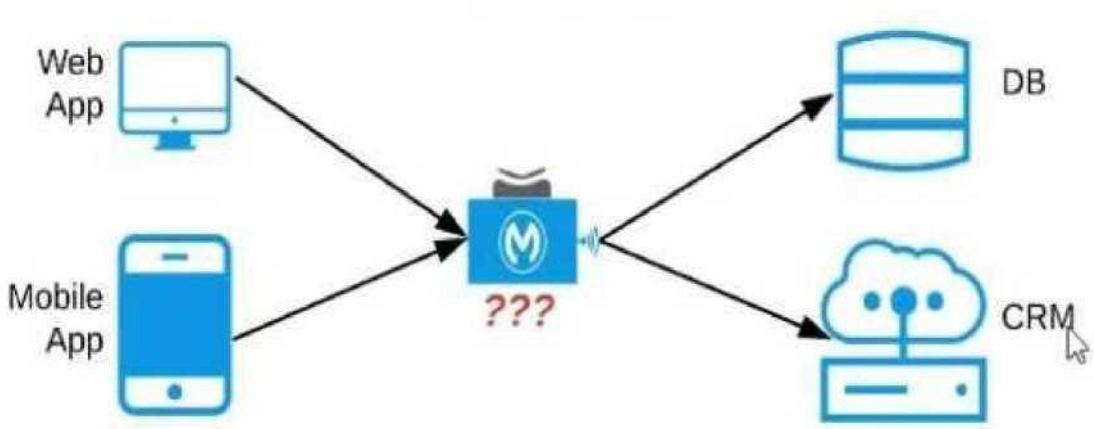
Answer: C

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.

Question: 9

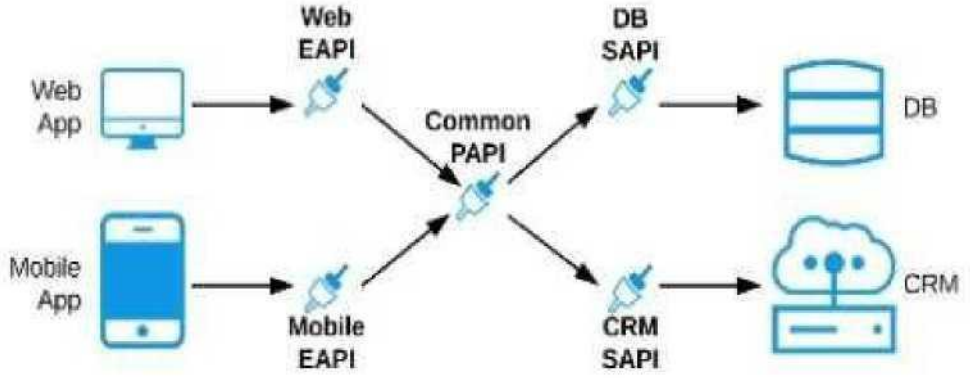
Refer to the exhibit. An organization needs to enable access to their customer data from both a mobile app and a web application, which each need access to common fields as well as certain unique fields.
The data is available partially in a database and partially in a 3rd-party CRM system.
What APIs 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:

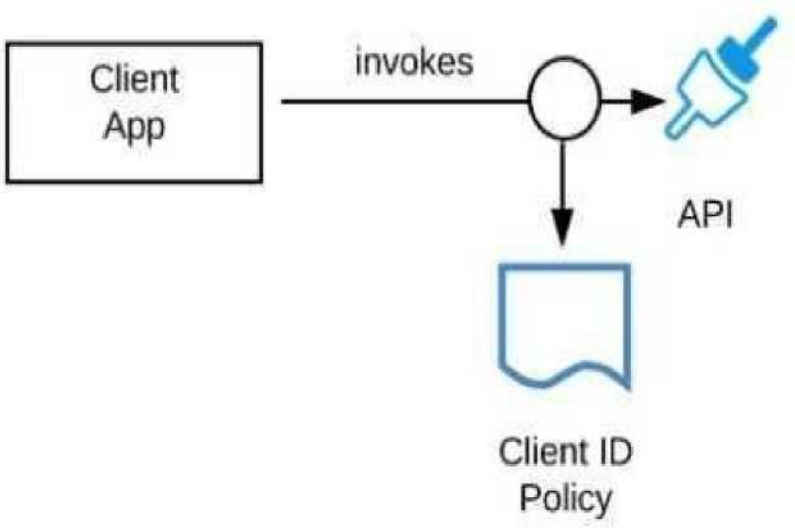
> > Experience APIs should be built as per each consumer needs and their experience.

> > 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.
Reference: https://blogs.mulesoft.com/dev/api-dev/what-is-api-led-connectivity/

Question: 10

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.
Reference:
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

Question: 11

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 seems 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 fron Mule Runtimes only irrespective of whether the deployment model is MuleSoft-hosted or Customer-hosted or Hybrid.

Question: 12

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 toad-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.

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

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

G. 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.

Reference: https://docs.mulesoft.com/http-connector/1.5/http-documentation#parameters-3

Question: 13

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.

Reference:

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

Question: 14

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

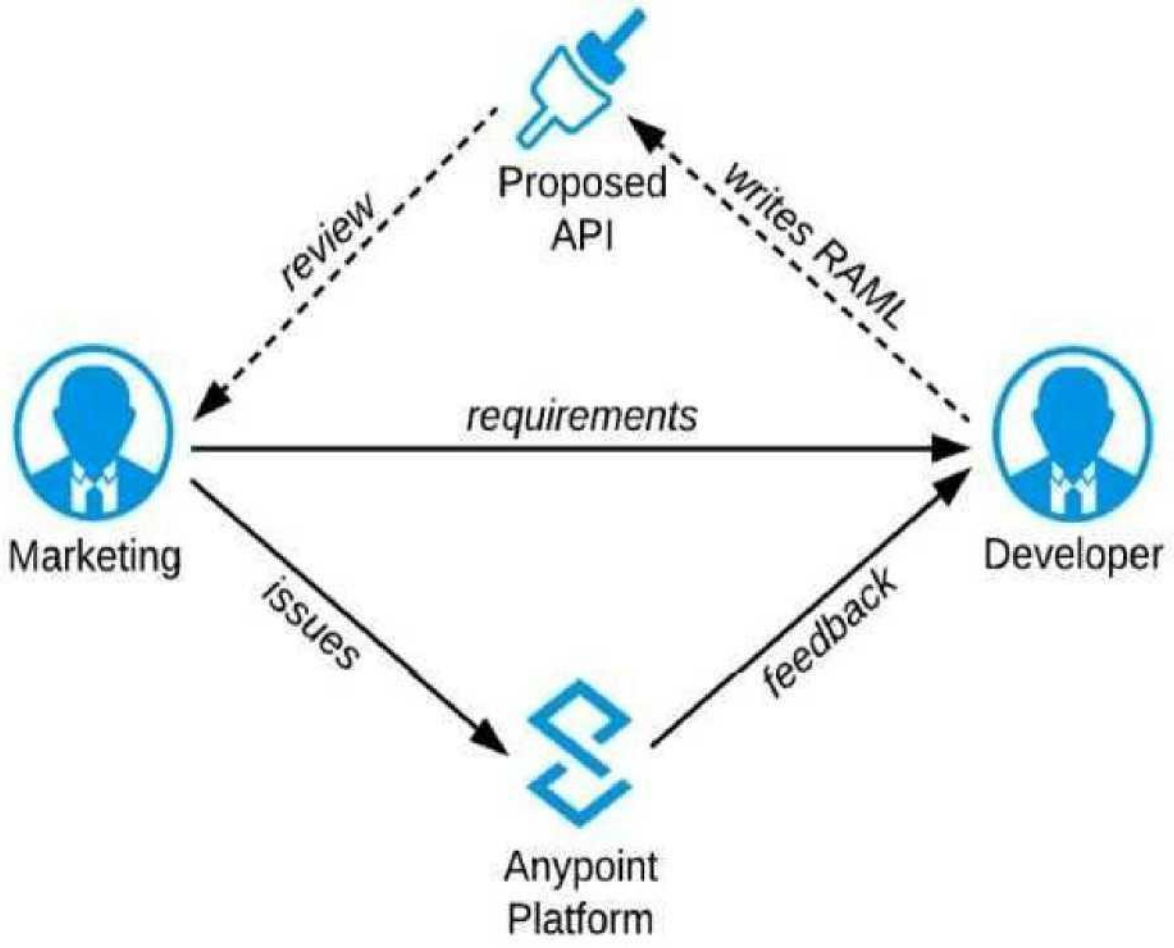
Explanation:

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

Reference: https://docs.mulesoft.com/exchange/to-share-api-asset-to-portal
https://docs.mulesoft.com/exchange/to-share-api-asset-to-portal

Question: 15

Refer to the exhibit.

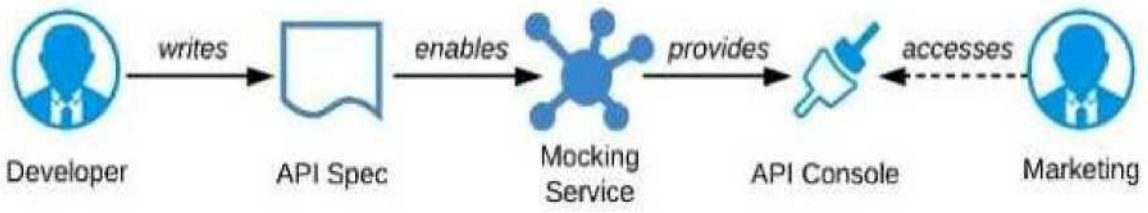


A RAML definition has been proposed for a new Promotions Process API, and has been published to Anypoint Exchange.

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

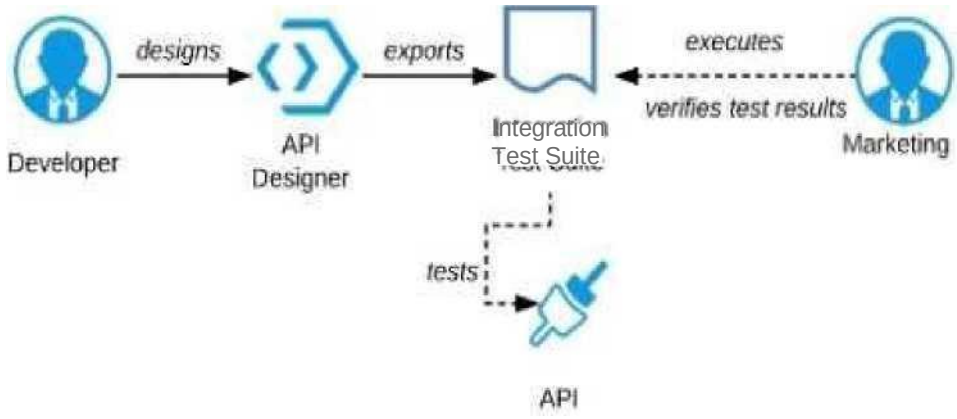


B) Organize a design workshop with the DBAs of the Marketing Department in which the database schema of the Marketing IT systems is translated into RAML

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



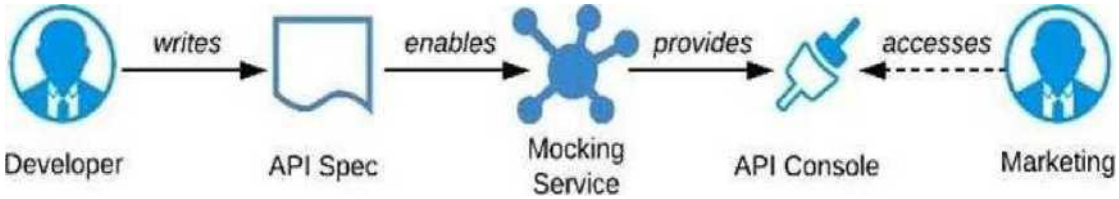
A. Option A
B. Option B
C. Option C
D. Option D

Answer: A

Explanation:

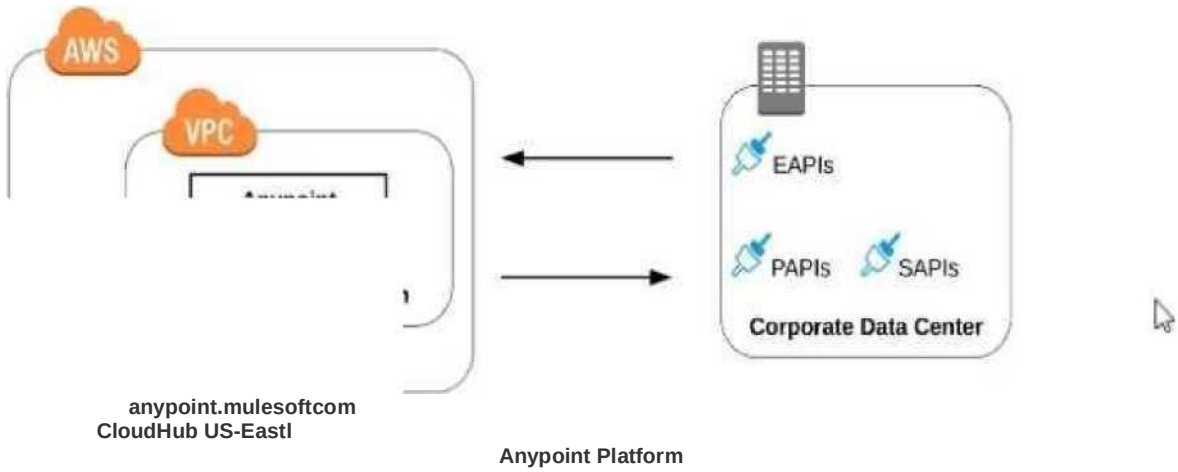
Correct Answer: Ask the Marketing Department to interact with a mocking implementation of the API using the automatically generated API Console.
As per MuleSoft's IT Operating Model:
> > API consumers need NOT wait until the full API implementation is ready.
> > NO technical test-suites needs to be shared 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.



Question: 16

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.

Reference:
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-2018

https://help.mulesoft.com/s/article/Runtime-Manager-cannot-manage-On-Prem-Applications-and-

Servers-from-US-Control-Plane-June-25th-2019
https://help.mulesoft.com/s/article/On-premise-Runtimes-Appear-Disconnected-in-Runtime- Manager-May-29th-2018

Question: 17

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 enforement 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

Question: 18

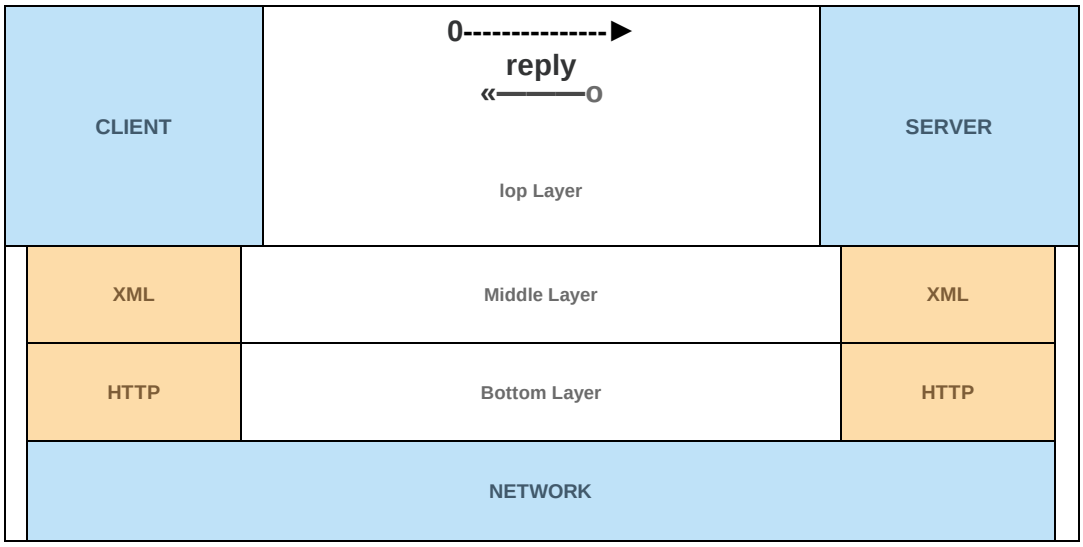
Refer to the exhibit.

What is a valid API in the sense of API-led connectivity and application networks?

A) Java RMI over TCP

B) Java RMI over TCP

operation()



C) CORBA over HOP

D) XML over UDP

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

Answer: D

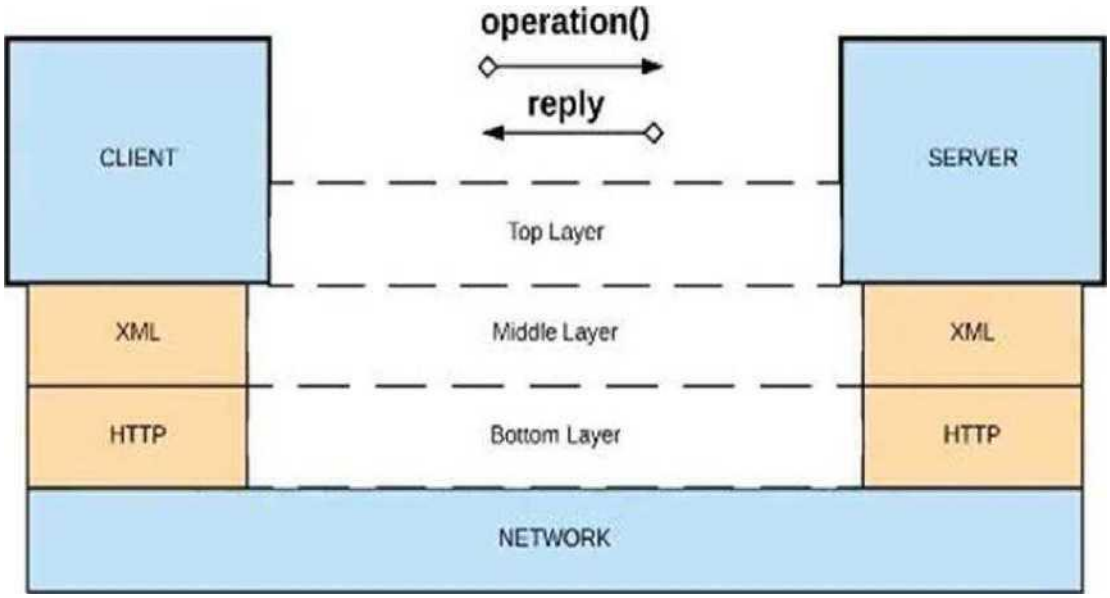
Explanation:

Correct Answer: XML over HTTP

> > API-led connectivity and Application Networks urge to have the APIs on HTTP based protocols for building most effective APIs and networks on top of them.

> > The HTTP based APIs allow the platform to apply various varities of policies to address many NFRs

> > The HTTP based APIs also allow to implement many standard and effective implementation patterns that adhere to HTTP based w3c rules.



Bottom of Form

Top of Form

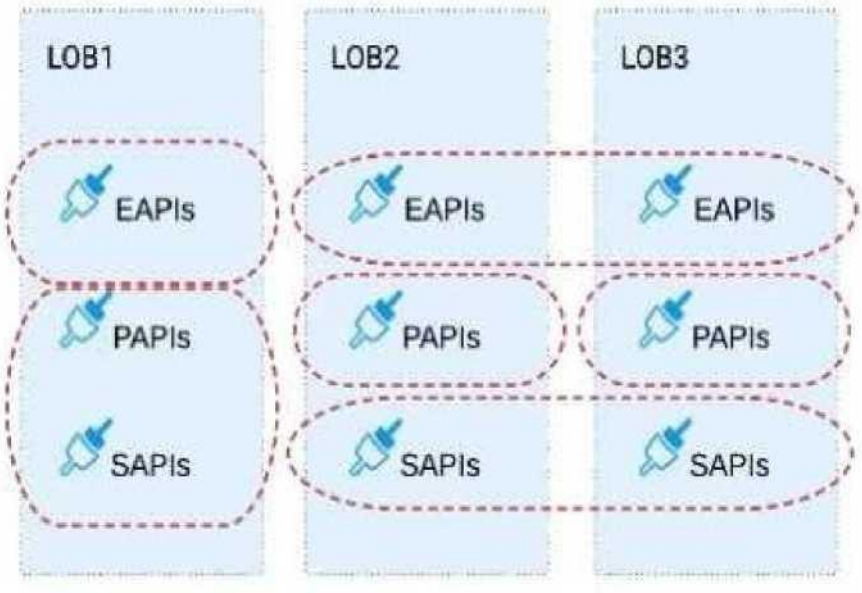
Question: 19

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

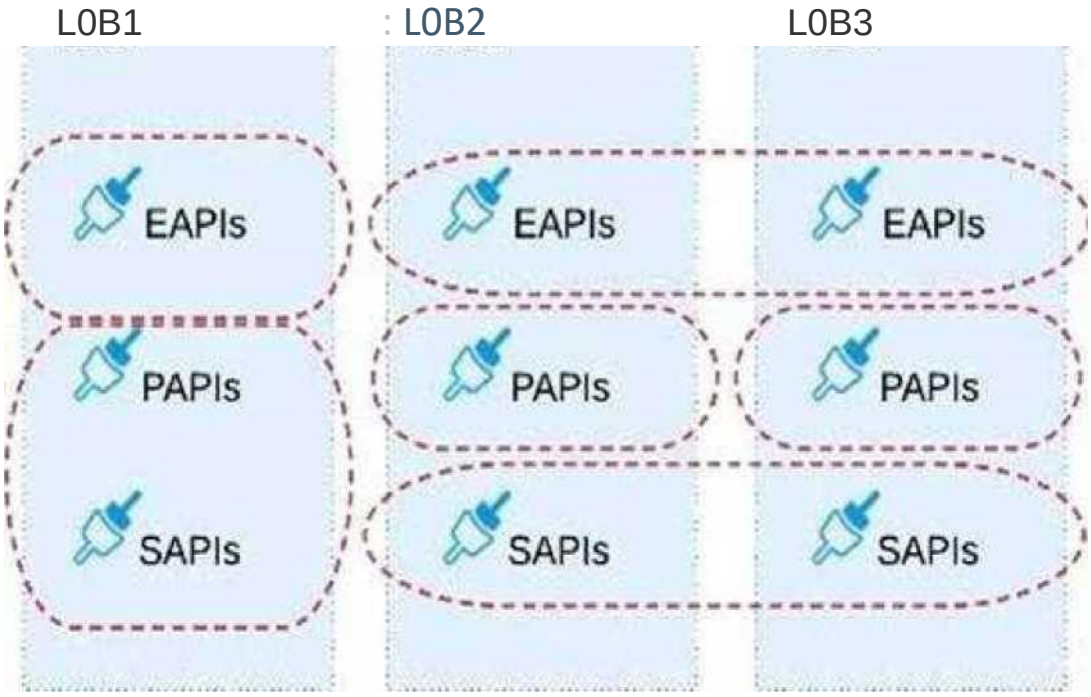
A. Option A B. Option B C. Option C D. Option D

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.



Question: 20

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.

Question: 21

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.

Question: 22

Refer to the exhibit.

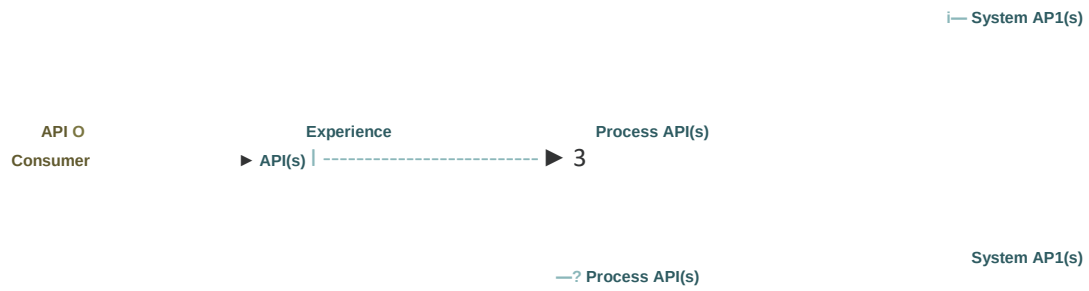
What is the best way to decompose one end-to-end business process into a collaboration of Experience, Process, and System APIs?

