Exam Questions MCPA-Level-1

MuleSoft Certified Platform Architect - Level 1

https://www.2passeasy.com/dumps/MCPA-Level-1/

NEW QUESTION 1
How are an API implementation, API client, and API consumer combined to invoke and process an API?

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

Answer: C

Explanation:

Correct Answer
The API consumer creates an API client, which sends API invocations to an API such that they are processed by an API implementation
********** Terminology:
>> API Client - It is a piece of code or program 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.

NEW QUESTION 2
In which layer of API-led connectivity, does the business logic orchestration reside?

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

Answer: C

Explanation:

Correct Answer
Process Layer
****
>> Experience layer is dedicated for enrichment of end user experience. This layer is to meet the needs of different API clients/ consumers.
>> System layer is dedicated to APIs which are modular in nature and implement/ expose various individual functionalities of backend systems
>> Process layer is the place where simple or complex business orchestration logic is written by invoking one or many System layer modular APIs
So, Process Layer is the right answer.

NEW QUESTION 3
True or False. We should always make sure that the APIs being designed and developed are self-servable even if it needs more man-day effort and resources.

A. FALSE
B. TRUE

Answer: B

Explanation:

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

NEW QUESTION 4
What best describes the Fully Qualified Domain Names (FQDNs), also known as DNS entries, created when a Mule application is deployed to the CloudHub Shared Worker Cloud?

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

Answer: B

Explanation:

Correct Answer
The FQDNs are determined by the application name chosen, IRRESPECTIVE of the region
****
>> When deploying applications to Shared Worker Cloud, the FQDN are always determined by application name chosen.
>> It does NOT matter what region the app is being deployed to.
>> Although it is fact and true that the generated FQDN will have the region included in it (Ex:
exp-salesorder-api.au-s1.cloudhub.io), it does NOT mean that the same name can be used when deploying to another CloudHub region.
>> Application name should be universally unique irrespective of Region and Organization and solely determines the FQDN for Shared Load Balancers.

NEW QUESTION 5
What best explains the use of auto-discovery in API implementations?

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

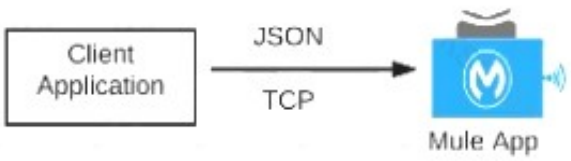
Answer: A

Explanation:

Correct Answer
It makes API Manager aware of API implementations and hence enables it to enforce policies.
****
>> API Autodiscovery is a mechanism that manages an API from API Manager by pairing the deployed application to an API created on the platform.
>> API Management includes tracking, enforcing policies if you apply any, and reporting API analytics.
>> Critical to the Autodiscovery process is identifying the API by providing the API name and version. References:
https://docs.mulesoft.com/api-manager/2.x/api-auto-discovery-new-concept https://docs.mulesoft.com/api-manager/1.x/api-auto-discovery
https://docs.mulesoft.com/api-manager/2.x/api-auto-discovery-new-concept

NEW QUESTION 6
What 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

NEW QUESTION 7
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/
So, Persistent Object Store is the right answer.

NEW QUESTION 8
Which of the following best fits the definition of API-led connectivity?