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:

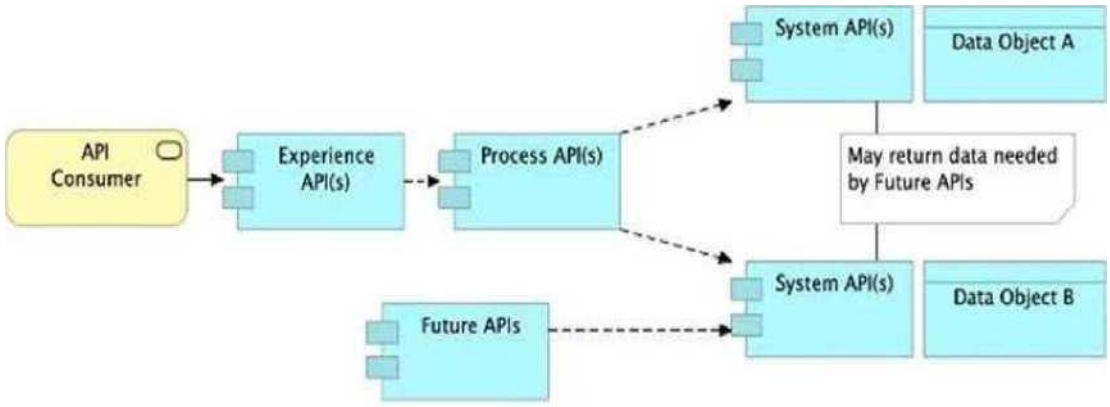
Correct Answer: Allow System APIs to return data that is NOT currently required by the identified Process or Experience APIs.

> > 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.



Question: 23

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".
Reference: https://docs.mulesoft.com/api-manager/2.x/latest-overview-concept

Question: 24

An API implementation is deployed to CloudHub.
What conditions can be alerted on using the default Anypoint Platform functionality, where the alert conditions depend on the end-to-end request processing of the API implementation?

A. When the API is invoked by an unrecognized API client
B. When a particular API client invokes the API too often within a given time period
C. When the response time of API invocations exceeds a threshold
D. When the API receives a very high number of API invocations

Answer: C

Explanation:

Correct Answer: When the response time of API invocations exceeds a threshold

> > Alerts can be setup for all the given options using the default Anypoint Platform functionality
> > However, the question insists on an alert whose conditions depend on the end-to-end request processing of the API implementation.
> > Alert w.r.t "Response Times" is the only one which requires end-to-end request processing of API implementation in order to determine if the threshold is exceeded or not.
Reference: https://docs.mulesoft.com/api-manager/2.x/using-api-alerts

Question: 25

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.
Reference:
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-Another-Cloudhub-Application
https://help.mulesoft.com/s/question/0D52T00004mXXULSA4/multiple-http-listerners-on-cloudhub-one-with-port-9090

Question: 26

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

Question: 27

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/

Question: 28

What is the most performant out-of-the-box solution in Anypoint Platform to track transaction state in an asynchronously executing long-running process implemented as a Mule application deployed to multiple CloudHub workers?

A. Redis distributed cache
B. java.util.WeakHashMap
C. Persistent Object Store
D. File-based storage

Answer: C

Explanation:
Correct Answer: Persistent Object Store

> > Redis distributed cache is performant but NOT out-of-the-box solution in Anypoint Platform
> > File-storage is neither performant nor out-of-the-box solution in Anypoint Platform
> > java.util.WeakHashMap needs a completely custom implementation of cache from scratch using Java code and is limited to the JVM where it is running. Which means the state in the cache is not worker aware when running on multiple workers. This type of cache is local to the worker. So, this is neither out-of-the-box nor worker-aware among multiple workers on cloudhub.
https://www.baeldung.com/java-weakhashmap
> > Persistent Object Store is an out-of-the-box solution provided by Anypoint Platform which is performant as well as worker aware among multiple workers running on CloudHub.
https://docs.mulesoft.com/object-store/
50, Persistent Object Store is the right answer.

Question: 29

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

Reference:
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

Question: 30

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

E. 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.

F. 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.

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

Question: 31

An API experiences a high rate of client requests (TPS) vwth small message paytoads. 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.

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

Reference: https://docs.mulesoft.com/api-manager/2.x/rate-limiting-and-throttling-sla-based-policies

Question: 32

A code-centric API documentation environment should allow API consumers to investigate and execute API client source code that demonstrates invoking one or more APIs as part of representative scenarios.

What is the most effective way to provide this type of code-centric API documentation environment using Anypoint Platform?

A. Enable mocking services for each of the relevant APIs and expose them via their Anypoint Exchange entry

B. Ensure the APIs are well documented through their Anypoint Exchange entries and API Consoles and share these pages with all API consumers

C. Create API Notebooks and include them in the relevant Anypoint Exchange entries

D. Make relevant APIs discoverable via an Anypoint Exchange entry

Answer: C

Explanation:

Correct Answer: Create API Notebooks and Include them in the relevant Anypoint exchange entries

E. API Notebooks are the one on Anypoint Platform that enable us to provide code-centric API documentation
Reference: https://docs.mulesoft.com/exchange/to-use-api-notebook

Bottom of Form
Top of Form

Question: 33

Refer to the exhibit. An organization is running a Mule standalone runtime and has configured Active Directory as the Anypoint Platform external Identity Provider. The organization does not have budget for other system components.

What policy should be applied to all instances of APIs in the organization to most effecuvelyKestrict access to a specific group of internal users?

A. Apply a basic authentication - LDAP policy; the internal Active Directory will be configured as the LDAP source for authenticating users
B. Apply a client ID enforcement policy; the specific group of users will configure their client applications to use their specific client credentials
C. Apply an IP whitelist policy; only the specific users' workstations will be in the whitelist
D. Apply an OAuth 2.0 access token enforcement policy; the internal Active Directory will be configured as the OAuth server

Answer: A

Explanation:

Correct Answer: Apply a basic authentication - LDAP policy; the internal Active Directory will be configured as the LDAP source for authenticating users.

> > IP Whitelisting does NOT fit for this purpose. Moreover, the users workstations may not necessarily have static IPs in the network.
> > OAuth 2.0 enforcement requires a client provider which isn't in the organizations system components.
> > It is not an effective approach to let every user create separate client credentials and configure those for their usage.
The effective way it to apply a basic authentication - LDAP policy and the internal Active Directory will be configured as the LDAP source for authenticating users.
Reference: https://docs.mulesoft.com/api-manager/2.x/basic-authentication-ldap-concept

Question: 34

What is a best practice when building System APIs?

A. Document the API using an easily consumable asset like a RAML definition

B. Model all API resources and methods to closely mimic the operations of the backend system
C. Build an Enterprise Data Model (Canonical Data Model) for each backend system and apply it to System APIs
D. Expose to API clients all technical details of the API implementation's interaction wifch the backend system

Answer: B

Explanation:

Correct Answer: Model all API resources and methods to closely mimic the operations of the backend system.

> > There are NO fixed and straight best practices while opting data models for APIs. They are completly contextual and depends on number of factors. Based upon those factors, an enterprise can choose if they have to go with Enterprise Canonical Data Model or Bounded Context Model etc.
> > One should NEVER expose the technical details of API implementation to their API clients. Only the API interface/ RAML is exposed to API clients.
> > It is true that the RAML definitions of APIs should be as detailed as possible and should reflect most of the documentation. However, just that is NOT enough to call your API as best documented API. There should be even more documentation on Anypoint Exchange with API Notebooks etc. to make and create a developer friendly API and repository..
> > The best practice always when creating System APIs is to create their API interfaces by modeling

their resources and methods to closely reflect the operations and functionalities of that backend system.

Question: 35

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.
Reference: https://help.mulesoft.com/s/article/DDos-Dos-at

Question: 36

An organization makes a strategic decision to move towards an IT operating model that emphasizes consumption

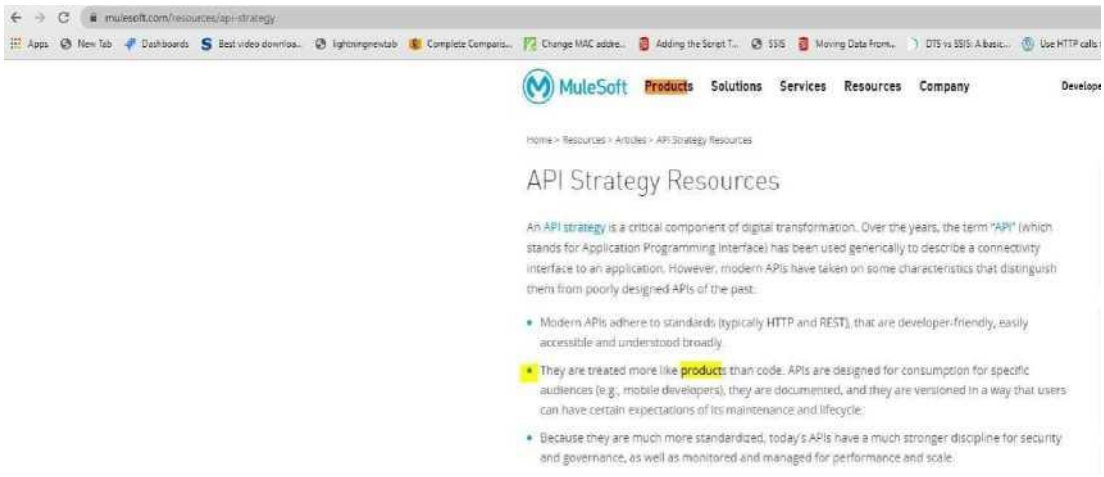
of reusable IT assets using modern APIs (as defined by MuleSoft).
What best describes each modern API in relation to this new IT operating model?

A. Each modern API has its own software development lifecycle, which reduces the need for documentation and automation
B. Each modem 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

Question: 37

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. IPwhitellst

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 occassionally 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.

Question: 38

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.

Question: 39

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 white 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.

Question: 40

What Mule application deployment scenario requires using Anypoint Platform Private Cloud Edition or Anypoint Platform for Pivotal Cloud Foundry?

A. When it Is required to make ALL applications highly available across multiple data centers
B. When it is required that ALL APIs are private and NOT exposed to the public cloud

C. When regulatory requirements mandate on-premises processing of EVERY data item, including meta-data
D. When ALL backend systems in the application network are deployed in the organization's intranet

Answer: C

Explanation:

Correct Answer: When regulatory requirements mandate on-premises processing of EVERY data item, including meta-data.

We need NOT require to use Anypoint Platform PCE or PCF for the below. So these options are OUT.
> > We can make ALL applications highly available across multiple data centers using CloudHub too.
> > We can use Anypoint VPN and tunneling from CloudHub to connect to ALL backend systems in the application network that are deployed in the organization's intranet.
> > We can use Anypoint VPC and Firewall Rules to make ALL APIs private and NOT exposed to the public cloud.
Only valid reason in the given options that requires to use Anypoint Platform PCE/ PCF is - When regulatory requirements mandate on-premises processing of EVERY data item, including meta-data.

Question: 41

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

Question: 42

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
Reference: https://docs.mulesoft.com/api-manager/2.x/defining-sla-tiers#defining-a-tier

Question: 43

What is a typical result of using a fine-grained rather than a coarse-grained API deployment model to implement a given business process?

A. A decrease in the number of connections within the application network supporting the business process
B. A higher number of discoverable API-related assets in the application network
C. A better response time for the end user as a result of the APIs being smaller in scope and complexity
D. An overall tower usage of resources because each fine-grained API consumes less resources

Answer: B

Explanation:

Correct Answer: A higher number of discoverable API-related assets in the application network.

>> We do NOT get faster response times in fine-grained approach when compared to coarse-grained approach.
>> In fact, we get faster response times from a network having coarse-grained APIs compared to a network having fine-grained APIs model. The reasons are below.
Fine-grained approach:
1. will have more APIs compared to coarse-grained
2. So, more orchestration needs to be done to achieve a functionality in business process.
3. Which means, lots of API calls to be made. So, more connections will needs to be established. So, obviously more hops, more network i/o, more number of integration points compared to coarsegrained approach where fewer APIs with bulk functionality embedded in them.
4. That is why, because of all these extra hops and added latencies, fine-grained approach will have bit more response times compared to coarse-grained.
5. Not only added latencies and connections, there will be more resources used up in fine-grained approach due to more number of APIs.
That's why, fine-grained APIs are good in a way to expose more number of resuable assets in your network and make them discoverable. However, needs more maintenance, taking care of integration points, connections, resources with a little compromise w.r.t network hops and response times.

Question: 44

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

Question: 45

An organization is deploying their new implementation of the OrderStatus System API to multiple workers in CloudHub. This API fronts the organization's on-premises Order Management System, which is accessed by the API implementation over an IPsec tunnel.
What type of error typically does NOT result in a service outage of the OrderStatus System API?

A. A CloudHub worker fails with an out-of-memory exception
B. API Manager has an extended outage during the initial deployment of the API implementation
C. The AWS region goes ofline with a major network failure to the relevant AWS data centers
D. The Order Management System is Inaccessible due to a network outage in the organization's onpremises data center

Answer: A

Explanation:

Correct Answer: A CloudHub worker fails with an out-of-memory exception.

> > An AWS Region itself going down will definitely result in an outage as it does not matter how many workers are assigned to the Mule App as all of those in that region will go down. This is a complete downtime and outage.
> > Extended outage of API manager during initial deployment of API implementation will of course cause issues in proper application startup itself as the API Autodiscovery might fail or API policy templates and polices may not be downloaded to embed at the time of applicaiton startup etc... there are many reasons that could cause issues.
> > A network outage onpremises would of course cause the Order Management System not accessible and it does not matter how many workers are assigned to the app they all will fail and cause outage for sure.
The only option that does NOT result in a service outage is if a cloudhub worker fails with an out-ofmemory exception. Even if a worker fails and goes down, there are still other workers to handle the requests and keep the API UP and Running. So, this is the right answer.

Question: 46

An Order API must be designed that contains significant amounts of integration logic and involves the invocation of the Product API.
The power relationship between Order API and Product API is one of "Customer/Supplier", because the Product API is used heavily throughout the organization and is developed by a dedicated development team located in the office of the CTO.
What strategy should be used to deal with the API data model of the Product API within the Order API?

A. Convince the development team of the Product API to adopt the API data model of the Order API such that the integration logic of the Order API can work with one consistent internal data model B. Work with the API data types of the Product API directly when implementing the integration logic of the Order API such that the Order API uses the same (unchanged) data types as the Product API C. Implement an anti-corruption layer in the Order API that transforms the Product API data model into internal data types of the Order API
D. Start an organization-wide data modeling initiative that will result in an Enterprise Data Model that will then be used in both the Product API and the Order API

Answer: C

Explanation:

Correct Answer: Convince the development team of the product API to adopt the API data model of the Order API such that integration logic of the Order API can work with one consistent internal data model
Key details to note from the given scenario:
>> Power relationship between Order API and Product API is customer/supplier
So, as per below rules of "Power Relationships", the caller (in this case Order API) would request for features to the called (Product API team) and the Product API team would need to accomodate those requests.

Question: 47

An API implementation is being designed that must invoke an Order API, which is known to repeatedly experience downtime.
For this reason, a fallback API is to be called when the Order API is unavailable.
What approach to designing the invocation of the fallback API provides the best resilience?

A. Search Anypoint Exchange for a suitable existing fallback API, and then implement invocations to this fallback API in addition to the Order API
B. Create a separate entry for the Order API in API Manager, and then invoke this API as a fallback API if the primary Order API is unavailable
C. Redirect client requests through an HTTP 307 Temporary Redirect status code to the fallback API whenever the Order API is unavailable
D. Set an option in the HTTP Requester component that invokes the Order API to instead invoke a fallback API whenever an HTTP 4xx or 5xx response status code is returned from the Order API

Answer: A

Explanation:

Correct Answer: Search Anypoint exchange for a suitable existing fallback API, and then implement invocations to this fallback API in addition to the order API

> > It is not ideal and good approach, until unless there is a pre-approved agreement with the API clients that they will receive a HTTP 3xx temporary redirect status code and they have to implement fallback logic their side to call another API.

> > Creating separate entry of same Order API in API manager would just create an another instance of it on top of same API implementation. So, it does NO GOOD by using clone od same API as a fallback API. Fallback API should be ideally a different API implementation that is not same as primary one.

> > There is NO option currently provided by Anypoint HTTP Connector that allows us to invoke a fallback API when we receive certain HTTP status codes in response.

The only statement TRUE in the given options is to Search Anypoint exchange for a suitable existing fallback API, and then implement invocations to this fallback API in addition to the order API.

Question: 48

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 the 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.

Question: 49

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

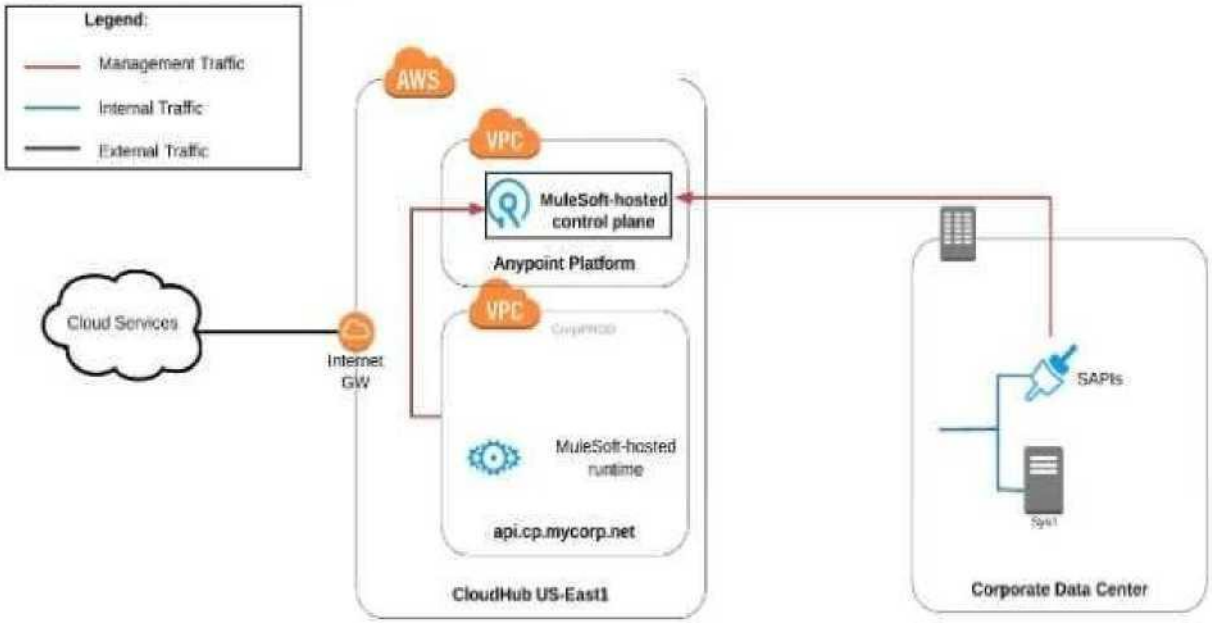
What is the best way to configure and use Anypoint Platform to support integrations with both the cloud-based SaaS systems and on-premises systems?

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.

Question: 50

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

Explanation:

Question: 51

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 te 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.

Question: 52

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

Explanation:
Reference: https://docs.mulesoft.com/exchange/to-change-raml-version

Question: 53

Mule applications that implement a number of REST APIs are deployed to their own subnet that is inaccessible from outside the organization.
External business-partners need to access these APIs, which are only allowed to be invoked from a separate subnet dedicated to partners - called Partner-subnet. This subnet is accessible from the

public internet, which allows these external partners to reach it.
Anypoint Platform and Mule runtimes are already deployed in Partner-subnet. These Mule runtimes can already access the APIs.
What is the most resource-efficient solution to comply with these requirements, while having the least impact on other applications that are currently using the APIs?

A. Implement (or generate) an API proxy Mule application for each of the APIs, then deploy the API proxies to the Mule runtimes
B. Redeploy the API implementations to the same servers running the Mule runtimes
C. Add an additional endpoint to each API for partner-enablement consumption
D. Duplicate the APIs as Mule applications, then deploy them to the Mule runtimes

Answer: A

Explanation:

Question: 54

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, …)

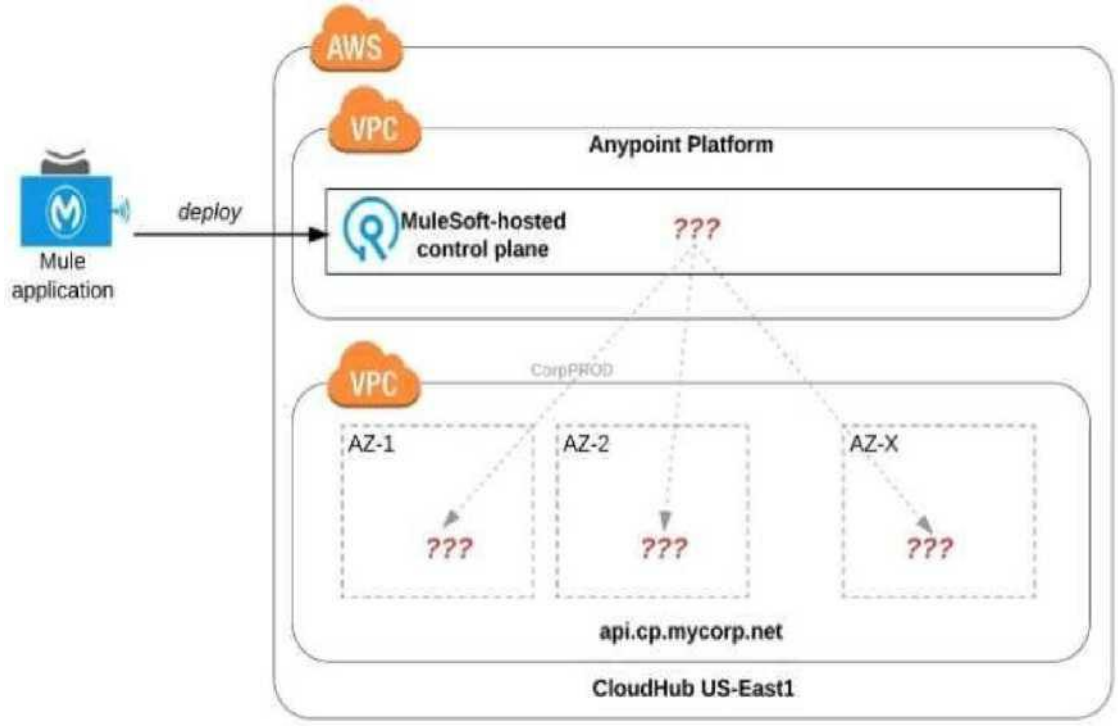
> > Allows the usual API policies to be applied to System APIs

> > 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

Question: 55

Refer to the exhibit.



An organization uses one specific CloudHub (AWS) region for all CloudHub deployments.

How are CloudHub workers assigned to availability zones (AZs) when the organization's Mule applications are deployed to CloudHub in that region?

A. Workers belonging to a given environment are assigned to the same AZ within that region
B. AZs are selected as part of the Mule application's deployment configuration
C. Workers are randomly distributed across available AZs within that region
D. An AZ is randomly selected for a Mule application, and all the Mule application's CloudHub workers are assigned to that one AZ

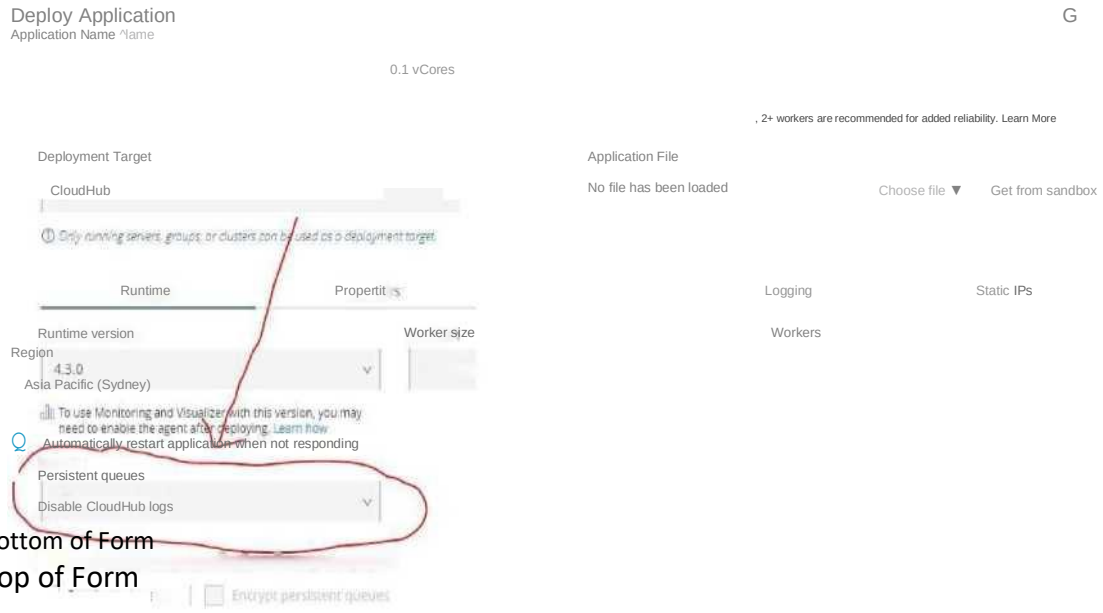
Answer: D

Explanation:

Correct Answer: Workers are randomly distributed across available AZs within that region.

> > Currently, we only have control to choose which AWS Region to choose but there is no control at all using any configurations or deployment options to decide what Availability Zone (AZ) to assign to what worker.
> > There are NO fixed or implicit rules on platform too w.r.t assignment of AZ to workers based on environment or application.
> > They are completely assigned in random. However, cloudhub definitely ensures that HA is achieved by assigning the workers to more than on AZ so that all workers are not assigned to same AZ for same application.

Reference: https://help.mulesoft.com/s/question/0D52T000051rqDj/one-cloudhub-aws-region-how- cloudhub-workers-are-assigned-to-availability-zones-azs-



Bottom of Form
Top of Form

Question: 56

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.

Reference: https://docs.mulesoft.com/mule-runtime/3.9/testing-strategies#integration-testing

Question: 57

An API implementation is updated. When must the RAML definition of the API also be updated?

A. When the API implementation changes the structure of the request or response messages
B. When the API implementation changes from interacting with a legacy backend system deployed on-premises to a modern, cloud-based (SaaS) system
C. When the API implementation is migrated from an older to a newer version of the Mule runtime
D. When the API implementation is optimized to improve its average response time

Answer: A

Explanation:

Correct Answer: When the API implementation changes the structure of the request or response messages

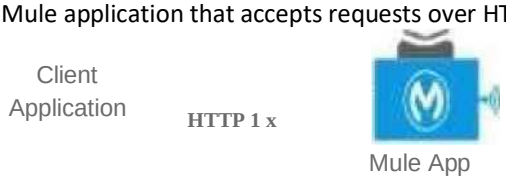
>> RAML definition usually needs to be touched only when there are changes in the request/response schemas or in any traits on API.

>> It need not be modified for any internal changes in API implementation like performance tuning, backend system migrations etc..

Question: 58

What Mule application can have API policies applied by
Anypoint Platform to the endpoint exposed by that Mule application?

A) A Mule application that accepts requests over HTTP/1.x



B) A Mule application that accepts JSON requests over TCP but is NOT required to provide a response

C) A Mute application that accepts JSON requests over WebSocket

D) A Mule application that accepts gRPC requests over HTTP/2

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

Answer: A

Explanation:

Correct Answer: Option A

>> Anypoint API Manager and API policies are applicable to all types of HTTP/1.x APIs.
>> They are not applicable to WebSocket APIs, HTTP/2 APIs and gRPC APIs
Reference: https://docs.mulesoft.com/api-manager/2.x/using-policies

Question: 59

What Anypoint Connectors support transactions?

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

Answer: A

Explanation:

Question: 60

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

Explanation:

Question: 61

A retail company is using an Order API to accept new orders. The Order API uses a JMS queue to submit orders to a backend order management service. The normal load for orders is being handled using two (2) CloudHub workers, each configured with 0.2 vCore. The CPU load of each CloudHub worker normally runs well below 70%. However, several times during the year the Order API gets four times (4x) the average number of orders. This causes the CloudHub worker CPU load to exceed 90% and the order submission time to exceed 30 seconds. The cause, however, is NOT the backend order management service, which still responds fast enough to meet the response SLA for the Order API. What is the MOST resource-efficient way to configure the Mule application's CloudHub deployment to help the company cope with this performance challenge?

A. Permanently increase the size of each of the two (2) CloudHub workers by at least four times (4x) to one (1) vCore
B. Use a vertical CloudHub autoscaling policy that triggers on CPU utilization greater than 70%
C. Permanently increase the number of CloudHub workers by four times (4x) to eight (8) CloudHub workers
D. Use a horizontal CloudHub autoscaling policy that triggers on CPU utilization greater than 70%

Answer: D

Explanation:

Correct Answer: Use a horizontal CloudHub autoscaling policy that triggers on CPU utilization greater than 70%

The scenario in the question is very clearly stating that the usual traffic in the year is pretty well handled by the existing worker configuration with CPU running well below 70%. The problem occurs only "sometimes" occasionally when there is spike in the number of orders coming in.

So, based on above, We neither need to permanently increase the size of each worker nor need to permanently increase the number of workers. This is unnecessary as other than those "occasional" times the resources are idle and wasted.
We have two options left now. Either to use horizontal Cloudhub autoscaling policy to automatically increase the number of workers or to use vertical Cloudhub autoscaling policy to automatically increase the vCore size of each worker.
Here, we need to take two things into consideration:
1. CPU
2. Order Submission Rate to JMS Queue

> > From CPU perspective, both the options (horizontal and vertical scaling) solves the issue. Both helps to bring down the usage below 90%.
> > However, If we go with Vertical Scaling, then from Order Submission Rate perspective, as the application is still being load balanced with two workers only, there may not be much improvement in the incoming request processing rate and order submission rate to JMS queue. The throughput would be same as before. Only CPU utilization comes down.
> > But, if we go with Horizontal Scaling, it will spawn new workers and adds extra hand to increase the throughput as more workers are being load balanced now. This way we can address both CPU and Order Submission rate.
Hence, Horizontal CloudHub Autoscaling policy is the right and best answer.

Question: 62

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.

Question: 63

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.

Reference: https://docs.mulesoft.com/runtime-fabric/1.7/

Question: 64

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 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.

Question: 65

An Anypoint Platform organization has been configured with an external identity provider (IdP) for identity management and client management. What credentials or token must be provided to Anypoint CLI to execute commands against the Anypoint Platform APIs?

A. The credentials provided by the IdP for identity management
B. The credentials provided by the IdP for client management
C. An OAuth 2.0 token generated using the credentials provided by the IdP for client management
D. An OAuth 2.0 token generated using the credentials provided by the IdP for identity management

Answer: A

Explanation:

Correct Answer: The credentials provided by the IdP for identity management

Reference: https://docs.mulesoft.com/runtime-manager/anypoint-platform-cli#authentication
>> There is no support for OAuth 2.0 tokens from client/identity providers to authenticate via Anypoint CLI. Only possible tokens are "bearer tokens" that too only generated using Anypoint Organization/Environment Client Id and Secret from https://anypoint.mulesoft.com/accounts/login. Not the client credentials of client provider. So, OAuth 2.0 is not possible. More over, the token is mainly for API Manager purposes and not associated with a user. You can NOT use it to call most APIs (for example Cloudhub and etc) as per this Mulesoft Knowledge article.
>> The other option allowed by Anypoint CLI is to use client credentials. It is possible to use client credentials of a client provider but requires setting up Connected Apps in client management but such details are not given in the scenario explained in the question.

>> So only option left is to use user credentials from identify provider

Question: 66

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 reusablity 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.

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

Question: 67

Version 3.0.1 of a REST API implementation represents time values in PST time using ISO 8601 hh:mm:ss format. The API implementation needs to be changed to instead represent time values in CEST time using ISO 8601 hh:mm:ss format. When following the semver.org semantic versioning specification, what version should be assigned to the updated API implementation?

A. 3.0.2
B. 4.0.0
C. 3.1.0
D. 3.0.1

Answer: B

Explanation:

Correct Answer: 4.0.0
As per semver.org semantic versioning
specification:
Given a version number MAJOR.MINOR.PATCH, increment the:
- MAJOR version when you make incompatible API changes.
- MINOR version when you add functionality in a backwards compatible manner.
- PATCH version when you make backwards compatible bug fixes.

As per the scenario given in the question, the API implementation is completely changing its behavior. Although

the format of the time is still being maintained as hh:mm:ss and there is no change in schema w.r.t format, the API will start functioning different after this change as the times are going to come completely different.
Example: Before the change, say, time is going as 09:00:00 representing the PST. Now on, after the change, the same time will go as 18:00:00 as Central European Summer Time is 9 hours ahead of Pacific Time.
- > This may lead to some uncertain behavior on API clients depending on how they are handling the times in the API response. All the API clients need to be informed that the API functionality is going to change and will return in CEST format. So, this considered as a MAJOR change and the version of API for this new change would be 4.0.0

Question: 68

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.

Question: 69

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)

Reference: https://docs.mulesoft.com/mule-runtime/3.9/testing-strategies
As per general IT testing practice and MuleSoft recommended practice, Integration and Performance tests should be done on full end to end setup for right evaluation. Which means all end systems should be connected while doing the tests. So, these options are OUT and we are left with Unit Tests and Functional Tests.
As per attached reference documentation from MuleSoft:
Unit Tests - are limited to the code that can be realistically exercised without the need to run it inside Mule itself. So good candidates are Small pieces of modular code, Sub Flows, Custom transformers, Custom components, Custom expression evaluators etc.
Functional Tests - are those that most extensively exercise your application configuration. In these tests, you have the freedom and tools for simulating happy and unhappy paths. You also have the possibility to create stubs for target services and make them success or fail to easily simulate happy and unhappy paths respectively.
As the scenario in the question demands for API implementation to be tested before deployment to Staging and also clearly indicates that there is enough/ sufficient amount of mock data to test the various components of API implementations with no active connections to the backend systems, Unit Tests are the one to be used to incorporate this mocked data.

Question: 70

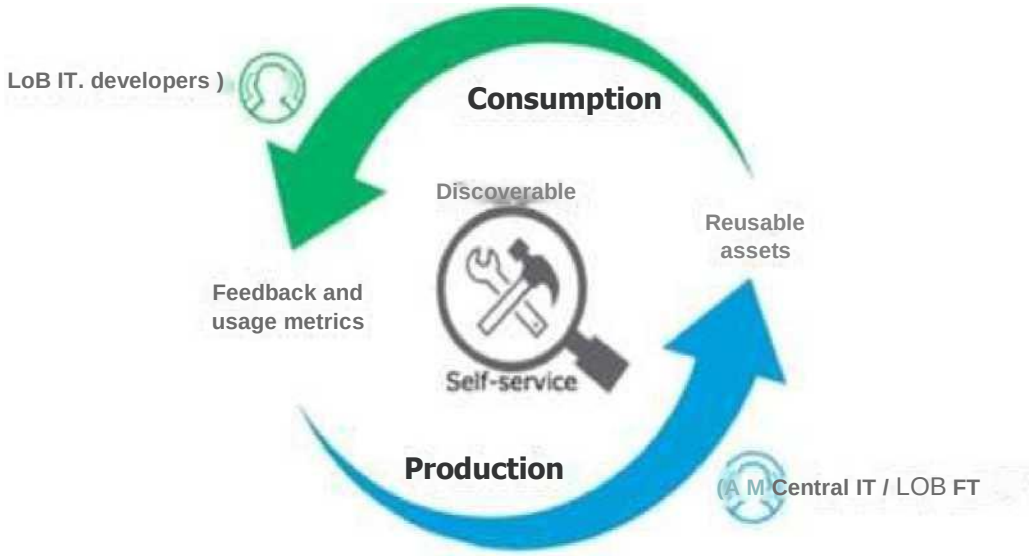
Which of the below, when used together, makes the IT Operational Model effective?

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 resuable 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.



Question: 71

Which of the following sequence is correct?

A. API Client implementes logic to call an API >> API Consumer requests access to API >> API Implementation routes the request to >> API

B. API Consumer requests access to API >> API Client implementes logic to call an API >> API routes the request to >> API Implementation

C. API Consumer implementes logic to call an API >> API Client requests access to API >> API Implementation routes the request to >> API

D. API Client implementes logic to call an API >> API Consumer requests access to API >> API routes the request to >> API Implementation

Answer: B

Explanation:

Correct Answer: API Consumer requests access to API >> API Client implementes logic to call an API >> API routes the request to >> API Implementation

> > API consumer does not implement any logic to invoke APIs. It is just a role. So, the option stating "API Consumer implementes logic to call an API" is INVALID.

> > API Implementation does not route any requests. It is a final piece of logic where functionality of target systems is exposed. So, the requests should be routed to the API implementation by some other entity. So, the options stating "API Implementation routes the request to >> API" is INVALID >> The statements in one of the options are correct but sequence is wrong. The sequence is given as "API Client implementes logic to call an API >> API Consumer requests access to API >> API routes the request to >> API Implementation". Here, the statements in the options are VALID but sequence is WRONG.

> > Right option and sequence is the one where API consumer first requests access to API on Anypoint Exchange and obtains client credentials. API client then writes logic to call an API by using the access client credentials requested by API consumer and the requests will be routed to API implementation via the API which is managed by API Manager.

Question: 72

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.

Question: 73

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.

Question: 74

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.

Question: 75

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

Question: 76

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.

Question: 77

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, comparitively, very less calls will be allowed into the DR environment API opposed to its primary environment. With this in mind, lets 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.

Question: 78

A company uses a hybrid Anypoint Platform deployment model that combines the EU control plane with customer-hosted Mule runtimes. After successfully testing a Mule API implementation in the Staging environment, the Mule API implementation is set with environment-specific properties and must be promoted to the Production environment. What is a way that MuleSoft recommends to configure the Mule API implementation and automate

its promotion to the Production environment? A.
Bundle properties files for each environment into the Mule API implementation's deployable archive, then promote the Mule API implementation to the Production environment using Anypoint CLI or the Anypoint Platform REST APIsB.
B.
Modify the Mule API implementation's properties in the API Manager Properties tab, then promote the Mule API implementation to the Production environment using API Manager C.
Modify the Mule API implementation's properties in Anypoint Exchange, then promote the Mule API implementation to the Production environment using Runtime Manager D.
Use an API policy to change properties in the Mule API implementation deployed to the Staging environment and another API policy to deploy the Mule API implementation to the Production environment

Answer: A

Explanation:

Correct Answer: Bundle properties files for each environment into the Mule API implementation's deployable archive, then promote the Mule API implementation to the Production environment using Anypoint CLI or the Anypoint Platform REST APIs

> > Anypoint Exchange is for asset discovery and documentation. It has got no provision to modify the properties of Mule API implementations at all.

> > API Manager is for managing API instances, their contracts, policies and SLAs. It has also got no provision to modify the properties of API implementations.

> > API policies are to address Non-functional requirements of APIs and has again got no provision to modify the properties of API implementations.

So, the right way and recommended way to do this as part of development practice is to bundle properties files for each environment into the Mule API implementation and just point and refer to respective file per environment.

Question: 79

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.

Question: 80

A company requires Mule applications deployed to CloudHub to be isolated between non-production and production environments. This is so Mule applications deployed to non-production environments can only access backend systems running in their customer-hosted non-production environment, and so Mule applications deployed to production environments can only access backend systems running in their customer-hosted production environment. How does MuleSoft recommend modifying Mule applications, configuring environments, or changing infrastructure to support this type of per-environment isolation between Mule applications and backend systems?

A.

Modify properties of Mule applications deployed to the production Anypoint Platform environments to prevent access from non-production Mule applications

B.

Configure firewall rules in the infrastructure inside each customer-hosted environment so that only IP addresses from the corresponding Anypoint Platform environments are allowed to communicate with corresponding backend systems

C.

Create non-production and production environments in different Anypoint Platform business groups D.

Create separate Anypoint VPCs for non-production and production environments, then configure connections to the backend systems in the corresponding customer-hosted environments

Answer: D

Explanation:

Correct Answer: Create separate Anypoint VPCs for non-production and production environments, then configure connections to the backend systems in the corresponding customer-hosted environments.

> > Creating different Business Groups does NOT make any difference w.r.t accessing the non-prod and prod customer-hosted environments. Still they will be accessing from both Business Groups unless process network restrictions are put in place.

> > We need to modify or couple the Mule Application Implementations with the environment. In fact, we should never implements application coupled with environments by binding them in the properties. Only basic things like endpoint URL etc should be bundled in properties but not environment level access restrictions.

> > IP addresses on CloudHub are dynamic until unless a special static addresses are assigned. So it is not possible to setup firewall rules in customer-hosted infrastrcture. More over, even if static IP addresses are assigned, there could be 100s of applications running on cloudhub and setting up rules for all of them would be a hectic task, non-maintainable and definitely got a good practice.

> > The best practice recommended by Mulesoft (In fact any cloud provider), is to have your Anypoint VPCs seperated for Prod and Non-Prod and perform the VPC peering or VPN tunneling for these Anypoint VPCs to respective Prod and Non-Prod customer-hosted environment networks.

Reference: https://docs.mulesoft.com/runtime-manager/virtual-private-cloud

Bottom of Form

Top of Form

Question: 81

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 APIs 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 API producer should be requested to run the old version in parallel with the new one
B. The API producer should be contacted to understand the change to existing functionality
C. The API client code only needs to be changed if it needs to take advantage of the new features
D. The API clients need to update the code on their side and need to do full regression

Answer: C

Explanation:

Question: 82

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

Question: 83

A retail company with thousands of stores has an API to receive data about purchases and insert it into a single database. Each individual store sends a batch of purchase data to the API about every 30 minutes. The API implementation uses a database bulk insert command to submit all the purchase data to a database using a custom JDBC driver provided by a data analytics solution provider. The API implementation is deployed to a single CloudHub worker. The JDBC driver processes the data into a set of several temporary disk files on the CloudHub worker, and then the data is sent to an analytics engine using a proprietary protocol. This process usually takes less than a few minutes. Sometimes a request fails. In this case, the logs show a message from the JDBC driver indicating an out-of-filespace message. When the request is resubmitted, it is successful. What is the best way to try to resolve this throughput issue?

A. se a CloudHub autoscaling policy to add CloudHub workers
B. Use a CloudHub autoscaling policy to increase the size of the CloudHub worker
C. Increase the size of the CloudHub worker(s)
D. Increase the number of CloudHub workers

Answer: D

Explanation:

Correct Answer: Increase the size of the CloudHub worker(s)

The key details that we can take out from the given scenario are:

> > API implementation uses a database bulk insert command to submit all the purchase data to a database
> > JDBC driver processes the data into a set of several temporary disk files on the CloudHub worker
> > Sometimes a request fails and the logs show a message indicating an out-of-file-space message Based on above details:
> > Both auto-scaling options does NOT help because we cannot set auto-scaling rules based on error messages. Auto-scaling rules are kicked-off based on CPU/Memory usages and not due to some given error or disk space issues.
> > Increasing the number of CloudHub workers also does NOT help here because the reason for the failure is not due to performance aspects w.r.t CPU or Memory. It is due to disk-space.
> > Moreover, the API is doing bulk insert to submit the received batch data. Which means, all data is handled by ONE worker only at a time. So, the disk space issue should be tackled on "per worker" basis. Having multiple workers does not help as the batch may still fail on any worker when disk is out of space on that particular worker.

Therefore, the right way to deal this issue and resolve this is to increase the vCore size of the worker so that a new worker with more disk space will be provisioned.

Question: 84

Due to a limitation in the backend system, a system API can only handle up to 500 requests per second. What is the best type of API policy to apply to the system API to avoid overloading the backend system?

A. Rate limiting
B. HTTP caching
C. Rate limiting - SLA based
D. Spike control

Answer: D

Explanation:

Correct Answer: Spike control

> > First things first, HTTP Caching policy is for purposes different than avoiding the backend system from overloading. So this is OUT.

> > Rate Limiting and Throttling/ Spike Control policies are designed to limit API access, but have different intentions.
> > Rate limiting protects an API by applying a hard limit on its access.
> > Throttling/ Spike Control shapes API access by smoothing spikes in traffic.
That is why, Spike Control is the right option.

Question: 85

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.

Question: 86

The application network is recomposable: it is built for change because it "bends but does not break"

A. TRUE
B. FALSE

Answer: A

Explanation:

> > Application Network is a disposable architecture.
> > Which means, it can be altered without disturbing entire architecture and its components.
> > It bends as per requirements or design changes but does not break
Reference: https://www.mulesoft.com/resources/api/what-is-an-application-network

Question: 87

A system API has a guaranteed SLA of 100 ms per request. The system API is deployed to a primary environment as well as to a disaster recovery (DR) environment, with different DNS names in each environment. An upstream process API invokes the system API and the main goal of this process API is to respond to client requests in the least possible time. In what order should the system APIs be invoked, and what changes should be made in order to speed up the response time for requests from the process API?

A.

In parallel, invoke the system API deployed to the primary environment and the system API deployed to the DR environment, and ONLY use the first response

B.

In parallel, invoke the system API deployed to the primary environment and the system API deployed to the DR environment using a scatter-gather configured with a timeout, and then merge the responses

C.

Invoke the system API deployed to the primary environment, and if it fails, invoke the system API deployed to the DR environment

D.

Invoke ONLY the system API deployed to the primary environment, and add timeout and retry logic to avoid intermittent failures

Answer: A

Explanation:

Correct Answer: In parallel, invoke the system API deployed to the primary environment and the system API deployed to the DR environment, and ONLY use the first response.

> > The API requirement in the given scenario is to respond in least possible time.

> > The option that is suggesting to first try the API in primary environment and then fallback to API in DR environment would result in successful response but NOT in least possible time. So, this is NOT a right choice of implementation for given requirement.

> > Another option that is suggesting to ONLY invoke API in primary environment and to add timeout and retries may also result in successful response upon retries but NOT in least possible time. So, this is also NOT a right choice of implementation for given requirement.

> > One more option that is suggesting to invoke API in primary environment and API in DR environment in parallel using Scatter-Gather would result in wrong API response as it would return merged results and moreover, Scatter-Gather does things in parallel which is true but still completes its scope only on finishing all routes inside it. So again, NOT a right choice of implementation for given requirement

The Correct choice is to invoke the API in primary environment and the API in DR environment parallelly, and using ONLY the first response received from one of them.

Question: 88

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.

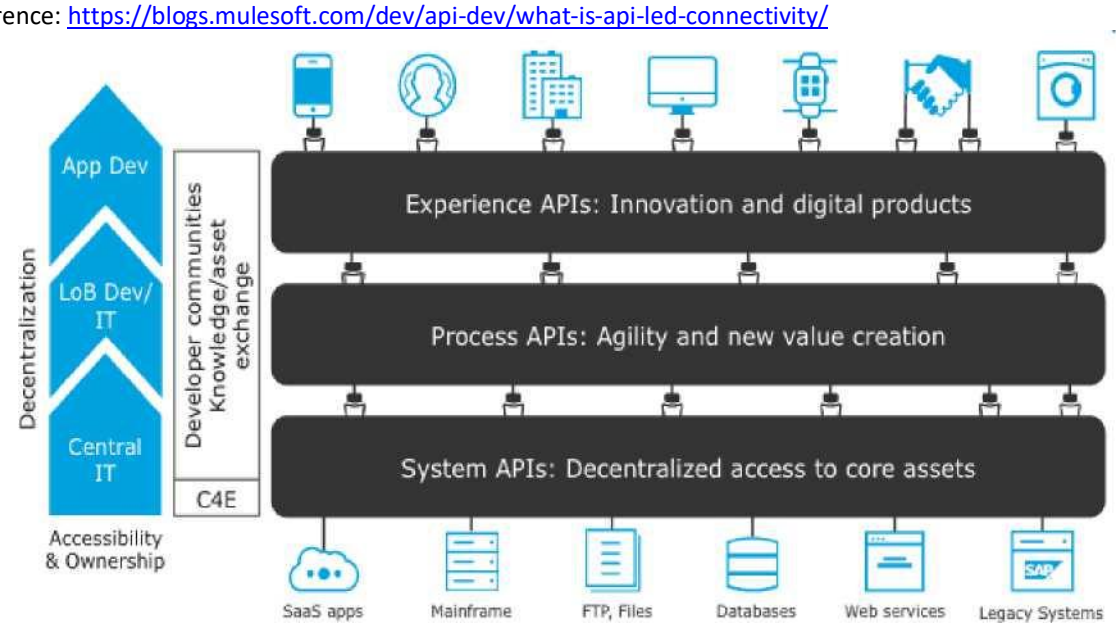
API-led connectivity is a technology which enabled us to implement Experience, Process and System layer based APIs

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.

Reference: https://blogs.mulesoft.com/dev/api-dev/what-is-api-led-connectivity/



Question: 89

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. Upon finishing creation of all the possible assets then inform the LOBs in the organization to start using them

B.

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

C.

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 Answe r:

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.

Reference: https://www.youtube.com/watch?v=U0FpYMnMjmM

Question: 90

A Mule application exposes an HTTPS endpoint and is deployed to three CloudHub workers that do not use static IP addresses. The Mule application expects a high volume of client requests in short time periods. What is the most cost-effective infrastructure component that should be used to serve the high volume of client requests?

A. A customer-hosted load balancer
B. The CloudHub shared load balancer
C. An API proxy
D. Runtime Manager autoscaling

Answer: B

Explanation:

Correct Answer: The CloudHub shared load balancer
The scenario in this question can be split as below:
> > There are 3 CloudHub workers (So, there are already good number of workers to handle high volume of requests)
> > The workers are not using static IP addresses (So, one CANNOT use customer load-balancing solutions without static IPs)
> > Looking for most cost-effective component to load balance the client requests among the workers.
Based on the above details given in the scenario:
> > Runtime autoscaling is NOT at all cost-effective as it incurs extra cost. Most over, there are already 3 workers running which is a good number.
> > We cannot go for a customer-hosted load balancer as it is also NOT most cost-effective (needs custom load balancer to maintain and licensing) and same time the Mule App is not having Static IP Addresses which limits from going with custom load balancing.
> > An API Proxy is irrelevant there as it has no role to play w.r.t handling high volumes or load balancing.
So, the only right option to go with and fits the purpose of scenario being most cost-effective is - using a CloudHub Shared Load Balancer.