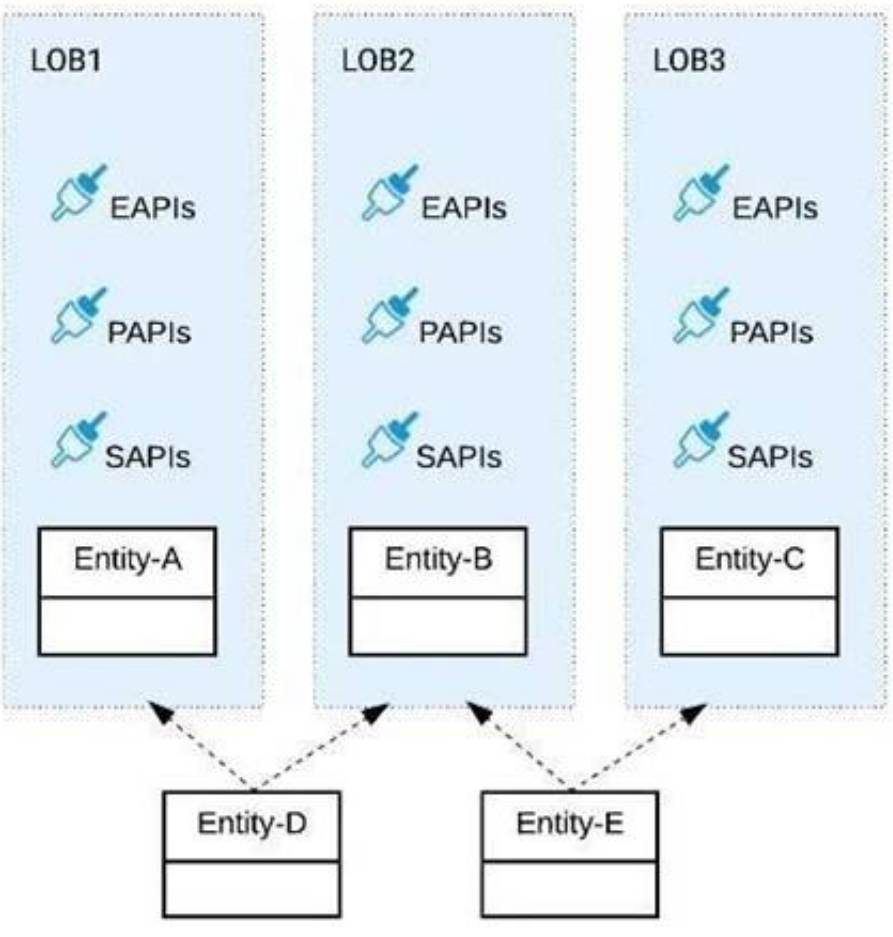
A. API-led connectivity is not just an architecture or technology but also a way to organize people and processes for efficient IT delivery in the organization
B. API-led connectivity is a 3-layered architecture covering Experience, Process and System layers
C. API-led connectivity is a technology which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

Answer: A

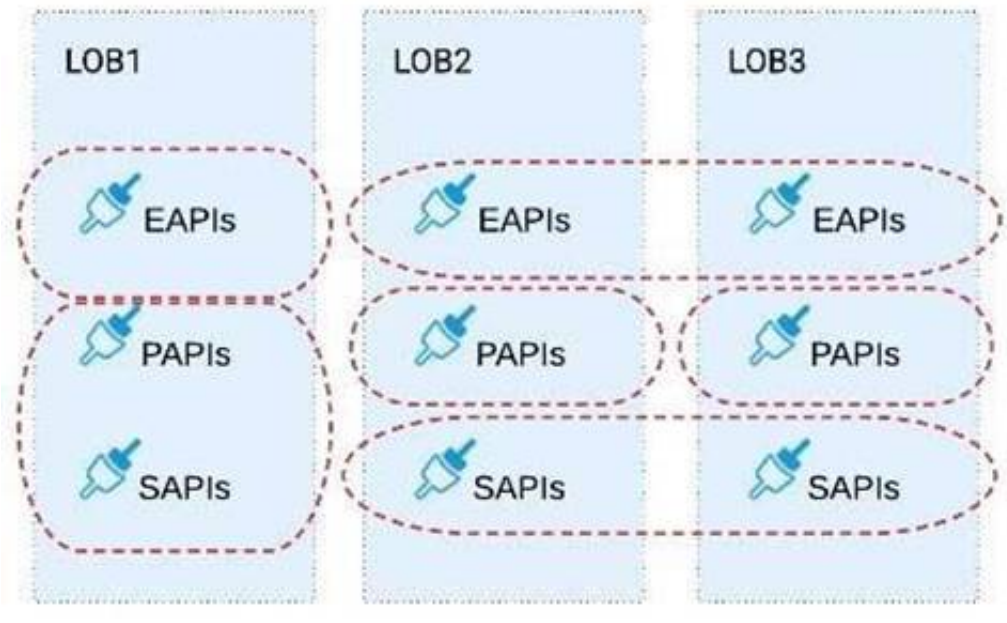
Explanation:

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

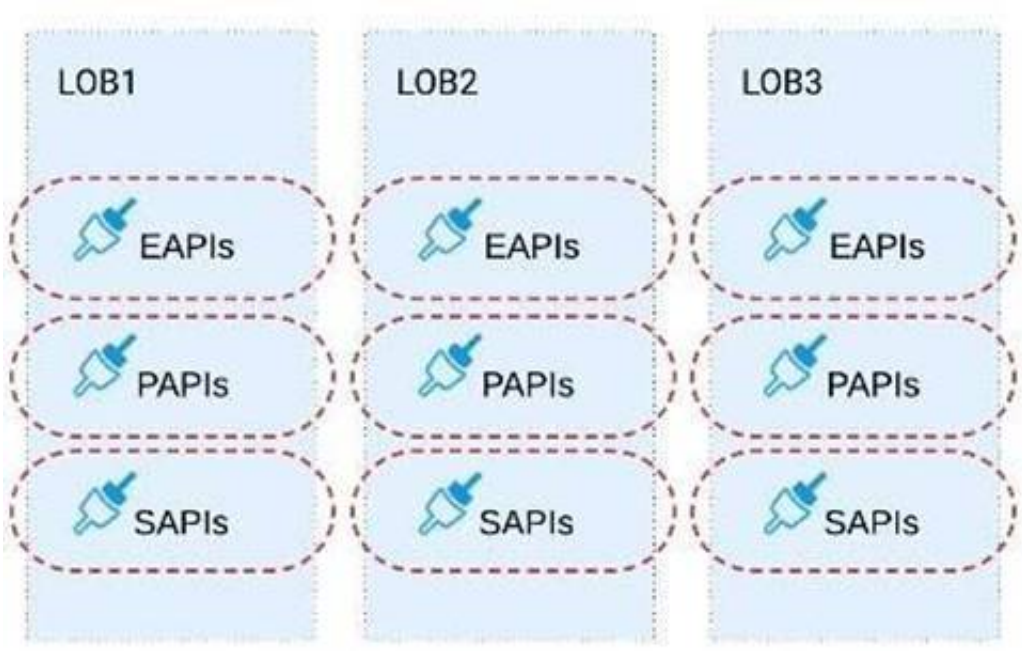
NEW QUESTION 9
Refer to the exhibit.



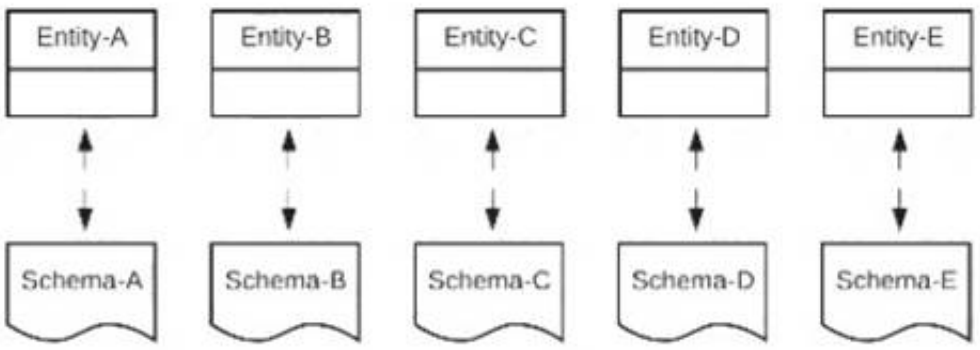
Three business processes need to be implemented, and the implementations need to communicate with several different SaaS applications.
These processes are owned by separate (siloed) LOBs and are mainly independent of each other, but do share a few business entities. Each LOB has one development team and their own budget
In this organizational context, what is the most effective approach to choose the API data models for the APIs that will implement these business processes with minimal redundancy of the data models?
A) Build several Bounded Context Data Models that align with coherent parts of the business processes and the definitions of associated business entities



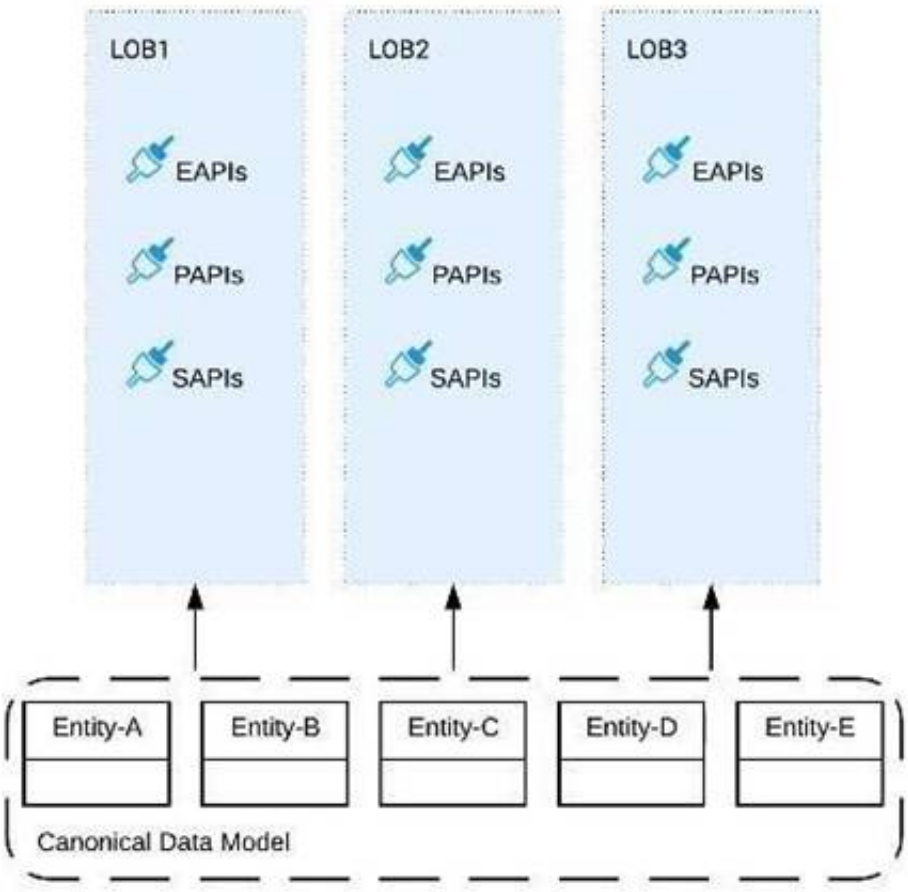
B) Build distinct data models for each API to follow established micro-services and Agile API-centric practices