Question: 91

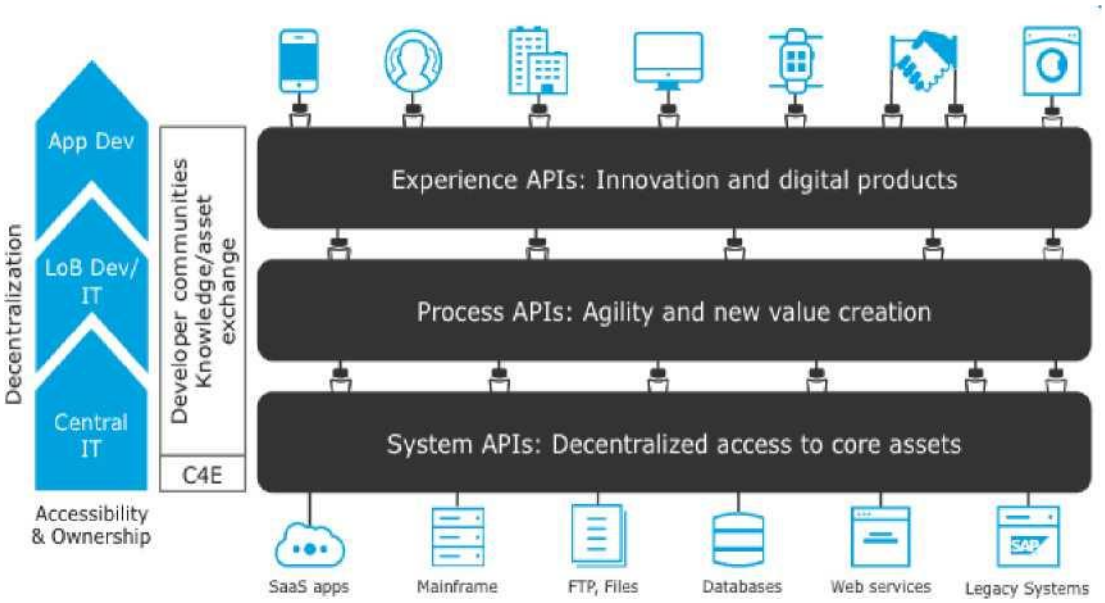
Which layer in the API-led connectivity focuses on unlocking key systems, legacy systems, data sources etc and exposes the functionality?

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

Answer: C

Explanation:

Correct Answer: System Layer



The APIs used in an API-led approach to connectivity fall into three categories:
System APIs – these usually access the core systems of record and provide a means of insulating the user from the complexity or any changes to the underlying systems. Once built, many users, can access data without any need to learn the underlying systems and can reuse these APIs in multiple
projects.
Process APIs – These APIs interact with and shape data within a single system or across systems (breaking down data silos) and are created here without a dependence on the source systems from which that data originates, as well as the target channels through which that data is delivered. Experience APIs – Experience APIs are the means by which data can be reconfigured so that it is most easily consumed by its intended audience, all from a common data source, rather than setting up separate point-to-point integrations for each channel. An Experience API is usually created with API- first design principles where the API is designed for the specific user experience in mind.

Question: 92

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 Answe r:
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

Reference:
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/

Question: 93

What Anypoint Platform Capabilities listed below fall under APIs and API Invocations/Consumers category? Select TWO.

A. API Operations and Management
B. API Runtime Execution and Hosting
C. API Consumer Engagement
D. API Design and Development

Answer: D

Explanation:

Correct Answers: API Design and Development and API Runtime Execution and Hosting

> > 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

Platform Capabilities

APi design and development

API clients and API Implementations

API runtime execution and hosting

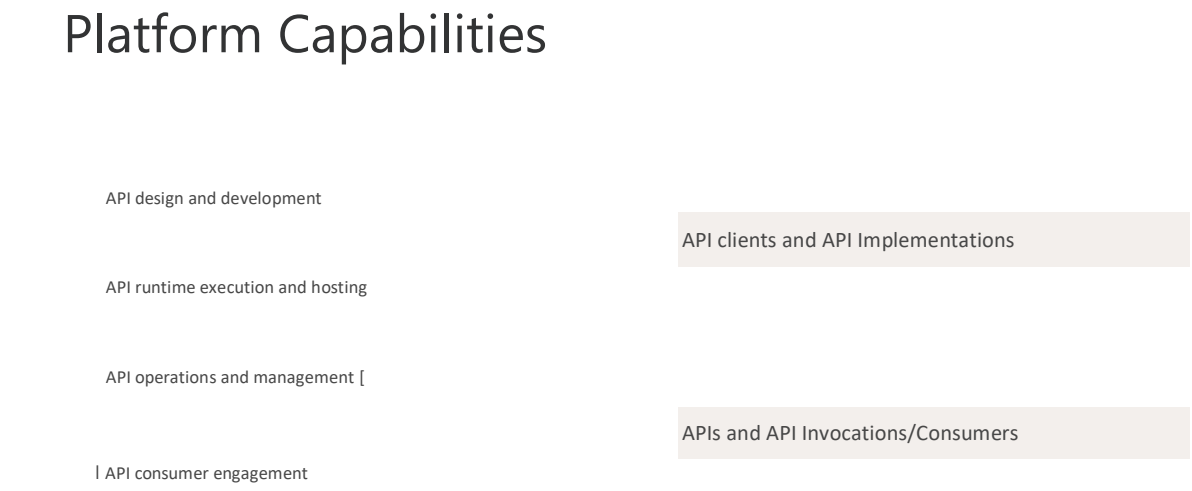
API operations and management

APIs and API Invocations/Consumers

API consumer engagement

Correct Answers: API Operations and Management and 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



Bottom of Form
Top of Form

Question: 94

What are 4 important Platform Capabilities offered by Anypoint Platform?

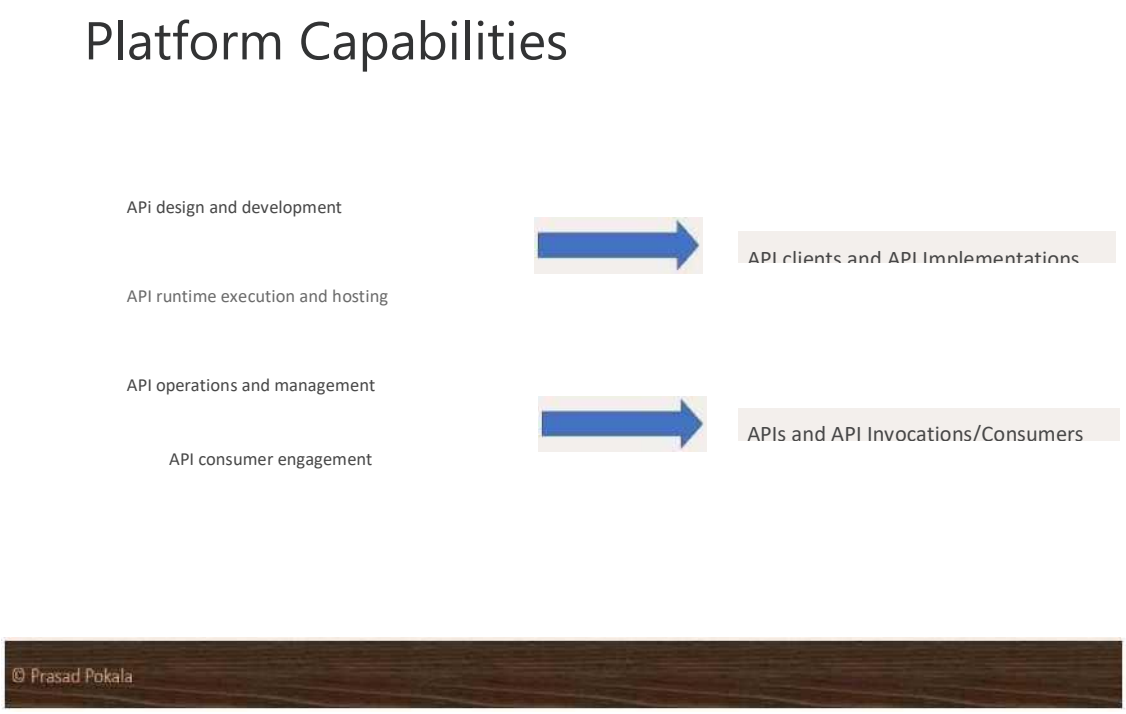
A. API Versioning, API Runtime Execution and Hosting, API Invocation, API Consumer Engagement
B. API Design and Development, API Runtime Execution and Hosting, API Versioning, API Deprecation
C. API Design and Development, API Runtime Execution and Hosting, API Operations and Management, API Consumer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



Question: 95

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.

Question: 96

A large lending company has developed an API to unlock data from a database server and web server. The API has been deployed to Anypoint Virtual Private Cloud (VPC) on CloudHub 1.0.
The database server and web server are in the customer's secure network and are not accessible through the public internet. The database server is in the customer's AWS
VPC, whereas the web server is in the customer's on-premises corporate data center.
How can access be enabled for the API to connect with the database server and the web server?

A. Set up VPC peering with AWS VPC and a VPN tunnel to the customer's on-premises corporate data center
B. Set up VPC peering with AWS VPC and the customer's on-premises corporate data center
C. Setup a transit gateway to the customer's on-premises corporate data center through AWS VPC
D. Set up VPC peering with the customer's on-premises corporate data center and a VPN tunnel to AWS VPC

Answer: A

Explanation:
Scenario Overview:
The API resides in Anypoint Virtual Private Cloud (VPC) on CloudHub 1.0, where it requires connectivity to both an AWS-hosted database server and an on-premises web server.
Both servers are isolated from the public internet: the database server is within the customer's AWS VPC, and the web server is within the customer's on-premises corporate data center.
Connectivity Requirements:
To connect to the AWS database server from the API in Anypoint VPC, VPC peering is ideal. This would allow a private network connection between the MuleSoft Anypoint VPC and the customer's AWS VPC, enabling secure, direct access to the database.
To connect to the on-premises web server, a VPN tunnel is suitable. This would establish a secure, encrypted connection from the Anypoint VPC to the customer's corporate data center, allowing secure data flow between the API and the on-premises web server.
Analysis of Options:
Option A (Correct Answer): Setting up VPC peering with AWS VPC enables private network connectivity with the database server, while a VPN tunnel to the on-premises data center allows
secure access to the web server. This combination meets the requirements for secure, controlled access to both resources.
Option B: VPC peering alone would not suffice because it does not support a connection from the Anypoint VPC directly to an on-premises network. A VPN is necessary for on-premises access.
Option C: Setting up a transit gateway would provide connectivity within AWS but would not enable direct

connectivity from CloudHub to the on-premises network.

Option D: VPC peering with the on-premises network is not possible because VPC peering is typically used to connect two VPCs, not a VPC with an on-premises network.

Conclusion:

Option A is the correct choice, as it provides a complete solution by using VPC peering for AWS VPC connectivity and a VPN tunnel for secure on-premises connectivity. This setup aligns with Anypoint Platform best practices for connecting Anypoint VPCs to both AWS-hosted and on-premises systems, ensuring secure, controlled access to both the database and web server.

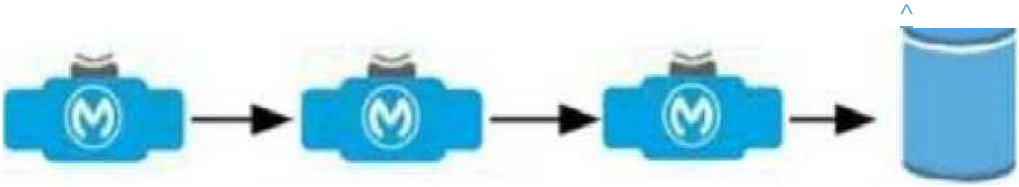
For more detailed reference, MuleSoft documentation on Anypoint VPC peering and VPN connectivity provides additional context on best practices for setting up these connections within a hybrid network infrastructure.

Question: 97

An Order API triggers a sequence of other API calls to look up details of an order's items in a backend inventory database. The Order API calls the OrderItems process API, which calls the Inventory system API. The Inventory system API performs database operations in the back-end inventory database.

The network connection between the Inventory system API and the database is known to be unreliable and hang at unpredictable times.

Where should a two-second timeout be configured in the API processing sequence so that the Order API never waits more than two seconds for a response from the OrderItems process API?



A. In the OrderItems process API implementation
B. In the Order API implementation
C. In the Inventory system API implementation
D. In the inventory database

Answer: A

Explanation:

Understanding the API Flow and Timeout Requirement:

The Order API initiates a call to the OrderItems process API, which in turn calls the Inventory system API to fetch details from the inventory database.

The requirement specifies that the Order API should not wait more than two seconds for a response from the OrderItems process API, even if there are delays further down the chain (between Inventory system API and the database).

Choosing the Appropriate Timeout Location:

Setting the timeout at the OrderItems process API level ensures that if the Inventory system API takes longer than two seconds to respond, the OrderItems process API will terminate the request and send a timeout response back to the Order API. This prevents the Order API from waiting indefinitely due to the unreliable connection

to the database.

If the timeout were set in the Inventory system API or database, it would not help the Order API directly, as the OrderItems process API would still be waiting for a response.

Detailed Analysis of Each Option:

Option A (Correct Answer): Setting the timeout in the OrderItems process API allows it to control how long it waits for a response from the Inventory system API. If the Inventory system API does not respond within two seconds, the OrderItems process API can terminate the call and return a timeout response to the Order API, meeting the requirement.

Option B: Setting the timeout in the Order API would not limit the wait time at the OrderItems process API level, meaning the OrderItems process API could still wait indefinitely for the Inventory system API, leading to a longer delay.

Option C: Setting the timeout in the Inventory system API only affects the connection to the database and does not influence how long the OrderItems process API waits for the Inventory system API's response.

Option D: Setting a timeout in the database is not feasible in this context since database timeouts are typically configured for database operations and would not directly control the API response times in the overall API chain.

Conclusion:

Option A is the best choice, as it ensures that the OrderItems process API does not hold the Order API longer than the required two seconds, even if the downstream connection to the database hangs. This configuration aligns with MuleSoft best practices for setting timeouts in API orchestration to manage dependencies and prevent delays across a chain of API calls.

For additional information on timeout settings, refer to MuleSoft documentation on handling timeouts and API orchestration best practices.

Question: 98

A circuit breaker strategy is planned in order to meet the goal of improved response time and demand on a downstream API.

* Circuit Open: More than 10 errors per minute for three minutes
* Circuit Half-Open: One error per minute
* Circuit Closed: Less than one error per minute for five minutes

Out of several proposals from the engineering team, which option will meet this goal?

A. Create a custom policy that implements the circuit breaker and includes policy template expressions for the required settings

B. Create Anypoint Monitoring alerts for Circuit Open/Closed configurations, and then implement a retry strategy for Circuit Half-Open configuration

C. Add the Circuit Breaker policy to the API instance, and configure the required settings

D. Implement the strategy in a Mule application, and provide the settings in the YAML configuration

Answer: C

Explanation:

Understanding Circuit Breaker Policy:

A circuit breaker is a design pattern used to detect failures and prevent an application from continually trying to execute a failing operation. In this case, it will help improve response time and reduce demand on the downstream API.

The specified configuration includes conditions for opening, half-opening, and closing the circuit based on error

rates over time:
Circuit Open: Triggered if there are more than 10 errors per minute for three consecutive minutes. Circuit Half-Open: The circuit transitions to half-open if there is one error per minute.
Circuit Closed: The circuit closes if the error rate is less than one error per minute for five minutes. Evaluating the Options:
Option A: Creating a custom policy with template expressions could work, but it would require custom development. Since the Anypoint Platform already has a Circuit Breaker policy available, this would be a less efficient and more complex solution.
Option B: Anypoint Monitoring alerts can be used for monitoring the API, but they do not provide circuit-breaking functionality. Additionally, implementing a retry strategy for the half-open state is not sufficient to achieve the required circuit breaker behavior.
Option C (Correct Answer): Adding the Circuit Breaker policy to the API instance on Anypoint Platform allows you to set up circuit-breaking conditions directly. This approach uses the built-in Circuit Breaker policy, where you can configure parameters such as error thresholds and time intervals to match the requirements. This solution is efficient, reliable, and leverages Anypoint's out- of-the-box capabilities.
Option D: Implementing the strategy within a Mule application with a YAML configuration could be complex and less manageable. Additionally, it does not leverage Anypoint Platform's built-in Circuit Breaker policy, which is more suited to this scenario.
Conclusion:
Option C is the correct choice, as it leverages Anypoint Platform's Circuit Breaker policy. This solution allows for configuring thresholds and time intervals as specified, improving response time and reducing demand on the downstream API while utilizing Anypoint's managed policy feature.
Refer to MuleSoft's documentation on implementing the Circuit Breaker policy in API Manager for detailed configuration guidance.

Question: 99

An online store's marketing team has noticed an increase in customers leaving online baskets without checking out. They suspect a technology issue is at the root cause of the baskets being left behind. They approach the Center for Enablement to ask for help identifying the issue. Multiple APIs from across all the layers of their application network are involved in the shopping application. Which feature of the Anypoint Platform can be used to view metrics from all involved APIs at the same time?

A. Custom dashboards
B. Built-in dashboards
C. Functional monitoring
D. API Manager

Answer: B

Explanation:
Understanding the Need for Cross-API Monitoring:
The Center for Enablement (C4E) needs to investigate potential technical issues across multiple APIs in the application network that may be causing customers to abandon their carts.
This requires a solution that allows viewing metrics across several APIs in real-time to identify any performance issues or bottlenecks.
Evaluating Anypoint Platform Features:
Built-in Dashboards: Anypoint Platform provides built-in dashboards in Anypoint Monitoring, allowing teams to view metrics from multiple APIs in a single interface. This feature is designed to monitor API performance, latency,

errors, and throughput, and is ideal for tracking performance across all layers of the application network.
Custom Dashboards: While custom dashboards allow for more tailored views, the built-in dashboards already aggregate metrics for multiple APIs, making it unnecessary to build a custom solution for this scenario.
Functional Monitoring: This feature is used to set up tests to monitor specific API functionality and uptime but is not suited for tracking metrics across multiple APIs in real-time.
API Manager: API Manager primarily focuses on managing API policies, contracts, and access control rather than providing detailed, real-time metrics across the entire application network.

Conclusion:

Option B (Built-in dashboards) is the best choice because it provides a comprehensive view of metrics from all APIs involved, enabling the C4E team to quickly identify any issues that may be contributing to abandoned shopping carts.
Refer to MuleSoft's documentation on Anypoint Monitoring and built-in dashboards for more details on configuring and using these dashboards effectively.

Question: 100

Which component monitors APIs and endpoints at scheduled intervals, receives reports about whether tests pass or fail, and displays statistics about API and endpoint performance?

A. API Analytics
B. Anypoint Monitoring dashboards
C. APT Functional Monitoring
D. Anypoint Runtime Manager alerts

Answer: C

Explanation:

Understanding API Functional Monitoring:
API Functional Monitoring is a feature within MuleSoft's Anypoint Platform that enables users to monitor the health and performance of APIs and endpoints by running functional tests at scheduled intervals.
It checks whether APIs are functioning as expected by running test calls and then evaluating if the response meets the desired conditions. This is particularly useful for testing endpoint availability,
checking for specific data in responses, and measuring API performance over time.

Component Features:

Scheduled Intervals: Functional monitoring allows configuring tests to run at regular intervals, such as every minute, hour, or day, depending on the monitoring requirements.
Reports on Test Pass/Fail Status: After each test run, API Functional Monitoring reports whether the API passed or failed the test conditions.
Performance Statistics: It displays metrics like average response time, success rate, and error rates, giving insights into API health and performance.

Evaluating the Options:

Option A (API Analytics): API Analytics provides insights on API usage and metrics but does not involve scheduled tests for pass/fail status or endpoint health checks.
Option B (Anypoint Monitoring Dashboards): These dashboards display API metrics but do not actively test API endpoints or provide pass/fail reporting on a scheduled basis.
Option C (Correct Answer): API Functional Monitoring fits the description, as it is designed to monitor API and endpoint health with scheduled test runs and display statistics about performance. Option D (Anypoint Runtime

Manager Alerts): Runtime Manager alerts notify users of issues with application status but do not actively test endpoints at scheduled intervals.

Conclusion:

Option C (API Functional Monitoring) is the correct answer because it provides the necessary tools to test API functionality, monitor endpoint health, and display performance statistics in real-time.

Refer to MuleSoft documentation on API Functional Monitoring for further guidance on setting up and configuring these tests in Anypoint Platform.

Question: 101

An eCommerce company is adding a new Product Details feature to their website, A customer will launch the product catalog page, a new Product Details link will
appear by product where they can click to retrieve the product detail description. Product detail data is updated with product update releases, once or twice a year, Presently the database response time has been very slow due to high volume.

What action retrieves the product details with the lowest response time, fault tolerant, and consistent data?

A. Select the product details from a database in a Cache scope and return them within the API response
B. Select the product details from a database and put them in Anypoint MQ; the Anypoint MO subseriber will receive the product details and return them within the API response
C. Use an object store to store and retrieve the product details originally read from a database and return them within the API response
D. Select the product details from a database and return them within the API response

Answer: C

Explanation:

Scenario Analysis:

The eCommerce company's Product Details feature requires low response time and consistent data for a feature where data rarely changes (only once or twice a year).

The database response time is slow due to high volume, so querying the database directly on each request would lead to poor performance and higher response times.

Optimal Solution Requirements:

Low Response Time: Data retrieval should be fast and not depend on database performance.

Fault Tolerance and Data Consistency: Cached or stored data should be consistent and resilient in case of database unavailability, as the product details data changes infrequently.

Evaluating the Options:

Option A: Using a Cache scope would temporarily store the product details in memory, which could improve performance but might not be suitable for infrequent updates (only twice a year), as cache expiration policies typically require shorter durations.

Option B: Storing product details in Anypoint MQ and then retrieving it through a subscriber is not suitable for this use case. Anypoint MQ is better for messaging rather than as a data storage mechanism.

Option C (Correct Answer): Using an object store to store and retrieve product details is ideal. Object stores in MuleSoft are designed for persistent storage of key-value pairs, which allows storing data retrieved from the database initially. This provides quick, consistent access without querying the database on every request, aligning with requirements for low response time, fault tolerance, and data consistency.

Option D: Selecting data directly from the database for each request would not meet the performance

requirement due to known slow response times from the database.

Conclusion:

Option C is the best answer, as using an object store allows caching the infrequently updated product details. This approach reduces the dependency on the database, significantly improving response time and ensuring consistent data.

Refer to MuleSoft documentation on Object Store v2 and best practices for data caching to implement this solution effectively.

Question: 102

A company deploys Mule applications with default configurations through Runtime Manager to customer-hosted Mule runtimes. Each Mule application is an API implementation that exposes RESTful interfaces to API clients. The Mule runtimes are managed by the MuleSoft-hosted control plane. The payload is never used by any Logger components.

When an API client sends an HTTP request to a customer-hosted Mule application, which metadata or data (payload) is pushed to the MuleSoft-hosted control plane?

A. Only the data
B. No data
C. The data and metadata
D. Only the metadata

Answer: D

Explanation:

Understanding the Data Flow Between Mule Runtimes and Control Plane:

When Mule applications are deployed on customer-hosted Mule runtimes, the MuleSoft-hosted control plane (Anypoint Platform) can monitor and manage these applications. However, due to data privacy and security, the control plane only collects specific types of information.

Typically, only metadata about the request and response (such as headers, status codes, and timestamps) is sent to the MuleSoft-hosted control plane. The actual payload data is not transmitted unless explicitly configured, ensuring that sensitive data remains within the customer's network. Evaluating the Options:

Option A (Only the data): This is incorrect because the payload data itself is not automatically sent to the control plane in default configurations.

Option B (No data): This is incorrect as well; while the payload is not sent, metadata is still collected and sent to the control plane.

Option C (The data and metadata): This option is incorrect because data (payload) is not transmitted to the control plane by default.

Option D (Correct Answer): Only the metadata is sent to the MuleSoft-hosted control plane by default, aligning with MuleSoft's design to prioritize security and data privacy for customer-hosted runtimes.

Conclusion:

Option D is the correct answer, as by default, only metadata is sent to the MuleSoft-hosted control plane, and not the payload. This configuration is designed to protect sensitive data from being exposed outside the customer's hosted environment.

For more details, refer to MuleSoft's documentation on telemetry data collected in customer-hosted Mule runtimes and the MuleSoft control plane.

Question: 103

A customer wants to monitor and gain insights about the number of requests coming in a given time period as well as to measure key performance indicators
(response times, CPU utilization, number of active APIs).
Which tool provides these data insights?

A. Anypoint Monitoring
B. APT Manager
C. Runtime Alerts
D. Functional Monitoring

Answer: A

Explanation:
Understanding Anypoint Monitoring and Its Capabilities:
Anypoint Monitoring provides comprehensive visibility into Mule applications, offering metrics and analytics such as request counts, response times, CPU utilization, memory usage, and other key performance indicators (KPIs).
This tool is designed to help teams monitor API usage, troubleshoot issues, and optimize application performance.
Evaluating the Options:
Option A (Correct Answer): Anypoint Monitoring is the ideal tool for this requirement. It provides real-time insights into metrics such as the number of requests, response times, CPU utilization, and active API usage.
Option B (API Manager): API Manager focuses on API lifecycle management, including applying policies, managing contracts, and setting access controls. It does not provide performance monitoring or KPI tracking.
Option C (Runtime Alerts): Runtime Alerts can notify users of specific conditions, like high CPU usage, but they do not provide a full suite of metrics or insights over a given time period.
Option D (Functional Monitoring): Functional Monitoring focuses on functional testing of APIs rather than performance and usage metrics. It does not provide continuous KPI tracking.
Conclusion:
Option A is the correct answer. Anypoint Monitoring is the most suitable tool to track the specified metrics, providing detailed insights into API requests, response times, CPU usage, and active API counts.
For further details, refer to MuleSoft's Anypoint Monitoring documentation on configuring dashboards and tracking performance metrics.

Question: 104

4 Production environment is running on a dedicated Virtual Private Cloud (VPC) on CloudHub 1,0, and the security team guidelines clearly state no traffic on HTTP.
Which two options support these security guidelines?
Choose 2 answers

A. Configure the HTTPS protocol in HTTP listener in the Mule application
B. Create a custom policy to apply to outgoing and incoming HTTP requests to control access to a configured API endpoint
C. Remove the entry from the VPC firewall rule

"CIDR Block": "0.00.0/0", //(Anywhev)

'Protocol'; "TCP".

'From port": 8081.

'CIDR Block" "10 111 0.0/24". it (Loco! VPC)

"Protocol"; "TCP",

"From port"; 8091.

D.

Configure the IP Blocklist policy to control access to a configured API endpoint from either a single IP address or a range of IP addresses.

E.

Add the entry in the VPC firewall rule.

1

"CIDR Block" "0 0 0.0/0". // (Anywho^)

"Protocol". "TCP"

"From port": 8081.

Answer: AC

Explanation:

Security Guidelines Overview:

The production environment is hosted on a dedicated Virtual Private Cloud (VPC) on CloudHub 1.0, with a specific requirement from the security team that no traffic should occur over HTTP. This implies that only secure HTTPS traffic should be permitted, and HTTP access (port 8081, the default HTTP port in Mule applications) should be disabled.

Evaluating the Options:

Option A (Correct Answer): Configuring the HTTPS protocol in the HTTP listener in the Mule application ensures that all traffic is encrypted and occurs over HTTPS (port 8092 by default for HTTPS on Mule applications). This directly aligns with the security guideline to prevent unencrypted HTTP traffic.

Option B: Creating a custom policy for incoming and outgoing HTTP requests could provide some control over access, but it does not enforce the use of HTTPS exclusively. This option does not disable HTTP traffic and, therefore, does not meet the guideline effectively.

Option C (Correct Answer): Removing the entry for HTTP (port 8081) in the VPC firewall rule ensures that HTTP traffic is completely blocked at the firewall level. This prevents any HTTP requests from reaching the application, adding a layer of security that complies with the guidelines.

Option D: The IP Blocklist policy controls access based on IP addresses but does not enforce the use of HTTPS. This policy does not address the specific requirement of preventing HTTP traffic.

Option E: Adding a firewall rule entry for HTTP (port 8081) would enable HTTP traffic, which directly contradicts the security guidelines. Therefore, this option should be avoided.

Conclusion:

Option A and Option C are the correct choices. Configuring the HTTPS protocol in the Mule application's HTTP listener ensures that only HTTPS traffic is allowed, and removing the firewall rule for HTTP (port 8081) blocks any HTTP traffic from reaching the application. Together, these options enforce secure traffic as required by

the security guidelines.

Refer to MuleSoft documentation on configuring HTTP listeners and managing VPC firewall rules for further details on implementing these security controls.

Question: 105

Which statement is true about Spike Control policy and Rate Limiting policy?

A. All requests are rejected after the limit is reached in Rate Limiting policy, whereas the requests are queued in Spike Control policy after the limit is reached

B. In a clustered environment, the Rate Limiting.and Spike Control policies are applied to each node in the cluster

C. To protect Experience APIs by limiting resource consumption, Rate Limiting policy must be applied

D. In order to apply Rate Limiting and Spike Control policies, a contract to bind client application and API is needed for both

Answer: B

Explanation:

Understanding Spike Control and Rate Limiting Policies:

Spike Control Policy: Limits the number of requests processed by the API in a short time to handle sudden bursts of traffic. It does not queue requests but rejects any request that exceeds the allowed burst rate.

Rate Limiting Policy: Sets a limit on the number of requests that an API can handle within a given timeframe. Once the limit is reached, additional requests are rejected.

Evaluating the Options:

Option A: Incorrect. In both Spike Control and Rate Limiting policies, requests are rejected once the limit is reached. Spike Control does not queue requests; it only controls the burst rate by rejecting excessive requests.

Option B (Correct Answer): In a clustered environment, each node independently enforces the Rate Limiting and Spike Control policies, meaning that the limits apply to each node separately. This ensures that each node can control its own resource usage independently within the cluster.

Option C: This is partially correct, as Rate Limiting is often used to protect Experience APIs, but Spike Control could also be useful in limiting resource consumption under high burst conditions.

Option D: Incorrect. Although a contract is required to enforce client-specific policies, Rate Limiting and Spike Control do not require a contract to function for general traffic control.

Conclusion:

Option B is the correct answer because, in a clustered environment, Rate Limiting and Spike Control policies apply separately to each node, helping each instance to manage its own load.

For more information, refer to MuleSoft's documentation on applying Rate Limiting and Spike Control policies in a clustered environment.

Question: 106

Several times a week, an API implementation shows several thousand requests per minute in an Anypoint Monitoring dashboard, Between these bursts, the dashboard shows between two and five requests per minute. The API implementation is running on Anypoint Runtime Fabric with two non-clustered replicas, reserved vCPU 1.0 and vCPU Limit 2.0.

An API consumer has complained about slow response time, and the dashboard shows the 99 percentile is

greater than 120 seconds at the time of the complaint. It also shows greater than 90% CPU usage during these time periods.

In manual tests in the QA environment, the API consumer has consistently reproduced the slow response time and high CPU usage, and there were no other API requests at

this time. In a brainstorming session, the engineering team has created several proposals to reduce the response time for requests.

Which proposal should be pursued first?

A. Increase the vCPU resources of the API implementation
B. Modify the API client to split the problematic request into smaller, less-demanding requests
C. Increase the number of replicas of the API implementation
D. Throttle the APT client to reduce the number of requests per minute

Answer: A

Explanation:

Scenario Analysis:

The API implementation is experiencing high CPU usage (over 90%) during bursts of requests, which correlates with slow response times, as indicated by a 99th percentile response time greater than 120 seconds.

The API implementation is running on Anypoint Runtime Fabric with two non-clustered replicas and has a reserved vCPU of 1.0 and a vCPU limit of 2.0.

The high CPU usage during bursts suggests that the current resources may not be sufficient to handle peak loads.

Evaluating the Options:

Option A (Correct Answer): Increasing the vCPU resources for each replica would provide more processing power to handle high traffic volumes, potentially reducing the response time during spikes. Since the CPU usage is consistently high during bursts, this option directly addresses the resource bottleneck.

Option B: Modifying the API client to split requests may reduce individual request load but could be complex to implement on the client side and may not fully address the high CPU issue.

Option C: Increasing the number of replicas could help distribute the load; however, with a high CPU load on each replica, adding more replicas without increasing CPU resources may not fully resolve the problem.

Option D: Throttling the client would reduce the number of requests, but this may not be acceptable if the client needs to maintain a high request rate. It also does not directly address the CPU limitations of the API implementation.

Conclusion:

Option A is the best choice as it addresses the root cause of high CPU usage by increasing the vCPU allocation, allowing the API to handle more requests efficiently. This should be pursued first before considering other options.

Refer to MuleSoft's documentation on Runtime Fabric and vCPU resource allocation for more details on optimizing API performance in high-demand environments.

Question: 107

When should idempotency be taken into account?

A. When making requests to update currently locked entities
B. When storing the results of s previous request for use in response to subsequent requests
C. When sending concurrent update requests for the same entity

D. When preventing duplicate processing from multiple sent requests

Answer: D

Explanation:
Understanding Idempotency:
Idempotency is a concept in APIs where an operation can be performed multiple times without changing the result beyond the initial application. This is particularly important for operations that may be repeated due to network retries or client errors.
When to Consider Idempotency:
Idempotency should be taken into account when there is a risk of duplicate processing due to multiple requests being sent (e.g., retries or errors). This ensures that repeated requests do not result in unintended side effects, such as creating multiple records or processing the same transaction more than once.
Evaluating the Options:
Option A: While locked entities may need special handling, this is not directly related to idempotency.
Option B: Storing results for future responses could be useful but does not relate to idempotent operations.
Option C: Concurrent requests for the same entity might require handling for conflicts, but this scenario is better suited for transaction management or concurrency control.
Option D (Correct Answer): Preventing duplicate processing from multiple requests is a key reason to implement idempotency, ensuring that repeat requests have no adverse effects.
Conclusion:
Option D is the correct answer as idempotency is specifically used to handle scenarios where duplicate requests might be sent, preventing unintended processing.
Refer to MuleSoft's documentation on best practices for idempotency in API design for more details.

Question: 108

The asset version 2.0.0 of the Order API is successfully published in Exchange and configured in API Manager with the Autodiscovery API ID correctly linked to the
API implementation, A new GET method is added to the existing API specification, and after updates, the asset version of the Order API is 2.0.1,
What happens to the Autodiscovery API ID when the new asset version is updated in API Manager?

A. The API ID changes, but no changes are needed to the API implementation for the new asset version in the API Autediscovery global element because the API ID is automatically updated B. The APL ID changes, so the API implementation must be updated with the latest API ID for the new asset version in the API Autodiscovery global element
C. The APLID does not change, so no changes to the APT implementation are needed for the new asset version in the API Autodiscovery global element
D. The APL ID does not change, but the API implementation must be updated in the AP] Autodiscovery global element to indicate the new asset version 2.0.4

Answer: C

Explanation:
Understanding API Autodiscovery in MuleSoft:
API Autodiscovery links an API implementation in Anypoint Platform with its configuration in API Manager. This is

controlled by the API ID which is set in the API Autodiscovery element in the Mule application.
The API ID remains consistent across minor updates to the API asset version in Exchange (e.g., from 2.0.0 to 2.0.1) as long as it is the same API.
Effect of Asset Version Update on API Autodiscovery:
When the asset version is updated (e.g., from 2.0.0 to 2.0.1), the API ID remains the same. Therefore, no changes are needed in the Autodiscovery configuration within the Mule application. The Autodiscovery will continue to link the API implementation to the latest version in API Manager. Evaluating the Options:
Option A: Incorrect, as the API ID does not automatically change with minor asset version updates. Option B: Incorrect, as the API ID remains the same, so no update is needed in the API implementation.
Option C (Correct Answer): The API ID does not change, so no changes are necessary in the API implementation for the new asset version.
Option D: Incorrect, as there is no need to update the API implementation in the Autodiscovery global element for minor version changes.
Conclusion:
Option C is the correct answer, as the API ID remains unchanged with minor version updates, and no changes are needed in the API Autodiscovery configuration.
Refer to MuleSoft documentation on API Autodiscovery and version management for more details.

Question: 109

Which out-of-the-box key performance indicator measures the success of a typical Center for Enablement and is immediately available in responses from Anypoint Platform APIs?

A. Per business group, the ratio of the number of production APT implementations deployed using a C1/CD pipeline to the number of production API implementations deployed manually
B. Per deployed API implementation, the amount of bandwidth consumed each day
C. Per published API, the number of developers that downloaded s version of the API specification
D. Per published API, the number of consumers that requested access to the API and have been approved in the Production environment

Answer: D

Explanation:
Center for Enablement (C4E) KPIs:
A Center for Enablement (C4E) in MuleSoft focuses on enabling self-service and reuse by providing APIs that can be consumed across the organization. A key metric of success is how many consumers are utilizing the published APIs.
The number of consumers who have requested and received access to an API indicates the level of adoption and reuse, which aligns with the goals of a C4E.
Evaluating the Options:
Option A: This metric could indicate deployment automation, but it is not a direct measure of C4E's success in enabling API reuse and consumption.
Option B: Bandwidth usage per API implementation provides insight into API traffic but does not measure C4E enablement or consumer engagement.
Option C: The number of developers downloading an API specification can be an indicator of interest but does not confirm actual usage or enablement.
Option D (Correct Answer): The number of consumers who have requested and received access to an API in

production is a key metric indicating API adoption and reuse, which aligns with C4E's goals.

Conclusion:

Option D is the correct answer as it provides a direct measure of consumer engagement and adoption, indicating the success of the C4E in promoting API usage across the organization. Refer to MuleSoft's documentation on C4E KPIs and API usage metrics for additional insights.

Question: 110

To minimize operation costs, a customer wants to use a CloudHub 1.0 solution. The customer's requirements are:

* Separate resources with two Business groups
* High-availability (HA) for all APIs
* Route traffic via Dedicated load balancer (DLBs)
* Separate environments into production and non-production

Which solution meets the customer's needs?

A. One production and one non-production Virtual Private Cloud (VPC).
Use availability zones to differentiate between Business groups.
Allocate maximum CIDR per VPCs to ensure HA across availability zones

B. One production and one non-production Virtual Private Cloud (VPC) per Business group.
Minimize CIDR aligning with projected application total.
Choose a MuleSoft CloudHub 1.0 region with multiple availability zones.
Deploy multiple workers for HA,

C. One production and one non-production Virtual Private Cloud (VPC) per Business group.
Minimize CIDR aligning with projected application total.
Divide availability zones during deployment of APIs for HA.

D. One production and one non-production Virtual Private Claud (VPC).
Configure subnet to differentiate between business groups.
Allocate maximum CIDR per VPCs to make it easier to add Child groups.
Span VPC to cover three availability zones.

Answer: B

Explanation:

Understanding the Requirements:

Business Groups: The solution must support two business groups, which typically require separate VPCs for logical separation.

High Availability (HA): Requires deploying resources across multiple availability zones.

Dedicated Load Balancer (DLB): Traffic should be routed via DLBs, which operate within VPCs on CloudHub.

Separate Environments: There needs to be separation between production and non-production environments.

Evaluating the Options:

Option A: Using a single production and non-production VPC and differentiating business groups via availability zones is not ideal as it does not provide full separation for each business group, and using maximum CIDR allocation is wasteful.

Option B (Correct Answer): Creating separate production and non-production VPCs per business group with minimized CIDR blocks, multiple availability zones, and multiple workers per application for HA meets all requirements effectively.

Option C: While this option separates VPCs per business group, it does not fully address the requirement for HA across availability zones by specifying multi-zone deployment only during API deployment, which may not guarantee redundancy.

Option D: Configuring subnets to differentiate business groups within a single production and nonproduction VPC does not fully separate the business groups, which is a requirement.

Conclusion:

Option B is the best choice as it meets the requirements for high availability, business group separation, and cost efficiency by using minimized CIDR allocations and deploying multiple workers
across availability zones.

For further reference, refer to MuleSoft's documentation on VPC configuration and high availability deployment strategies.

Question: 111

An IT Security Compliance Auditor is assessing which nonfunctional requirements (NFRs) are already being implemented to meet security measures.

* The Web API has Rate-Limiting SLA
* Basic Authentication - LDAP
* JSON Threat Protection
* TP Allowlist policies applied

Which two NFRs-are enforced?

A. The API invocations are coming from a known subnet range
B. Username/password supported to validate login credentials
C. Sensitive data is masked to prevent compromising critical information
D. The API is protected against XML invocation attacks
E. Performance expectations are to be allowed up to 1,000 requests per second

Answer: A, B

Explanation:

Understanding Nonfunctional Requirements (NFRs):

The NFRs in this context are related to security measures implemented for the Web API, such as rate limiting, LDAP-based authentication, JSON threat protection, and IP allowlist policies.

Evaluating the Options:

Option A (Correct Answer): The IP allowlist policy restricts access to known subnets, ensuring that API invocations come from a defined range of IPs.

Option B (Correct Answer): Basic Authentication with LDAP enforces a username/password validation, satisfying an NFR for identity verification.

Option C: Masking sensitive data is not part of the listed NFRs, as none of the mentioned policies address data masking.

Option D: XML threat protection is not mentioned, so this option is incorrect.

Option E: While rate-limiting implies performance control, it does not directly enforce a specific performance expectation.

Conclusion:

Options A and B are correct as they directly address the implemented security measures related to IP range restrictions and username/password authentication.

Refer to MuleSoft's documentation on API security policies for details on LDAP, rate limiting, and allowlist policies.

Question: 112

A business process is being implemented within an organization's application network. The architecture group proposes using a more coarse-grained application
network design with relatively fewer APIs deployed to the application network compared to a more fine-grained design.
Overall, which factor typically increases with a more coarse-grained design for this business process implementation and deployment compared with using a more fine-grained design?

A. The complexity of each API implementation
B. The number of discoverable assets related to APIs deployed in the application network
C. The number of possible connections between API implementations in the application network
D. The usage of network infrastructure resources by the application network

Answer: A

Explanation:
Understanding Coarse-Grained vs. Fine-Grained API Design:
A coarse-grained design consolidates multiple operations within a single API, leading to fewer APIs but with more complex implementations. Conversely, a fine-grained design breaks down functionalities into smaller, more specific APIs, resulting in simpler implementations but a larger number of APIs.
Evaluating the Options:
Option A (Correct Answer): With a coarse-grained design, each API handles more functionalities, increasing the complexity of each API implementation as it needs to manage more use cases and logic.
Option B: A coarse-grained design typically reduces the number of APIs, so fewer discoverable assets are available.
Option C: Fewer APIs generally mean fewer connections between them in the application network. Option D: Network infrastructure usage may actually decrease with fewer APIs, as there are fewer calls between APIs.
Conclusion:
Option A is the correct answer, as the complexity of each API implementation increases in a coarsegrained design due to the consolidation of multiple functionalities into single APIs.
Refer to MuleSoft's documentation on API design principles and best practices for coarse-grained vs. fine-grained API implementation.

Question: 113

A client has several applications running on the Salesforce service cloud. The business requirement for integration is to get daily data changes from Account and Case
Objects. Data needs to be moved to the client's private cloud AWS DynamoDB instance as a single JSON and the business foresees only wanting five attributes from the
Account object, which has 219 attributes (some custom) and eight attributes from the Case Object.
What design should be used to support the API/ Application data model?

A. Create separate entities for Account and Case Objects by mimicking all the attributes in SAPI, which are combined by the PAPI and filtered to provide JSON output containing 13 attributes.
B. Request client's AWS project team to replicate all the attributes and create Account and Case JSON table in DynamoDB. Then create separate entities for Account and Case Objects by mimicking all the attributes in SAPI

to
transfer ISON data to DynamoD for respective Objects
C. Start implementing an Enterprise Data Model by defining enterprise Account and Case Objects and implement SAPI and DynamoDB tables based on the Enterprise Data Model,
D. Create separate entities for Account with five attributes and Case with eight attributes in SAPI, which are combined by the PAPI to provide JSON output containing 13 attributes.

Answer: D

Explanation:
Understanding the Requirements:
The business needs to transfer daily data changes from the Salesforce Account and Case objects to AWS DynamoDB in a single JSON format.
Only a subset of attributes (5 from Account and 8 from Case) is required, so it is not necessary to include all 219 attributes of the Account object.
Design Approach:
A System API (SAPI) should be created for each Salesforce object (Account and Case), exposing only the required fields (5 attributes for Account and 8 for Case).
A Process API (PAPI) can be used to aggregate and transform the data from these SAPIs, combining the 13 selected attributes from Account and Case into a single JSON structure for DynamoDB.
Evaluating the Options:
Option A: Mimicking all attributes in the SAPI is inefficient and unnecessary, as only 13 attributes are required.
Option B: Replicating all attributes in DynamoDB is excessive and would result in higher storage and processing costs, which is unnecessary given the requirement for only a subset of attributes.
Option C: Implementing an Enterprise Data Model could be useful in broader data management but is not required here, as the focus is on a lightweight integration.
Option D (Correct Answer): Creating separate entities in SAPI for Account and Case with only the required attributes and using the PAPI to aggregate them into a single JSON is the most efficient and meets the requirements effectively.
Conclusion:
Option D is the best choice as it provides a lightweight, efficient design that meets the requirements by transferring only the necessary attributes and minimizing resource use.
Refer to MuleSoft's best practices for API-led connectivity and data modeling to structure SAPIs and PAPIs efficiently.

Question: 114

An API implementation is deployed to CloudHub.
What conditions can be alerted on using the default Anypoint Platform functionality, where the alert conditions depend on the API invocations to an API implementation?

A. When the API invocations are sent directly to the internal DNS record of the API implementation
B. When the API invocations are not over-a- secure TLS/SSL communication channel
C. When the APL invecations originate from a geography different than the API
D. When the number of API invocations are below a threshold

Answer: D

Explanation:

Default Alert Capabilities in Anypoint Platform:

Anypoint Platform provides out-of-the-box alerting capabilities for monitoring API invocation conditions, including setting thresholds for the number of invocations.

Alerts can be configured for conditions such as high or low traffic (invocations exceeding or falling below a defined threshold).

Evaluating the Options:

Option A: Anypoint Platform does not provide direct alerting based on DNS records.

Option B: Anypoint Platform does not provide default alerts based on whether invocations use TLS/SSL; this would require custom configuration.

Option C: Geolocation-based alerting is not natively supported in Anypoint Platform.

Option D (Correct Answer): Alerts based on API invocation thresholds (e.g., invocations falling below a set threshold) are supported and can be configured as part of the default Anypoint alerting functionality.

Conclusion:

Option D is correct, as Anypoint Platform allows configuring alerts based on the number of API invocations falling below or exceeding a threshold.

Refer to MuleSoft's documentation on Anypoint Monitoring and alert configurations for more details.

Question: 115

An organization requires several APIs to be secured with OAuth 2.0, and PingFederate has been identified as the identity provider for API client authorization, The PingFederate Client Provider is configured in access management, and the PingFederate OAuth 2.0 Token Enforcement policy is configured for the API instances required by the organization. The API instances reside in two business groups (Group A and Group B) within the Master Organization (Master Org).

What should be done to allow API consumers to access the API instances?

A. The API administrator should configure the correct client discovery URL in both child business groups, and the API consumer should request access to the API in Ping Identity

B. The API administrator should grant access to the API consumers by creating contracts in the relevant API instances in API Manager

C. The APL consumer should create a client application and request access to the APT in Anypoint Exchange, and the API administrator should approve the request

D. The APT consumer should create a client application and request access to the API in Ping Identity, and the organization's Ping Identity workflow will grant access

Answer: C

Explanation:

OAuth 2.0 and PingFederate Setup:

The organization uses PingFederate as the identity provider, integrated with Anypoint Platform for OAuth 2.0 authentication and authorization.

The PingFederate OAuth 2.0 Token Enforcement policy is applied to the API instances, requiring clients to be registered and authenticated via PingFederate.

Accessing Secured APIs:

API consumers need to register their client applications in Anypoint Exchange to request access to the secured APIs.
The API administrator then reviews and approves the access request in API Manager. This grants the client application a contract, allowing it to access the API using OAuth 2.0 tokens issued by PingFederate.
Evaluating the Options:
Option A: Configuring the client discovery URL in child business groups is not relevant to granting access; this is part of setting up PingFederate, not managing consumer access.
Option B: While creating contracts in API Manager is necessary, this option lacks the detail about the process in Anypoint Exchange, where consumers request access.
Option C (Correct Answer): API consumers must create a client application in Anypoint Exchange to request access to the API, and the API administrator then approves the request in API Manager.
Option D: The access request and approval process happens within Anypoint Platform (Exchange and API Manager), not directly in Ping Identity.
Conclusion:
Option C is the correct answer as it accurately describes the process within Anypoint Platform where API consumers request access through Exchange, and the API administrator approves it.
Refer to MuleSoft's documentation on OAuth 2.0 setup with PingFederate and managing API client access in Exchange and API Manager.

Question: 116

An organization has built an application network following the API-led connectivity approach recommended by MuleSoft. To protect the application network against
attacks from malicious external API clients, the organization plans to apply JSON Threat Protection policies.
To which API-led connectivity layer should the JSON Threat Protection policies most commonly be applied?

A. All layers
B. System layer
C. Process layer
D. Experience layer

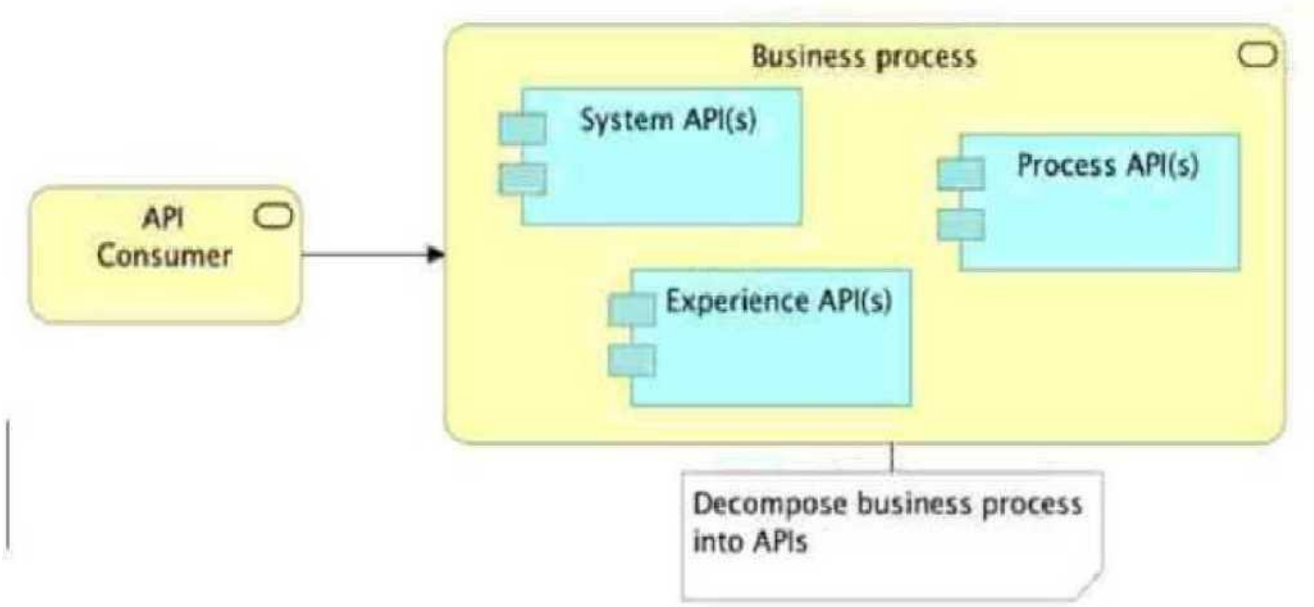
Answer: D

Explanation:
Understanding JSON Threat Protection Policies:
JSON Threat Protection policies are used to protect APIs from attacks that exploit JSON payloads, such as oversized payloads, deeply nested objects, and excessive array elements. This helps prevent Denial of Service (DoS) attacks and other malicious payload-related threats.
These policies are typically applied to safeguard APIs that are directly exposed to external clients, where the risk of receiving malicious payloads is highest.
API-led Connectivity Layers:
Experience Layer: This layer is designed to expose APIs to end-users or external API clients, often acting as the interface that interacts with users or applications.
Process Layer: This layer is used for orchestration and aggregation of data from various System APIs, typically operating within a trusted environment and not directly exposed to external clients.
System Layer: This layer provides access to backend systems and databases, often within the organization's secure environment and not directly accessible to external clients.

Evaluating the Options:
Option A (All layers): While JSON Threat Protection can technically be applied to all layers, it is most commonly applied at the Experience layer, where APIs are exposed to external traffic and are more vulnerable to attacks.
Option B (System layer): The System layer is generally not exposed to external clients directly, so JSON Threat Protection is less critical here.
Option C (Process layer): Similar to the System layer, the Process layer is typically internal and not exposed directly to external clients, so JSON Threat Protection is less commonly applied.
Option D (Correct Answer): The Experience layer is the correct answer because it is the layer that directly interacts with external clients, making it the primary target for malicious payloads. Applying JSON Threat Protection here effectively protects the application network from external threats. Conclusion:
Option D is the correct answer, as the Experience layer is the most common layer for applying JSON Threat Protection policies to protect against external attacks.
For further reference, consult MuleSoft's documentation on API security policies and best practices for securing APIs at the Experience layer.

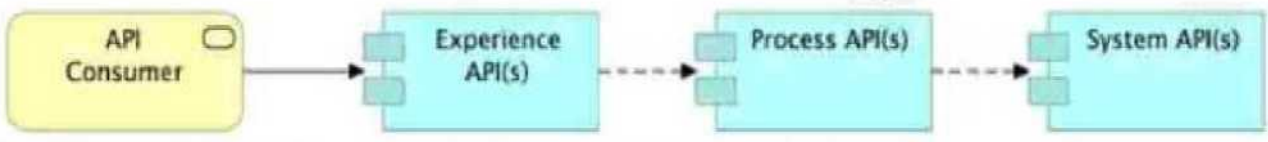
Question: 117

Refer to the exhibits.

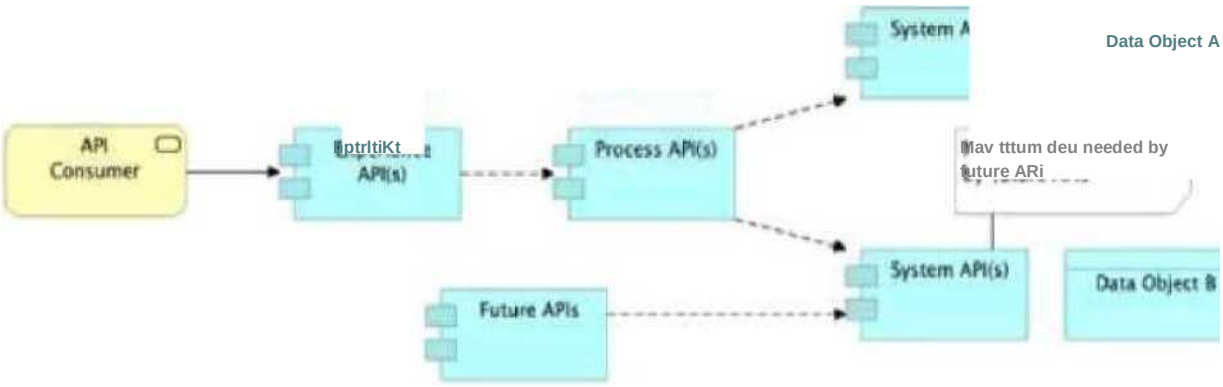


Which architectural constraint is compatible with the API-led connectivity architectural style?

A. Always use a tiered approach by creating exactly one API for each of the three layers (Experience, Process, and System)



B. Use a Process API to-orchestrate calls to multiple System APIs but not to other Process APIs:
C. Allow System APIs to return data that is not currently required by the identified Process or Experience APIs



D. Handle customizations for the end-user application at the Process layer rather than at the Experience layer

Answer: B

Explanation:

Understanding API-led Connectivity Layers:

In MuleSoft's API-led connectivity approach, APIs are categorized into three layers:

Experience Layer: This layer is responsible for providing data to the end-user applications and is often customized to meet the needs of different user interfaces.

Process Layer: This layer is used to orchestrate and combine data from multiple System APIs. It acts as a mediator and business logic layer without directly interacting with the backend systems.

System Layer: This layer provides direct access to the backend systems (e.g., databases, ERPs) and is usually focused on exposing atomic data operations.

Evaluating the Architectural Constraints:

Option A: Always using a strict tiered approach by creating exactly one API per layer is not necessarily an architectural constraint of API-led connectivity. While a layered approach is recommended, it is common to have multiple APIs in each layer as needed for different functionalities.

Option B (Correct Answer): In API-led connectivity, Process APIs are generally responsible for orchestrating calls to System APIs and should not call other Process APIs. This maintains a clear separation of concerns, ensuring that Process APIs aggregate data from System APIs only and provide it to Experience APIs.

Option C: System APIs are generally designed to provide only the necessary data to meet current business requirements. Allowing them to return extra data that is not needed by Process or Experience APIs is not a best practice, as it can lead to inefficiencies.

Option D: Customizations specific to end-user applications are typically handled at the Experience Layer rather than the Process Layer, as the Experience Layer is intended to tailor the data to fit the needs of each specific client or front-end application.

Conclusion:

Option B is the correct answer as it aligns with the API-led connectivity principles. In this architectural style, Process APIs should orchestrate System APIs but should avoid interacting with other Process APIs to keep a clear separation of responsibilities across the layers.

For additional details, refer to MuleSoft documentation on API-led connectivity best practices, particularly around the roles of each layer in API orchestration and data handling.

Question: 118

A large company wants to implement IT infrastructure in its own data center, based on the corporate IT policy requirements that data and metadata reside locally.

Which combination of Mule control plane and Mule runtime plane(s) meets the requirements?

A. Anypoint Platform Private Cloud Edition for the control plane and the MuleSoft-hosted runtime plane
B. The MuleSoft-hosted control plane and Anypoint Runtime Fabric for the runtime plane
C. The MuleSoft-hosted control plane and customer-hosted Mule runtimes for the runtime plane
D. Anypoint Platform Private Cloud Edition for the control plane and customer-hosted Mule runtimes for the runtime plane

Answer: D

Explanation:
Understanding Control and Runtime Planes:
Control Plane: The control plane is responsible for managing, monitoring, and deploying Mule applications. In a Private Cloud Edition (PCE), this control plane is deployed on-premises within the customer's infrastructure, meeting data residency and security requirements.
Runtime Plane: The runtime plane consists of Mule runtimes that execute Mule applications. By hosting these runtimes within the customer's infrastructure, data and metadata can remain local, which complies with corporate policies regarding data residency.
Evaluating the Options:
Option A: Using Anypoint Platform Private Cloud Edition for the control plane and the MuleSoft- hosted runtime plane would not meet the requirement, as the runtime plane is hosted by MuleSoft and would not keep data local.
Option B: The MuleSoft-hosted control plane with Anypoint Runtime Fabric for the runtime plane would still mean that metadata is managed in MuleSoft's cloud, which does not comply with the requirement to keep data and metadata on-premises.
Option C: A MuleSoft-hosted control plane and customer-hosted Mule runtimes also mean that metadata resides in the cloud, not on-premises, failing the residency requirement.
Option D (Correct Answer): Anypoint Platform Private Cloud Edition (PCE) for the control plane and customer-hosted Mule runtimes fulfill both requirements, as both the control plane and runtime plane would be hosted within the customer's data center.
Conclusion:
Option D is the correct answer, as it ensures that both the control plane and runtime plane are hosted on-premises, allowing data and metadata to reside locally per the corporate IT policy. Refer to MuleSoft's documentation on Private Cloud Edition deployment and on-premise runtime configurations for further details.

Question: 119

A company is building an application network using MuleSoft's recommendations for various API layers.
What is the main (default) role of a process API in an application network?
A. To secure and optimize the data synchronization processing of large data dumps between backend systems
B. To manage and process the secure direct communication between a back-end system and an enduser client of mobile device in the application network
C. To automate parts of business processes by coordinating and orchestrating the invocation of other APIs in the application network
D. To secure, Manage, and process communication with specific types of end-user client applications or devices in the application network

Answer: C

Explanation:
Role of Process API in API-led Connectivity:
In MuleSoft's API-led connectivity approach, a Process API is used to coordinate, aggregate, and orchestrate data from various System APIs. It is primarily responsible for implementing business logic that spans multiple backend systems or entities, transforming and combining data as needed to support business processes.
Process APIs are not directly exposed to end-user clients; rather, they work between System APIs and Experience APIs, providing business logic and orchestration capabilities.
Evaluating the Options:
Option A: Process APIs are not typically responsible for large data dumps or data synchronization.
That function would be handled by a System API or a specialized batch process.
Option B: Managing direct, secure communication between back-end systems and end-user clients is typically the role of Experience APIs rather than Process APIs.
Option C (Correct Answer): Process APIs are designed to coordinate and orchestrate calls to multiple other APIs in the network, which supports the automation of business processes.
Option D: Securing and managing communication with end-user clients is typically the role of Experience APIs, not Process APIs.
Conclusion:
Option C is the correct answer, as the main role of a Process API is to coordinate and orchestrate interactions between other APIs, enabling business processes to function seamlessly across multiple systems.
Refer to MuleSoft's API-led connectivity documentation for further explanation of the roles and responsibilities of Process APIs in an application network.

Question: 120

A company is using an on-prem cluster in the data center as a runtime plane and MuleSoft-hosted control plane.
How can the company monitor the detailed performance metrics on the Mule applications deployed to the cluster from the control plane?

A. The settings of the Monitoring section in the control plane must be updated to enable detailed logging on the metrics to be captured
B. Monitoring Agent must be installed on each node in the cluster
C. Due to the potential performance impact on the runtime nodes, the Monitoring agent should be installed on a separate server
D. There is no action needed as the on-prem runtime automatically sends the performance data to the control plane

Answer: B

Explanation:
Monitoring On-Premise Mule Applications:
For Mule applications deployed on an on-premises cluster, monitoring detailed performance metrics requires communication with the MuleSoft-hosted control plane. The control plane, when used with on-premises runtimes, relies on Anypoint Monitoring and requires a Monitoring Agent to gather and send detailed performance metrics.

Setting Up Monitoring:
To enable detailed metrics, the Monitoring Agent must be installed on each node in the cluster where Mule applications are deployed. This agent collects data on memory usage, CPU load, response times, and other metrics, and sends it to the control plane for aggregation and visualization.
Evaluating the Options:
Option A: Updating settings in the control plane alone does not enable detailed monitoring; the agent must be installed on each node to capture detailed metrics.
Option B (Correct Answer): Installing the Monitoring Agent on each node ensures that each runtime node in the cluster can send its metrics to the control plane, enabling detailed monitoring.
Option C: Installing the agent on a separate server would not be effective, as each node in the cluster needs to independently report its metrics to ensure full visibility.
Option D: The on-prem runtime does not automatically send detailed metrics to the control plane without the Monitoring Agent installed.
Conclusion:
Option B is the correct answer, as installing the Monitoring Agent on each node is essential for detailed performance monitoring of on-prem applications in a cluster.
Refer to MuleSoft's documentation on configuring Anypoint Monitoring for on-premises deployments and using the Monitoring Agent.

Question: 121

4A developer for a transportation organization is implementing exactly one processing functionality in a Reservation Mule application to process and store passenger
records. This Reservation application will be deployed to multiple CloudHub workers/replicas. It is possible that several external systems could send duplicate passenger records to the Reservation application.
An appropriate storage mechanism must be selected to help the Reservation application process each passenger record exactly once as much as possible. The selected storage mechanism must be shared by all the CloudHub workers/replicas in order to synchronize the state information to assist attempting exactly once processing of each passenger record by the deployed Reservation Mule application.
Which type of simple storage mechanism in Anypoint Platform allows the Reservation Mule application to update and share data between the CloudHub workers/replicas exactly once, with minimal development effort?

A. Persistent Object Store
B. Runtime Fabric Object Store
C. Non-persistent Object Store
D. In-memory Mule Object Store

Answer: A

Explanation:
Processing Requirements and Storage Mechanism:
The Reservation Mule application will be deployed to multiple CloudHub workers/replicas, meaning that each worker must share state information to handle records exactly once. This requires a shared storage mechanism where state can be stored and accessed by multiple instances to avoid duplicate processing of the same records.
A Persistent Object Store in Anypoint Platform can be used to store records in a way that is accessible across multiple workers, providing a reliable mechanism for "exactly once" processing.
Evaluating the Options:

Option A (Correct Answer): A Persistent Object Store is designed to retain data across different application instances and can be shared by all workers on CloudHub. It helps achieve idempotency by ensuring that a record is processed exactly once.

Option B: Runtime Fabric Object Store is used for applications deployed in Anypoint Runtime Fabric, not CloudHub. This option would not be compatible with the CloudHub deployment.

Option C: A Non-persistent Object Store does not retain data across application restarts or different instances, making it unsuitable for the requirement of synchronized storage for exactly-once processing.

Option D: An In-memory Mule Object Store is local to each worker and is not shared across instances, so it does not meet the requirement for a shared storage mechanism accessible to all CloudHub workers.

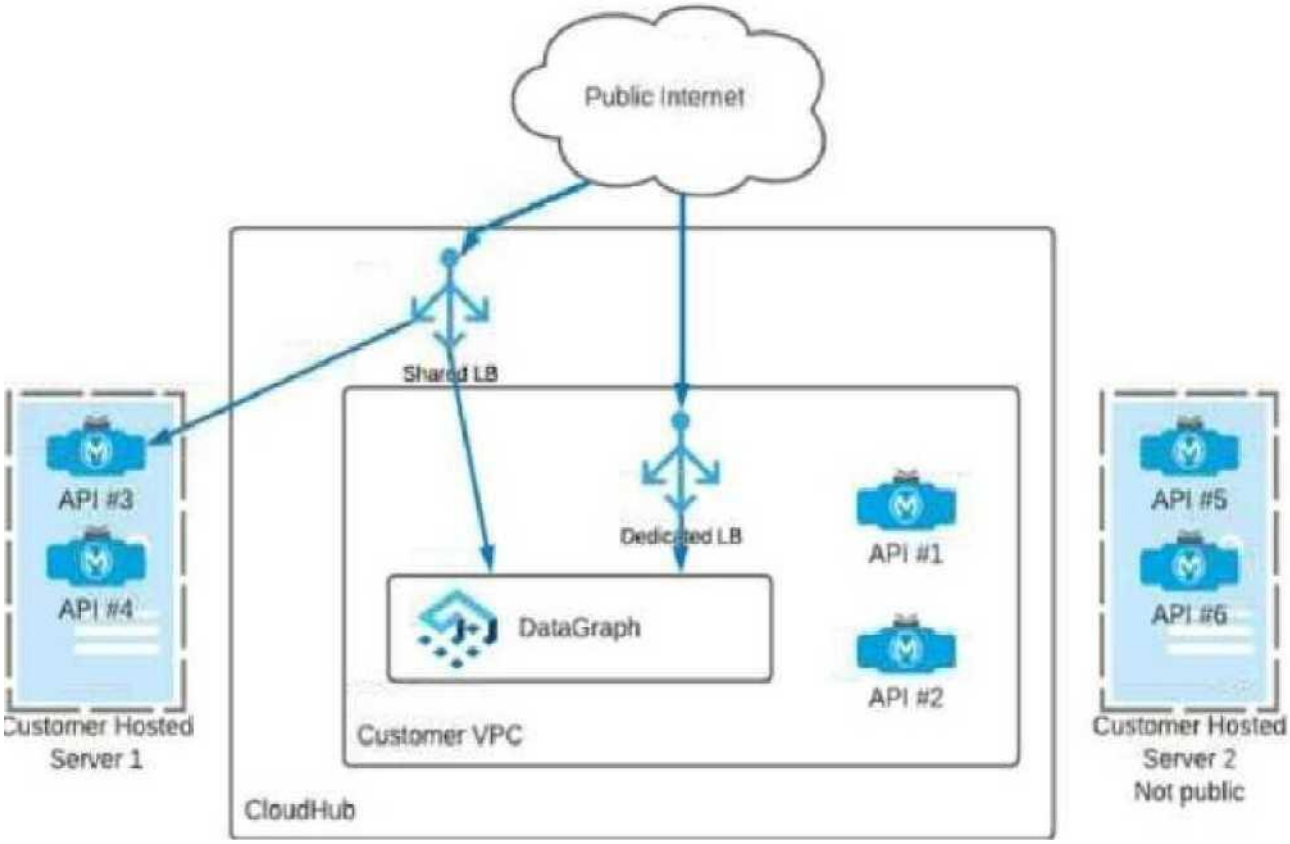
Conclusion:

Option A is the correct answer, as a Persistent Object Store allows data sharing across multiple CloudHub workers, enabling them to synchronize and achieve "exactly once" processing of passenger records with minimal development effort.

Refer to MuleSoft's documentation on Object Store configurations and usage for best practices on handling state across distributed instances.

Question: 122

Which APIs can be used with DataGraph to create a unified schema?



A. APIs 1, 3, 5
B. APIs 2, 4 ,6
C. APIs 1, 2, s5, 6
D. APIs 1, 2, 3, 4

Answer: D

Explanation:
To create a unified schema in MuleSoft's DataGraph, APIs must be exposed in a way that allows DataGraph to pull and consolidate data from these APIs into a single schema accessible to consumers. DataGraph provides a federated approach, combining multiple APIs to form a single, unified API endpoint.
In this setup:
APIs 1, 2, 3, and 4 are suitable candidates for DataGraph because they are hosted within the Customer VPC on CloudHub and are accessible either through a Shared Load Balancer (LB) or a Dedicated Load Balancer (DLB). Both of these load balancers provide public access, which is a necessary condition for DataGraph as it must access the APIs to aggregate data.
APIs 5 and 6 are hosted on Customer Hosted Server 2, which is explicitly marked as "Not public". Since DataGraph requires API access through a publicly reachable endpoint to aggregate them into a unified schema, APIs 5 and 6 cannot be used with DataGraph in this configuration.
APIs 3 and 4 on Customer Hosted Server 1 appear accessible through a Shared LB, implying public accessibility that meets DataGraph's requirements.
By combining APIs 1, 2, 3, and 4 within DataGraph, you can create a unified schema that enables clients to query data seamlessly from all these APIs as if it were from a single source.
This setup allows for efficient data retrieval and can simplify API consumption by reducing the need to call multiple APIs individually, thus optimizing performance and developer experience. Reference
For more detailed information on setting up and managing unified schemas in DataGraph, refer to the DataWeave documentation and MuleSoft DataGraph resources which provide in-depth guidelines on schema aggregation and API federation.

Question: 123

A customer wants to host their MuleSoft applications in CloudHub 1.0, and these applications should be available at the domain https://api.acmecorp.com.
After creating a dedicated load balancer (DLB) called acme-dib-prod, which further action must the customer take to complete the configuration?

A. Configure the DLB with a TLS certificate for api.acmecorp.com and create an A record for api.acmecorp.com to the public IP addresses associated with their DLB
B. Configure the DLB with a TLS certificate for api.acmecorp.com and create a CNAME record from api.acmecorp.com to acme-dib-prod.|lb.anypointdns.net
C. Configure the DLB with a TLS certificate for acme-dib-prod.Jb.anypointdns.net and create a CNAME record from api.acmecorp:com to acme-dlb-prod.lb.anypointdns.net
D. Configure the DLB with a TLS certificate for aplacmecorp.com and create a CNAME record from api.aomecorp.com to acme-dib-prod.ei.cloubhub.io

Answer: B

Explanation:
When setting up a custom domain for MuleSoft applications hosted on CloudHub 1.0 using a Dedicated Load Balancer (DLB), follow these steps:
Set Up the TLS Certificate: Configure the DLB (acme-dib-prod) with a TLS certificate that covers the custom domain api.acmecorp.com. This certificate will allow HTTPS traffic to be securely directed through the DLB to your

Mule applications.
DNS Configuration with CNAME:
Create a CNAME record that points api.acmecorp.com to the DLB hostname acme-dib-prod.lb.anypointdns.net.
The CNAME record enables the custom domain to resolve to the DLB provided by MuleSoft's Anypoint Platform.
This CNAME mapping directs all traffic to the correct DLB for processing and load distribution.
Why Option B is Correct:
A CNAME record provides the necessary aliasing to acme-dib-prod.lb.anypointdns.net, which is the endpoint managed by Anypoint Platform for your DLB.
Option B also correctly identifies the need to configure the DLB with a TLS certificate specifically for api.acmecorp.com rather than for the DLB's internal hostname.
of Incorrect Options:
Options that suggest configuring the DLB with a TLS certificate for the DLB's internal hostname or using an A record are not suitable in this scenario. MuleSoft CloudHub 1.0 DLBs work with CNAME records to provide flexible and scalable domain management, and a direct IP (A record) is not supported for these load balancers.
Reference
For more information on configuring custom domains and DLBs on CloudHub 1.0, refer to the MuleSoft documentation on DLB setup and DNS configuration.

Question: 124

A large organization with an experienced central IT department is getting started using MuleSoft.
There is a project to connect a siloed back-end system to a new Customer Relationship Management (CRM) system. The Center for Enablement is coaching them to use API-led connectivity.
What action would support the creation of an application network using API-led connectivity?

A. Invite the business analyst to create a business process model to specify the canonical data model between the two systems
B. Determine if the new CRM system supports the creation of custom: REST APIs, establishes 4 private network with CloudHub, and supports GAuth 2.0 authentication
C. To expedite this project, central IT should extend the CRM system and back-end systems to connect to one another using built in integration interfaces
D. Create a System API to unlock the data on the back-end system using a REST API

Answer: D

Explanation:
For an organization starting with API-led connectivity to integrate a siloed back-end system with a new CRM, the following approach aligns with best practices and MuleSoft's Center for Enablement (C4E) guidance:
API-led Connectivity: This model organizes APIs into distinct layers (System, Process, and Experience) to improve reusability, modularity, and manageability.
System APIs are used to expose and unlock data from core systems (such as back-end applications or databases).
Process APIs orchestrate data across multiple systems and transform it as needed.
Experience APIs format the data specifically for consumption by applications or devices, such as the CRM in this case.
Step to Support Application Network:

Create a System API for the back-end system. This API should expose the necessary data to support integration with the CRM.
By creating a System API with a RESTful interface, data can be accessed in a standardized way, making it easier to integrate with other systems and supporting future scalability.
Why Option D is Correct:
Creating a System API aligns with the principle of API-led connectivity, ensuring that data is exposed in a reusable manner. This API can then be orchestrated by Process APIs as needed to meet CRM requirements and can easily be extended to other applications.
of Incorrect Options:
Option A (creating a business process model) does not directly enable connectivity or expose backend data through APIs.
Option B is unnecessary at this stage; assessing CRM capabilities like OAuth 2.0 support is not directly related to creating the application network via System APIs.
Option C contradicts API-led best practices by suggesting a point-to-point integration, which API-led connectivity seeks to avoid due to its lack of flexibility and scalability.
Reference
Refer to MuleSoft's API-led Connectivity resources for a detailed framework on building scalable integration layers using System, Process, and Experience APIs.

Question: 125

A Rate Limiting policy is applied to an API implementation to protect the back-end system. Recently, there have been surges in demand that cause some API client
POST requests to the API implementation to be rejected with policy-related errors, causing delays and complications to the API clients.
How should the API policies that are applied to the API implementation be changed to reduce the frequency of errors returned to API clients, while still protecting the back-end system?

A. Keep the Rate Limiting policy and add 9 Client ID Enforcement policy
B. Remove the Rate Limiting policy and add an HTTP Caching policy
C. Remove the Rate Limiting policy and add a Spike Control policy
D. Keep the Rate Limiting policy and add an SLA-based Spike Control policy

Answer: D

Explanation:
When managing high traffic to an API, especially with POST requests, it is crucial to ensure the API's policies both protect the back-end systems and provide a smooth client experience. Here's the approach to reducing errors:
Rate Limiting Policy: This policy enforces a limit on the number of requests within a defined time period. However, rate limiting alone may cause clients to hit limits during demand surges, leading to errors.
Adding an SLA-based Spike Control Policy:
Spike Control is designed to handle sudden increases in traffic by smoothing out bursts of requests, which is particularly useful during high-demand periods.
By configuring SLA-based Spike Control, you can define thresholds for specific client tiers. For instance, premium clients might have higher limits or more flexibility in traffic bursts than standard clients.
Why Option D is Correct:

Keeping the Rate Limiting policy continues to provide baseline protection for the back-end.
Adding the SLA-based Spike Control policy allows for differentiated control, where requests are queued or delayed during bursts rather than outright rejected. This approach significantly reduces error responses to clients while still controlling overall traffic.
of Incorrect Options:
Option A (adding Client ID Enforcement) would not reduce errors related to traffic surges.
Option B (HTTP Caching) is not applicable as caching is generally ineffective for non-idempotent requests like POST.
Option C (only Spike Control without Rate Limiting) may leave the back-end system vulnerable to sustained high traffic levels, reducing protection.
Reference
For more information on configuring Rate Limiting and SLA-based Spike Control policies, refer to MuleSoft documentation on API Policies and Rate Limiting.

Question: 126

A Platinum customer uses the U.S. control plane and deploys applications to CloudHub in Singapore with a default log configuration.
The compliance officer asks where the logs and monitoring data reside?

A. Logs are held in:Singapore and monitoring data is held in the United States
B. Logs and monitoring data are held in the United States
C. Logs are held in the United States and monitoring data is held in Singapore
D. Logs and monitoring data are held in Singapore

Answer: B

Explanation:
For applications deployed on CloudHub in a foreign region (e.g., Singapore), MuleSoft handles log and monitoring data in the region where the control plane resides. This data storage policy is standard for CloudHub deployments to maintain centralized log and monitoring data.
Data Location:
For a U.S.-based control plane, all logs and monitoring data are stored in the United States, regardless of the deployment region.
Although the application itself runs in Singapore, data related to application performance and logs is not localized to the deployment region.
of Correct Answer (B):
Since the control plane is based in the United States, all operational data like logs and monitoring will also be stored there, ensuring compliance with MuleSoft's data handling policies.
of Incorrect Options:
Option A and D are incorrect because MuleSoft does not store logs or monitoring data in the application deployment region when the control plane is located in the United States.
Option C suggests mixed storage, which does not align with MuleSoft's data policy structure. Reference
For details on data residency in CloudHub deployments, refer to MuleSoft's documentation on CloudHub control planes and data handling policies.

Question: 127

A European company has customers all across Europe, and the IT department is migrating from an older platform to MuleSoft. The main requirements are that the
new platform should allow redeployments with zero downtime and deployment of applications to multiple runtime versions, provide security and speed, and utilize Anypoint MQ as the message service.
Which runtime plane should the company select based on the requirements without additional network configuration?

A. Runtime Fabric on VMs / Bare Metal for the runtime plane
B. Customer-hosted runtime plane
C. MuleSoft-hosted runtime plane (CloudHub)
D. Anypoint Runtime Fabric on Self-Managed Kubernetes for the runtime plane

Answer: C

Explanation:
For a European company with requirements such as zero-downtime redeployment, deployment to multiple runtime versions, secure and fast performance, and the use of Anypoint MQ without additional network configuration, CloudHub is the best choice for the following reasons: Zero-Downtime Redeployment: CloudHub supports zero-downtime deployment, which allows seamless redeployment of applications without impacting availability.
Support for Multiple Runtime Versions: CloudHub allows deploying applications across different Mule runtime versions, giving flexibility to test and migrate applications as needed.
Integrated Anypoint MQ: Anypoint MQ, which is fully integrated with CloudHub, provides reliable messaging across applications. Choosing CloudHub removes the need for additional network configurations, as Anypoint MQ can be directly accessed in this hosted environment.
Security and Performance: CloudHub offers secure networking, automatic scaling, and optimized performance without requiring a complex setup. This is managed by MuleSoft's infrastructure, meeting the speed and security requirements with minimal overhead.
of Incorrect Options:
Option A and D (Runtime Fabric on VMs or Self-Managed Kubernetes): While Runtime Fabric offers flexibility, it requires more complex network and infrastructure configurations, which is not ideal if the company seeks simplicity.
Option B (Customer-hosted): This would require additional network and security configuration, which does not align with the requirement of minimizing setup complexity.
Reference
For more information on CloudHub's capabilities regarding zero-downtime deployments and integration with Anypoint MQ, refer to MuleSoft documentation on CloudHub.

Question: 128

An operations team is analyzing the effort needed to set up monitoring of their application network. They are looking at which API invocation metrics can be used to identify and predict trouble without having to write custom scripts or install additional analytics software or tools.
Which type of metrics can satisfy this goal of directly identifying and predicting failures?

A. The number and types of API policy violations per day

B. The effectiveness of the application network based on the level of reuse
C. The number and types of past API invocations across the application network
D. The ROI from each APT invocation

Answer: A

Explanation:
To monitor an application network and predict issues without custom scripts, policy violation metrics are critical. They provide insights into potential problems by tracking instances where API usage does not conform to defined policies. Here's why this approach is suitable:
Predictive Monitoring:
Tracking API policy violations (such as rate limits or spike controls being hit) can indicate surges in traffic or misuse, which may lead to throttling or service degradation if not addressed.
By monitoring these violations, teams can proactively adjust limits or optimize API handling to prevent actual failures.
No Custom Scripting Needed:
Policy violation metrics are available within MuleSoft's Anypoint Monitoring, meaning there's no need to implement custom solutions or external tools to gather and interpret this data.
of Incorrect Options:
Option B (effectiveness based on reuse) does not directly predict failures.
Option C (past invocation counts) offers historical usage data but does not inherently identify issues.
Option D (ROI from API invocation) is a business metric and does not provide technical insights for failure prediction.
Reference
For more details on leveraging policy violation metrics for proactive monitoring, refer to MuleSoft documentation on Anypoint Monitoring.

Question: 129

A developer from the Central IT team has created an initial version of the RAML definition in Design Center for an OAuth 2.0-protected System API and published it to Exchange. Another developer from LoB IT discovered the System API in Exchange and would like to leverage it in the Process API.
What is the MuleSoft-recommended approach for Process API to invoke the System API?

A. The Process API needs to import an CAuth 2.0 module from Exchange first and update it with OAuth 2.0 credentials before the System API can be invoked
B. The Process API uses property YAML files to store the System API URLs and uses the HTTP Request Connector to invoke the Systerm API
C. The Process APL uses the REST Connect Connector autogenerated in Exchange for the System API
D. The Process API manually updates the Process API POM file to include the System API as a dependency

Answer: C

Explanation:
In MuleSoft's ecosystem, when a Process API needs to consume a System API (published to Exchange and protected by OAuth 2.0), the recommended approach is to utilize the REST Connect Connector. Here's how it aligns with best practices:

Automated Connector Generation:
When a RAML or OAS specification is published in Exchange, MuleSoft automatically generates a REST Connect Connector for that API. This connector simplifies integration as it abstracts the complexity of making HTTP requests and handling OAuth authentication.
Streamlined Integration:
The Process API can import this generated connector from Exchange and configure OAuth credentials, streamlining secure access to the System API without manual HTTP setup.
Why Option C is Correct:
Using the REST Connect Connector directly leverages MuleSoft's automated tooling, minimizing manual configurations and ensuring a more maintainable integration.
of Incorrect Options:
Option A (importing an OAuth module) is unnecessary; OAuth is handled within the connector's configuration.
Option B (property YAML files with HTTP requests) involves manual setup, which is more error-prone and not recommended.
Option D (manually updating POM file) does not directly aid in invoking an API through Exchange. Reference
For more information on using REST Connect Connectors and OAuth integration in MuleSoft, refer to the MuleSoft documentation on API Management and Connectors.

Question: 130

When can CloudHub Object Store v2 be used?

A. To store an unlimited number of key-value pairs
B. To store payloads with an average size greater than 15MB
C. To store information in Mule 4 Object Store v1
D. To store key-value pairs with keys up to 300 characters

Answer: D

Explanation:
CloudHub Object Store v2 is a managed key-value store provided by MuleSoft to support various use cases where temporary data storage is required. Here's why Option D is correct:
Key Length Support: Object Store v2 allows storage of keys with a length of up to 300 characters, making it suitable for applications needing flexible and descriptive keys.
Limitations on Size:
Object Store v2 is not intended for large payload storage and has a recommended size limit below 10 MB for each value. Payloads exceeding 15 MB may cause performance issues and are better suited to a file storage system or database.
Option B is incorrect because storing payloads above 15 MB exceeds Object Store's optimal usage specifications.
Key-Value Limits: Object Store v2 is designed for moderate, transient storage needs, and does not support unlimited storage. Thus, Option A is incorrect.
Backward Compatibility: Object Store v2 does not support Mule 4 applications running Object Store v1. Option C is incorrect as Object Store v1 and v2 are distinct.
Reference
For more on CloudHub Object Store v2, refer to MuleSoft documentation on Object Store limitations and configuration.

Question: 131

Which scenario is suited for MUnit tests instead of integration tests?

A. For read-only interactions to any dependencies (such as other web APIs)
B. When testing does not require knowledge of implementation details
C. When no mocking is permissible
D. For tests that are implemented using SoapUI

Answer: A

Explanation:
MUnit is MuleSoft's testing framework for creating and running automated tests within Anypoint Studio. It is specifically designed for unit testing Mule applications and is best suited when testing doesn't require understanding the inner workings or implementation details of the components being tested.
Ideal Use Cases for MUnit:
MUnit is optimal when testing individual flows, functions, or components in isolation. This type of testing focuses on verifying the behavior of each unit without needing to understand the complete system.
Since unit tests do not require external integrations or dependencies to be live, mocking is commonly used in MUnit to simulate the behavior of external services and APIs.
Why Option B is Correct:
Option B aligns with the concept of unit testing, where the emphasis is on testing functionality rather than system integration. Integration tests, on the other hand, would require implementation knowledge and live endpoints, making them unsuitable for MUnit's scope.
of Incorrect Options:
Option A (read-only interactions) and Option C (no mocking) do not suit MUnit's typical testing environment as MUnit is designed with mocking capabilities to simulate dependencies.
Option D (SoapUI-based tests) suggests an external testing tool, while MUnit is specific to MuleSoft. Reference
For more on MUnit best practices, refer to MuleSoft's MUnit documentation.

Question: 132

A customer has an ELA contract with MuleSoft. An API deployed to CloudHub is consistently experiencing performance issues. Based on the root cause analysis, it is determined that autoscaling needs to be applied.
How can this be achieved?

A. Configure a policy so that when the number of HTTP requests reaches a certain threshold the number of workers/replicas increases (horizontal scaling)
B. Configure two separate policies: When CPU and memory reach certain threshold, increase the worker/replica type (vertical sealing) and the number of workers/replicas (horizontal sealing) C. Configure a policy based on CPU usage so that CloudHub auto-adjusts the number of workers/replicas (horizontal scaling)
D. Configure a policy so that when the response time reaches a certain threshold the worker/replica type increases (vertical scaling)

Answer: C

Explanation:
In MuleSoft CloudHub, autoscaling is essential to managing application load efficiently. CloudHub supports

horizontal scaling based on CPU usage, which is well-suited to applications experiencing variable demand and needing responsive resource allocation.
Autoscaling on CloudHub:
Horizontal scaling increases the number of workers in response to CPU usage thresholds, allowing the application to handle higher loads dynamically. This approach improves performance without downtime or manual intervention.
Why Option C is Correct:
Setting up autoscaling based on CPU usage aligns with MuleSoft's best practices for scalable and responsive applications on CloudHub, particularly in an environment with fluctuating load patterns. Option C correctly leverages CloudHub's autoscaling features based on resource metrics, which are part of CloudHub's managed scaling solutions.
of Incorrect Options:
Option A (based on HTTP request thresholds) and Option B (separate policies for CPU and memory) do not represent CloudHub's recommended scaling practices.
Option D suggests vertical scaling based on response time, which is not how CloudHub handles autoscaling.
Reference
For more on CloudHub's autoscaling configuration, refer to MuleSoft documentation on CloudHub autoscaling policies.

Question: 133

A company deployed an API to a single worker/replica in the shared cloud in the U.S. West Region. What happens when the Availability Zone experiences an outage?

A. CloudHub will auto-redeploy the APL in the U.S. East Region
B. The APT will be unavailable until the availability comes back online, at which time the worker/replica will be auto-restarted
C. CloudHub will auto-redeploy the API in another Availability Zone in the U.S. West Region
D. The Anypoint Platform admin is alerted when the AP] is experiencing an outage and needs the trigger the CI/CD pipeline to redeploy to the US. East Region

Answer: B

Explanation:
In a CloudHub deployment with a single worker/replica located in a specific Availability Zone (AZ), if an AZ experiences an outage, here's what happens:
Worker Availability: Since the application is deployed in a single AZ, CloudHub does not automatically redeploy the application in a different zone or region during an outage. Thus, if the current AZ is unavailable, the application will be ofline.
Auto-Restart upon AZ Recovery: Once the affected AZ is back online, CloudHub will auto-restart the worker in the same AZ without manual intervention. This ensures that as soon as the AZ is functional, the application resumes automatically.
of Correct Answer (B):
Option B accurately describes the situation, as the API will remain unavailable until the original AZ is restored.
CloudHub does not currently support automatic failover across regions or other availability zones within the same region for single-worker deployments on the shared cloud.
of Incorrect Options:

Option A (auto-redeployment in the U.S. East region) is incorrect, as CloudHub does not migrate across regions automatically.

Option C (redeployment in another AZ within the U.S. West) is not a feature for single-worker deployments.

Option D (manual redeployment triggered by an admin) is unnecessary as CloudHub handles restarts automatically when the AZ is back online.

Reference

Refer to MuleSoft CloudHub's availability and disaster recovery documentation for more information on how CloudHub manages availability in shared environments.

Question: 134

Which three tools automate the deployment of Mule applications?

Choose 3 answers

A. Runtime Manager
B. Anypoint Platform CLI
C. Platform APIs
D. Anypoint Studio
E. Mule Mayen plugin
F. API Community Manager

Answer: ABC

Explanation:

MuleSoft offers various tools to automate the deployment of Mule applications, which can streamline deployment and management processes. Here's how each tool supports automated deployment:

Runtime Manager:

Anypoint Runtime Manager is MuleSoft's web-based interface that allows users to deploy, manage, and monitor applications directly. It provides deployment automation through its user-friendly interface.

Anypoint Platform CLI:

The Anypoint CLI enables scripting of deployment and management tasks, making it possible to automate deployments via command-line scripts. This tool is ideal for CI/CD pipelines as it integrates with automated processes.

Platform APIs:

MuleSoft's Platform APIs allow programmatic access to deployment functions, enabling integration with external automation tools and CI/CD systems. These APIs facilitate deployment through RESTful calls, which can be automated for continuous delivery.

of Incorrect Options:

Option D (Anypoint Studio) is primarily for development and does not support deployment automation.

Option E (Maven Plugin) can be used for building and deploying Mule applications but isn't classified as a platform tool for deployment.

Option F (API Community Manager) is unrelated to deployment and instead focuses on managing API communities.

Reference

For detailed steps on automating deployments with these tools, refer to MuleSoft documentation on Runtime Manager, CLI, and Platform APIs.

Question: 135

An application updates an inventory running only one process at any given time to keep the inventory consistent. This process takes 200 milliseconds (.2 seconds) to execute; therefore, the scalability threshold of the application is five requests per second. What is the impact on the application if horizontal scaling is applied, thereby increasing the number of Mule workers?

A. The application scalability threshold is five requests per second regardless of the horizontal scaling B. The total process execution time is now 100 milliseconds (.1 seconds)
C. The application scalability threshold is now 10 requests per second
D. Horizontal scaling cannot be applied to an already-running application

Answer: A

Explanation:
Given that the application is designed to handle only one process at a time to maintain data consistency, here's why horizontal scaling won't increase the processing limit: Single-Process Constraint:
The application limits to processing one transaction at a time due to its design for consistency, meaning horizontal scaling (adding more workers) does not increase processing speed beyond this limit.
Execution Time:
Since each request takes 200 ms, five requests per second is the maximum processing threshold. Increasing the number of workers does not bypass this single-process limitation.
of Correct Answer (A):
The scalability remains at five requests per second, as this constraint is intrinsic to the application's design.
of Incorrect Options:
Option B suggests a change in execution time, which horizontal scaling does not affect.
Option C assumes doubling the throughput, which isn't possible due to the single-threaded nature of the application.
Option D suggests horizontal scaling cannot apply, which is incorrect; however, scaling does not increase throughput in this context.
Reference
For more on understanding scaling and concurrency in Mule applications, see MuleSoft's documentation on application performance and scaling limitations.

Question: 136

A Platform Architect inherits a legacy monolithic SOAP-based web service that performs a number of tasks, including showing all policies belonging to a client. The service connects to two back-end systems — a life-insurance administration system and a general-insurance administration system — and then queries for insurance policy information within each system, aggregates the results, and presents a SOAP-based response to a user interface (UI).
The architect wants to break up the monolithic web service to follow API-led conventions. Which part of the service should be put into the process layer?

A. Combining the insurance policy information from the administration systems
B. Presenting the SOAP-based response to the UI
C. Authenticating and maintaining connections to each of the back-end administration systems

D. Querying the data from the administration systems

Answer: A

Explanation:

In the API-led connectivity approach, each layer (System, Process, and Experience) has a distinct purpose:

System APIs: These APIs connect directly to backend systems to expose and unlock data in a standardized way.

Process APIs: These are responsible for orchestrating and processing data across different systems, combining information where needed.

Experience APIs: These are designed for specific user interfaces or applications, often transforming data formats to fit the needs of each consumer application.

Why Option A is Correct:

Process APIs are designed to combine data from multiple systems, which aligns with the function of aggregating policy information from both the life and general insurance systems. This aggregation logic would ideally reside in the Process layer, separating data retrieval from data orchestration.

Moving this functionality to the Process layer enables reusability and modularity, as other Experience APIs or services could also leverage the combined policy data if needed.

of Incorrect Options:

Option B (Presenting the SOAP-based response) would be managed by the Experience layer, as this layer adapts data formats for specific interfaces.

Option C (Authenticating and maintaining backend connections) would typically be handled within the System layer, where backend integration and security handling occurs.

Option D (Querying data) is the function of System APIs, which access the backend systems directly and expose the raw data without additional processing.

Reference

For further details on API-led architecture and the roles of each layer, refer to MuleSoft's documentation on API-led connectivity and API layers.

Question: 137

An enterprise is embarking on the API-led digital transformation journey, and the central IT team has started to define System APIs. Currently there is no Enterprise

Data Model being defined within the enterprise, and the definition of a clean Bounded Context Data Model requires too much effort.

According to MuleSoft's recommended guidelines, how should the System API data model be defined?

A. If there are misspellings of the data fields in the back-end system, Systerm APIs should not correct it, and expose it as-is to mirror the back-end systems

B. The data model of the System APIs should make use of data types that approximately mirror those from the back-end systems

C. The data model should define its own naming convention, and not follow the same naming as the back-end systems

D. The System APIs should expose all back-end system fields

Answer: B

Explanation:
When defining data models for System APIs without an established Enterprise Data Model, MuleSoft recommends mirroring the back-end systems' data types to achieve quick and effective integration without adding complexity. This approach has several benefits:
Alignment with Backend Systems:
Mirroring data types ensures consistency with backend data sources, which simplifies integration, reduces mapping requirements, and minimizes potential data transformation issues.
Flexibility for Future Enhancements:
By retaining close alignment with backend data structures, System APIs can evolve to support an Enterprise Data Model in the future without immediate restructuring.
of Incorrect Options:
Option A (exposing misspellings) is not recommended as System APIs should still ensure a professional and coherent interface.
Option C (custom naming) complicates the API structure without adding immediate value in the absence of a clear data model.
Option D (exposing all fields) is unnecessary and can reduce performance and add complexity. Reference
Refer to MuleSoft best practices for data modeling in System APIs for additional information on mirroring backend systems.

Question: 138

A team is planning to enhance an Experience API specification, and they are following API-led connectivity design principles.
What is their motivation for enhancing the API?

A. The primary API consumer wants certain kinds of endpoints changed from the Center for Enablement standard to the consumer system standard
B. The underlying System API is updated to provide more detailed data for several heavily used resources
C. An IP Allowlist policy is being added to the API instances in the Development and Staging environments
D. A Canonical Data Model is being adopted that impacts several types of data included in the API

Answer: D

Explanation:
In API-led design, an Experience API is enhanced to improve how data is delivered to end-user applications. One primary reason to enhance an Experience API is when new data standards, such as a Canonical Data Model, are adopted. Here's why:
Canonical Data Model (CDM):
Adopting a CDM standardizes data representations across the organization, making APIs more consistent and easier to consume across various services and applications.
Updating the Experience API ensures that it delivers data in this standardized format, improving interoperability and reusability.
of Correct Answer (D):
A CDM impacts the structure and types of data the API provides, and this update would be directly relevant to an Experience API, as it is the primary point of interaction for applications.

of Incorrect Options:

Option A involves adapting to consumer-specific standards, which is against API-led design principles.

Option B involves changes in System APIs, which don't directly mandate changes to the Experience API unless data formatting adjustments are required.

Option C (IP Allowlist) relates to security rather than API design and would not motivate a functional enhancement of the API.

Reference

For more details on the use of Canonical Data Models in API-led architecture, refer to MuleSoft's guidelines on data standardization and Experience API best practices.

Question: 139

An organization wants to create a Center for Enablement (C4E). The IT director schedules a series of meetings with IT senior managers.

What should be on the agenda of the first meeting?

A. Define C4E objectives, mission statement, guiding principles, a
B. Explore API monetization options based on identified use cases through MuleSoft
C. A walk through of common-services best practices for logging, auditing, exception handling, caching, security via policy, and rate limiting/throttling via policy
D. Specify operating model for the MuleSoft Integrations division

Answer: A

Explanation:

In the initial meeting for establishing a Center for Enablement (C4E), it's essential to lay the foundational vision, objectives, and guiding principles for the team. Here's why this is crucial: Clear Vision and Mission:

Defining the mission statement and objectives at the start ensures alignment within the organization and clarifies the C4E's role in supporting API-led development and integration practices.

Guiding Principles:

Establishing guiding principles will help the C4E maintain consistent practices and strategies across projects. This serves as a framework for decisions and fosters shared understanding among IT leaders and stakeholders.

of Correct Answer (A):

By prioritizing the C4E's objectives and mission, the organization builds a solid foundation, paving the way for subsequent meetings focused on technical standards, processes, and operating models. of Incorrect Options:

Option B (API monetization) and Option C (common services best practices) are specific topics better suited for later discussions.

Option D (specifying the operating model) is an important step but typically follows the establishment of the C4E's objectives and vision.

Reference

For more on C4E objectives and foundational setup, refer to MuleSoft's documentation on establishing a C4E and the roles and mission statements recommended for such initiatives.

Question: 140

Which statement is true about identity management and client management on Anypoint Platform?

A. If an external identity provider is configured, the SAML 2.0 bearer tokens issued by the identity provider cannot be used for invocations of the Anypoint Platform web APIs
B. If an external client provider is configured, it must be configured at the Anypoint Platform organization level and cannot be assigned to individual business groups and environments
C. Anypoint Platform supports configuring one external identity provider
D. Both client management and identity management require an identity provider

Answer: C

Explanation:
Anypoint Platform allows organizations to integrate one external identity provider (IdP) for identity and access management (IAM), supporting SSO and centralized user authentication.
Identity Provider Limit:
Anypoint Platform supports configuring a single IdP for the organization, which can be used to authenticate all users across business groups and environments within that Anypoint organization. of Correct Answer (C):
Configuring one IdP ensures centralized and secure identity management, aligned with MuleSoft's architecture.
of Incorrect Options:
Option A is incorrect because SAML 2.0 bearer tokens from external IdPs can indeed be used for invoking Anypoint Platform APIs.
Option B is incorrect as client providers can be assigned to specific business groups and environments.
Option D is incorrect since only identity management strictly requires an IdP; client management does not.
Reference
For further details on identity management options, consult MuleSoft documentation on Anypoint Platform's IAM capabilities.

Question: 141

Which two statements are true about the technology architecture of an Anypoint Virtual Private Cloud (VPC)?
Choose 2 answers

A. Ports 8081 and 8082 are used
B. CIDR blacks are used
C. Anypoint VPC is responsible for load balancing the applications
D. Round-robin load balancing is used to distribute client requests across different applications
E. By default, HTTP requests can be made from the public internet to workers at port 6091

Answer: BE

Explanation:
An Anypoint Virtual Private Cloud (VPC) provides a secure and private networking environment for MuleSoft applications, using specific architectural elements:
CIDR Blocks:
Anypoint VPCs utilize CIDR blocks to define IP ranges, allowing organizations to control and segment the VPC's IP address space.
Port 6091 for HTTP Requests:
By default, HTTP requests can be made to workers on port 6091 from the public internet, providing an accessible entry point unless additional restrictions are applied.

of Correct Answers (B, E):

CIDR blocks enable IP range management for VPCs, and port 6091 is the default public entry port, which is part of Anypoint VPC's default settings.

of Incorrect Options:

Option A (Ports 8081 and 8082) is incorrect; these are not default public ports for Anypoint VPC.

Option C (responsible for load balancing) is incorrect as load balancing requires a separate Dedicated Load Balancer (DLB).

Option D (round-robin load balancing) applies to DLBs, not directly to VPCs.

Reference

For more on VPC setup and networking, refer to MuleSoft documentation on VPC configurations and default port settings.

Question: 142

The Line of Business (LoB) of an eCommerce company is requesting a process that sends automated notifications via email every time a new order is processed through the customer's mobile application or through the internal company's web application. In the future, multiple notification channels may be added: for example, text messages and push notifications.

What is the most effective API-led connectivity approach for the scenario described above?

A.

Create one Experience API for the web application and one for the mobile application.

Create a Process API to orchestrate and retrieve the email template from = database.

Create a System API that sends the email using the Anypoint Connector for Email.

Create one Experience API for the web application and one for the mobile application.

Create a Process API to orchestrate and retrieve the email template from = database. Create a System API that sends the email using the Anypoint Connector for Email.

B.

Create one Experience API for the web application and one for the mobile application, Create a Process API to orchestrate, retrieve the email template from a database, and send the email using the Anypoint Connector for Email.

C.

Create Experience APIs for both the web application and mobile application.

Create a Process API ta orchestrate, retrieve the email template from e database, and send the email using the Anypoint Connector for Email.

\

D.

Create Experience APIs for both the web application and mobile application.

(Create 3 Process API to orchestrate and retrieve the email template from 2 databese.

Create a System API that sends the email using the Anypoint Connector for Email.

Answer: A

Explanation:

In this scenario, the best approach to satisfy the API-led connectivity principles and support future scalability is:

Experience APIs:

Create separate Experience APIs for the web application and the mobile application. This allows each application to have an optimized interface, supporting different needs and potential differences in request/response structures or security configurations.

Process API:

A single Process API can be used to orchestrate the workflow, including retrieving the email template from a database and preparing the email content. By centralizing this logic in the Process layer, we can ensure it is reusable and easily adaptable for different notification channels in the future.

System API:

A System API specifically designed for sending emails (using the Anypoint Connector for Email) abstracts the email-sending functionality from the business logic. This approach ensures that the email-sending function is reusable and scalable, and it can easily be extended or modified if other notification channels (like SMS or push notifications) are added later.

Why Option A is Correct:

This structure aligns with API-led connectivity principles by separating concerns across Experience, Process, and System layers. It provides flexibility for future notification channels and isolates each layer's responsibility, making it easier to maintain and scale.

of Incorrect Options:

Option B lacks a separate System API for sending emails, which goes against the principle of isolating back-end functionality in System APIs.

Option C similarly lacks a dedicated System API, reducing flexibility and reusability.

Option D suggests creating multiple Process APIs for database retrieval, which adds unnecessary complexity and does not adhere to the single-orchestration principle typically followed in API-led design.

Reference

For further guidance on API-led connectivity and the responsibilities of each API layer, refer to MuleSoft's documentation on API-led architecture and design best practices.

Question: 143

A manufacturing company has deployed an API implementation to CloudHub and has not configured it to be automatically restarted by CloudHub when the worker is not responding.

Which statement is true when no API Client invokes that API implementation?

A. No alert on the API invocations and APT implementation can be raised
B. Alerts on the APT invocation and API implementation can be raised
C. No alert on the API invocations is raised but alerts on the API implementation can be raised
D. Alerts on the API invocations are raised but no alerts on the API implementation can be raised

Answer: C

Explanation:

When an API implementation is deployed on CloudHub without configuring automatic restarts in case of worker non-responsiveness, MuleSoft's monitoring and alerting behavior is as follows: API Invocation Alerts:

If no clients are invoking the API, there will be no invocation alerts triggered, as alerts related to invocations depend on actual client requests.

Implementation-Level Alerts:

Even without invocation, CloudHub can still monitor the state of the API implementation. If the worker becomes unresponsive, an alert related to the API implementation's health or availability could still be raised.

Why Option C is Correct:

This option correctly identifies that no invocation-related alerts would be triggered in the absence of client requests, while implementation-level alerts could still be generated based on the worker's state.

Reference

For additional information, check MuleSoft documentation on CloudHub monitoring and alert configurations to understand worker status alerts versus invocation alerts.

Question: 144

An auto manufacturer has a mature CI/CD practice and wants to automate packaging and deployment of any Mule applications to various deployment targets, including CloudHub workers/replicas, customer-hosted Mule runtimes, and Anypoint Runtime Fabric.

Which MuleSoft-provided tool or component facilitates automating the packaging and deployment of Mule applications to various deployment targets as part of the company's CI/CD practice?

A. Anypoint Runtime Manager
B. Mule Maven plugin
C. Anypoint Platform CLI
D. Anypoint Platform REST APIs

Answer: B

Explanation:

For organizations with established CI/CD practices, the Mule Maven plugin is the recommended tool for automating packaging and deployment across multiple environments, including CloudHub, on-premise Mule runtimes, and Anypoint Runtime Fabric. Here's why: Automation with Maven:

The Mule Maven plugin allows for CI/CD integration by supporting automated build and deployment processes. It is commonly used in CI/CD pipelines to handle application packaging and deployment directly through Maven commands, making it ideal for teams that want consistent deployment automation across different MuleSoft environments.

Supported Deployment Targets:

The Mule Maven plugin supports deployment to various targets, including CloudHub, Runtime Fabric, and on-premises servers, thus meeting the needs of environments with diverse deployment destinations.

Why Option B is Correct:

The Mule Maven plugin is specifically designed for CI/CD pipelines and integrates with Jenkins, GitLab, and other CI/CD tools to facilitate continuous deployment. It is the most efficient MuleSoft-provided tool for this purpose.

of Incorrect Options:

Option A (Anypoint Runtime Manager) provides deployment management but does not automate CI/CD processes.

Option C (Anypoint Platform CLI) can script deployments but lacks direct integration with CI/CD tools.

Option D (Anypoint Platform REST APIs) requires custom scripting for deployment, which can be more complex than using the Mule Maven plugin.

Reference

For more details, refer to MuleSoft documentation on using the Mule Maven plugin for CI/CD.

Question: 145

An existing Quoting API is defined in RAML and used by REST clients for interacting with the quoting engine. Currently there is a resource defined in the RAML that allows the creation of quotes; however, a new requirement was just received to allow for the updating of existing quotes.

Which two actions need to be taken to facilitate this change so it can be processed?

Choose 2 answers

A. Update the API implementation to accommodate the new update request
B. Remove the old client applications and create new client applications to account for the changes
C. Update the RAML with new method details for the update request
D. Deprecate existing versions of the API in Exchange
E. Add a new API policy to API Manager to allow access to the updated endpoint

Answer: AC

Explanation:
To accommodate the new requirement of allowing updates to existing quotes, the following actions should be taken:
Update the RAML Definition (Option C):
The RAML specification defines the structure and behavior of the API. Adding a new method (such as PUT or PATCH) for updating quotes requires modifying the RAML to include this new endpoint. This ensures the API specification is up-to-date and accurately reflects the new functionality.
Update the API Implementation (Option A):
Once the RAML is updated, the backend API implementation must also be modified to handle the new update requests. This could involve adding logic to process and validate update requests, connect to necessary backend resources, and apply the changes to existing quotes.
of Incorrect Options:
Option B (removing and creating new clients) is unnecessary; client applications can remain as they are, with no need for complete replacement.
Option D (deprecating existing versions) may not be required if backward compatibility is maintained.
Option E (adding a new policy) does not facilitate functional changes and is unrelated to implementing the update feature.
Reference
For more details on updating RAML definitions and API implementations, refer to MuleSoft's API Design documentation on RAML and RESTful API practices.

Question: 146

A Mule 4 API has been deployed to CloudHub and a Basic Authentication - Simple policy has been applied to all API methods and resources. However, the API is still accessible by clients without using authentication.
How is this possible?

A. The APE Router component is pointing to the incorrect Exchange version of the APT
B. The Autodiscovery element is not present, in the deployed Mule application
C. No… for client applications have been created of this API
D. One of the application's CloudHub workers restarted

Answer: B

Explanation:
When a Basic Authentication policy is applied to an API on CloudHub but clients can still access the API without authentication, the likely cause is a missing Autodiscovery element. Here's how this affects API security:

Autodiscovery in MuleSoft:
The Autodiscovery element is essential for linking an API implementation deployed in CloudHub with its API instance defined in API Manager. This connection allows the policies applied in API Manager, such as Basic Authentication, to be enforced on the deployed API.
Why Option B is Correct:
Without Autodiscovery, the deployed application does not "know" about the policies configured in API Manager, resulting in unrestricted access. Adding Autodiscovery enables the API to enforce the policies correctly.
of Incorrect Options:
Option A (incorrect Exchange version) would not cause bypassing of security policies.
Option C (missing client applications) does not impact authentication policy enforcement.
Option D (worker restart) is irrelevant to policy enforcement.
Reference
Refer to MuleSoft documentation on Autodiscovery configuration and linking API Manager policies for additional information on setting up secure API policies.

Question: 147

A TemperatureSensors API instance is defined in API Manager in the PROD environment of the CAR_FACTORY business group. An AcmelemperatureSensors Mule
application implements this API instance and is deployed from Runtime Manager to the PROD environment of the CAR_FACTORY business group. A policy that requires a valid client ID and client secret is applied in API Manager to the API instance.
Where can an API consumer obtain a valid client ID and client secret to call the AcmeTemperatureSensors Mule application?

A. In secrets manager, request access to the Shared Secret static username/password
B. In API Manager, from the PROD environment of the CAR_FACTORY business group
C. In access management, from the PROD environment of the CAR_FACTORY business group
D. In Anypoint Exchange, from an API client application that has been approved for the TemperatureSensors API instance

Answer: D

Explanation:
When an API policy requiring a client ID and client secret is applied to an API instance in API Manager, API consumers must obtain these credentials through a registered client application.
Here's how it works:
Anypoint Exchange and Client Applications:
To access secured APIs, API consumers need to create or register a client application in Anypoint Exchange. This process involves requesting access to the specific API, and once approved, the consumer can retrieve the client ID and client secret associated with the application.
Why Option D is Correct:
Option D accurately describes the process, as the client ID and client secret are generated and managed within Anypoint Exchange. API consumers can use these credentials to authenticate with the TemperatureSensors API.
of Incorrect Options:
Option A (secrets manager) is incorrect because client credentials for API access are not managed via secrets

manager.
Option B (API Manager) is incorrect as API Manager manages policies but does not provide clientspecific credentials.
Option C (Access Management) does not apply, as Access Management is primarily used for user roles and permissions, not API client credentials.
Reference
For further details on managing client applications in Anypoint Exchange, consult MuleSoft documentation on client application registration and API security policies.

Question: 148

An established communications company is beginning its API-led connectivity journey, The company has been using a successful Enterprise Data Model for many years. The company has identified a selfservice account management app as the first effort for API-led, and it has identified the following APIs.
Experience layer: Mobile Account Management EAPI, Browser Account Management EAPI
Process layer: Customer Lookup PAPI, Service Lookup PAPI, Account Lookup PAPI
System layer: Customer SAPI, Account SAPI, Product SAPI, Service SAPI
According to MuleSoft's API-led connectivity approach, which API would not be served by the Enterprise Data Model?

A. Customer SAPI
B. Customer Lookup PAPI
C. Mobile Account Management EAPI
D. Service SAPI

Answer: C

Explanation:
In the API-led connectivity approach, APIs are categorized into Experience, Process, and System layers:
Enterprise Data Model Scope:
The Enterprise Data Model (EDM) generally supports System APIs and some Process APIs by defining standard data structures used across the organization. Experience APIs, however, are tailored to specific applications or interfaces and are less likely to be served directly by the EDM, as they may require customized data representations to meet the unique needs of each user interface.
Why Option C is Correct:
The Mobile Account Management EAPI serves mobile-specific needs and often requires data formatted differently from the standardized data models. Thus, it would be outside the direct scope of the EDM and might employ custom mappings to fit mobile application requirements.
of Incorrect Options:
Option A (Customer SAPI), Option B (Customer Lookup PAPI), and Option D (Service SAPI) would typically align with the EDM as they are closer to the core data and services the EDM supports. Reference
For additional guidance, review MuleSoft's best practices on API-led connectivity and data modeling.

Question: 149

A Mule application implements an API. The Mule application has an HTTP Listener whose connector configuration sets the HTTPS protocol and hard-codes the port

value. The Mule application is deployed to an Anypoint VPC and uses the CloudHub 1.0 Shared Load Balancer (SLB) for all incoming traffic.
Which port number must be assigned to the HTTP Listener's connector configuration so that the Mule application properly receives HTTPS API invocations routed through the SLB?

A. 8082
B. 8092
C. 80
D. 443

Answer: B

Explanation:
When using CloudHub 1.0's Shared Load Balancer (SLB) for a Mule application configured with HTTPS in an Anypoint VPC, specific ports must be configured for the application to correctly route incoming traffic:
Port Requirement for SLB:
The CloudHub Shared Load Balancer for HTTPS traffic requires that applications listen on port 8092 for secure (HTTPS) communication. This port is reserved specifically for SSL traffic when using SLB with Anypoint VPCs.
Why Option B is Correct:
Setting the HTTP Listener's connector configuration to 8092 aligns with CloudHub requirements for HTTPS via the Shared Load Balancer.
of Incorrect Options:
Option A (8082) is used for non-HTTPS (HTTP) traffic.
Option C (80) and Option D (443) are standard web ports but are not applicable within CloudHub SLB's internal configuration for VPC routing.
Reference
For more information on the Shared Load Balancer port configurations, refer to MuleSoft's documentation on CloudHub and VPC load balancer requirements.

Question: 150

An API with multiple API implementations (Mule applications) is deployed to both CloudHub and customer-hosted Mule runtimes. All the deployments are managed by the MuleSoft-hosted control plane. An alert needs to be triggered whenever an API implementation stops responding to API requests, even if no API clients have called the API implementation for some time.
What is the most effective out-of-the-box solution to create these alerts to monitor the API implementations?

A. Create monitors in Anypoint Functional Monitoring for the API implementations, where each monitor repeatedly invokes an API implementation endpoint
B. Add code to each API client to send an Anypoint Platform REST API request to generate a custom alert in Anypoint Platform when an API invocation times out
C. Handle API invocation exceptions within the calling API client and raise an alert from that API client when such an exception is thrown
D. Configure one Worker Not Responding alert.in Anypoint Runtime Manager for all API implementations that will then monitor every API implementation

Answer: A

Explanation:
In scenarios where multiple API implementations are deployed across different environments (CloudHub and customer-hosted runtimes), Anypoint Functional Monitoring is the most effective tool to monitor API availability and trigger alerts when an API implementation becomes unresponsive. Here's how it works:
Using Anypoint Functional Monitoring:
Functional Monitoring allows you to create monitors that periodically invoke specific endpoints on the API implementations, simulating a client request. This helps ensure that the API is responsive, even if no actual client requests are being made.
If an API implementation does not respond as expected, Functional Monitoring can generate alerts, notifying administrators of potential issues.
Why Option A is Correct:
By setting up Functional Monitoring to automatically invoke the API endpoints at regular intervals, you ensure continuous monitoring and alerting capabilities, which are especially useful for APIs that may experience periods of low or no traffic. This approach provides a proactive solution, allowing you to identify and address issues before actual users are impacted.
of Incorrect Options:
Option B suggests modifying client applications to trigger alerts, which is not a best practice as it shifts monitoring responsibility to clients, reducing control and consistency.
Option C involves handling exceptions within client applications, which does not address situations where no clients are making requests.
Option D proposes a Worker Not Responding alert in Runtime Manager, which is limited to workerspecific alerts and may not reliably monitor the API's actual responsiveness to requests.
Reference
For further information, refer to MuleSoft documentation on Anypoint Functional Monitoring setup and usage for API availability monitoring.

Question: 151

An API is protected with a Client ID Enforcement policy and uses the default configuration. Access is requested for the client application to the API, and an approved contract now exists between the client application and the API
How can a consumer of this API avoid a 401 error "Unauthorized or invalid client application credentials"?

A. Send the obtained token as a header in every call
B. Send the obtained: client_id and client_secret in the request body
C. Send the obtained clent_id and clent_secret as URI parameters in every call
D. Send the obtained clent_id and client_secret in the header of every API Request call

Answer: C

Explanation:
When using the Client ID Enforcement policy with default settings, MuleSoft expects the client_id and client_secret to be provided in the URI parameters of each request. This policy is typically used to control and monitor access by validating that each request has valid credentials. Here's how to avoid a 401 Unauthorized error:
URI Parameters Requirement:

The default configuration for the Client ID Enforcement policy requires the client_id and client_secret to be included in each request's URI parameters. This is a straightforward way to authenticate API requests without additional configurations.

Why Option C is Correct:

Providing client_id and client_secret in the URI parameters meets the policy's requirements for each request, ensuring authorized access and avoiding the 401 error.

of Incorrect Options:

Option A (sending a token in the header) would be applicable for token-based authentication (like OAuth 2.0), not Client ID Enforcement.

Option B (request body) and Option D (header) are not valid locations for client_id and client_secret under the default configuration of Client ID Enforcement, which expects them in the URI.

Reference

For more details, consult MuleSoft's documentation on Client ID Enforcement policies and expected request configurations

Question: 152

A company stores financial transaction data in two legacy systems. For each legacy system, a separate, dedicated System API (SAPI) exposes data for that legacy system. A Process API (PAPI) merges the data retrieved from ail of the System APIs into a common format. Several API clients call the PAPI through its public domain name.

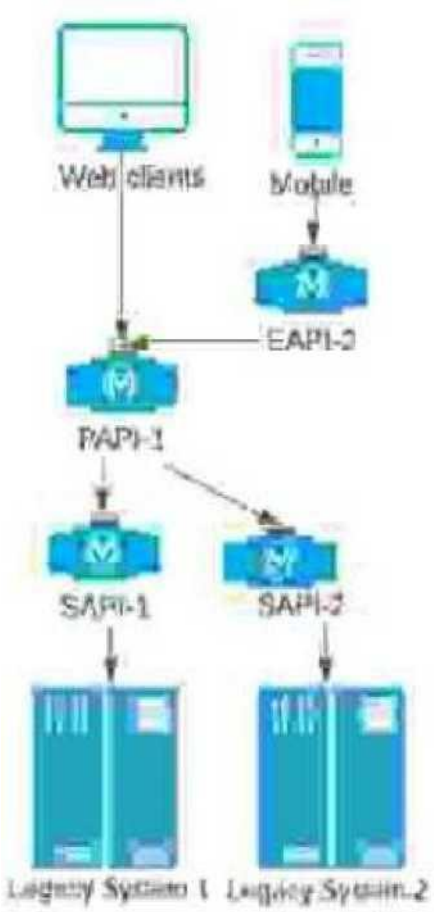
The company now wants to expose a subset of financial data to a newly developed mobile application that uses a different Bounded Context Data Model. The company wants to follow MuleSoft's best practices for building out an effective application network.

Following MuleSoft's best practices, how can the company expose financial data needed by the mobile application in a way that minimizes the impact on the currently running API clients, API implementations, and support asset reuse?

A. Add two new Experience APIs (EAPI-i and EAPI-2}.
Add Mobile PAPI-2 to expose the Intended subset of financial data as requested.
Both PAPIs access the Legacy Systems via SAPI-1 and SAP]-2.

B. Add two new Experience APIs (EAPI-i and EAPI-2}.
Add Mobile PAPI-2 to expose the Intended subset of financial data as requested.
Both PAPIs access the Legacy Systems via SAPI-1 and SAP]-2.

C. Create a new mobile Experince API (EAPI) chat exposes that subset of PAPI endpoints. Add transformtion login to the mobile Experince API implementation to make mobile data compatible with the required PAPIs.



D. Develop and deploy is new PAPI implementation with data transformation and ... login to support this required endpoints of both mobile and web clients.

Deploy an API Proxy with an endpoint from API Manager that redirect the existing PAPI endpoints to the new PAPI.

Answer: A

Explanation:

To achieve the goal of exposing financial data to a new mobile application while following MuleSoft's best practices, the company should follow an API-led connectivity approach. This approach ensures minimal disruption to existing clients, maximizes reusability, and respects the separation of concerns

across API layers.

of Solution:

Experience APIs for Client-Specific Requirements:

Create two new Experience APIs (EAPI-1 and EAPI-2) for the mobile application, tailored to meet the specific data and format requirements of the mobile application. These APIs encapsulate the clientspecific needs and provide a custom interface without impacting other clients.

Process API Layer for Data Transformation:

By adding Mobile PAPI-2, we allow the mobile application to access the required subset of data, formatted according to the mobile application's requirements. This approach ensures that data transformation and aggregation are handled in the Process layer, maintaining consistency and reusability across different applications.

Reuse of System APIs:
Both the new Mobile PAPI-2 and existing PAPI-1 access data from System APIs (SAPI-1 and SAPI-2), which continue to expose data from each legacy system in a consistent, reusable manner. This avoids duplicating logic and ensures that data access remains centralized and manageable.
Why Option A is Correct:
Option A aligns with MuleSoft's best practices by isolating client-specific requirements in the Experience layer, utilizing Process APIs for data orchestration and transformation, and maintaining reusable System APIs for backend access.
This approach also ensures that the current API clients are not impacted, as new clients (e.g., the mobile app) interact with newly defined Experience APIs without modifying the existing API setup. of Incorrect Options:
Option B: This option seems similar but lacks clarity on the separation of mobile-specific requirements and does not explicitly mention data transformation, which is essential in this scenario. Option C: Creating a single mobile Experience API that exposes a subset of PAPI endpoints directly adds unnecessary complexity and may violate the separation of concerns, as transformation logic should not be in the Experience layer.
Option D: Deploying a new PAPI and using an API Proxy to redirect existing endpoints would add unnecessary complexity, disrupt the current API clients, and increase maintenance efforts. Reference
For additional guidance, refer to MuleSoft documentation on API-led connectivity best practices and best practices for structuring Experience, Process, and System APIs.