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



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

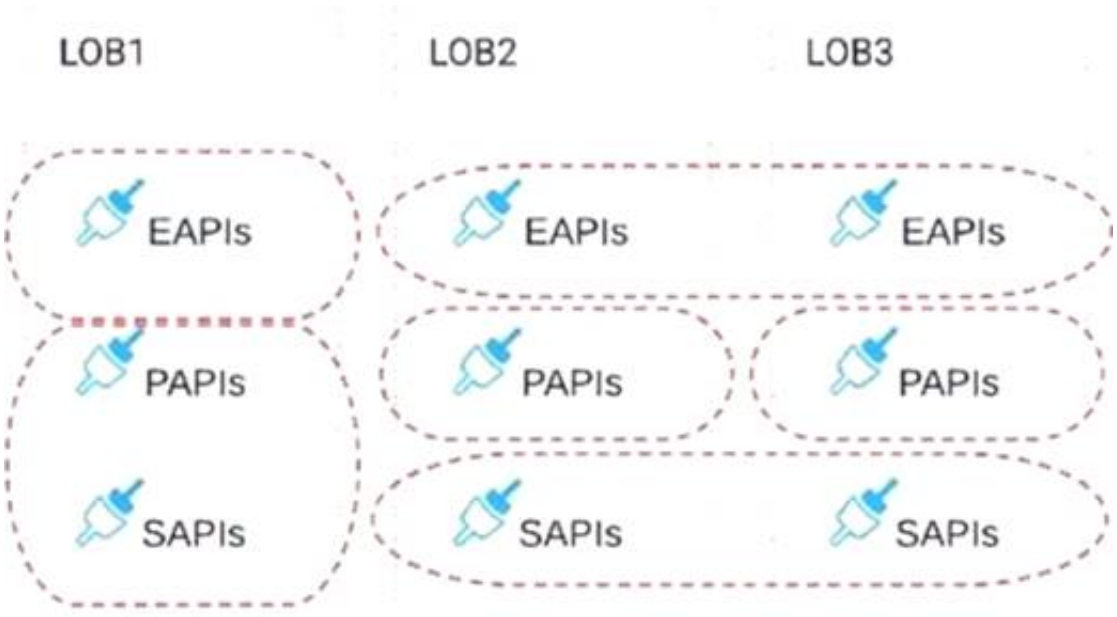


A. Option A
B. Option B
C. Option C
D. Option D

Answer: A

Explanation:

Correct Answer
Build several Bounded Context Data Models that align with coherent parts of the business processes and the definitions of associated business entities.
****
>> The options w.r.t building API data models using XML schema/ Agile API-centric practices are irrelevant to the scenario given in the question. So these two are INVALID.
>> Building EDM (Enterprise Data Model) is not feasible or right fit for this scenario as the teams and LOBs work in silo and they all have different initiatives, budget etc.. Building EDM needs intensive coordination among all the team which evidently seems not possible in this scenario.
So, the right fit for this scenario is to build several Bounded Context Data Models that align with coherent parts of the business processes and the definitions of associated business entities.



NEW QUESTION 10
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 firs
B. Upon finishing creation of all the possible assets then inform the LOBs in the organization to start using them
C. * 1. Decrease the IT delivery gap* 2. Meet various business demands by increasing the IT capacity and forming various IT departments* 3. Make consumption of assets at the rate of production
D. * 1. Decrease the IT delivery gap* 2. Meet various business demands without increasing the IT capacity* 3. Make consumption of assets at the rate of production

Answer: C

Explanation:

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

NEW QUESTION 10
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.
********** Diagram, arrow Description automatically generated

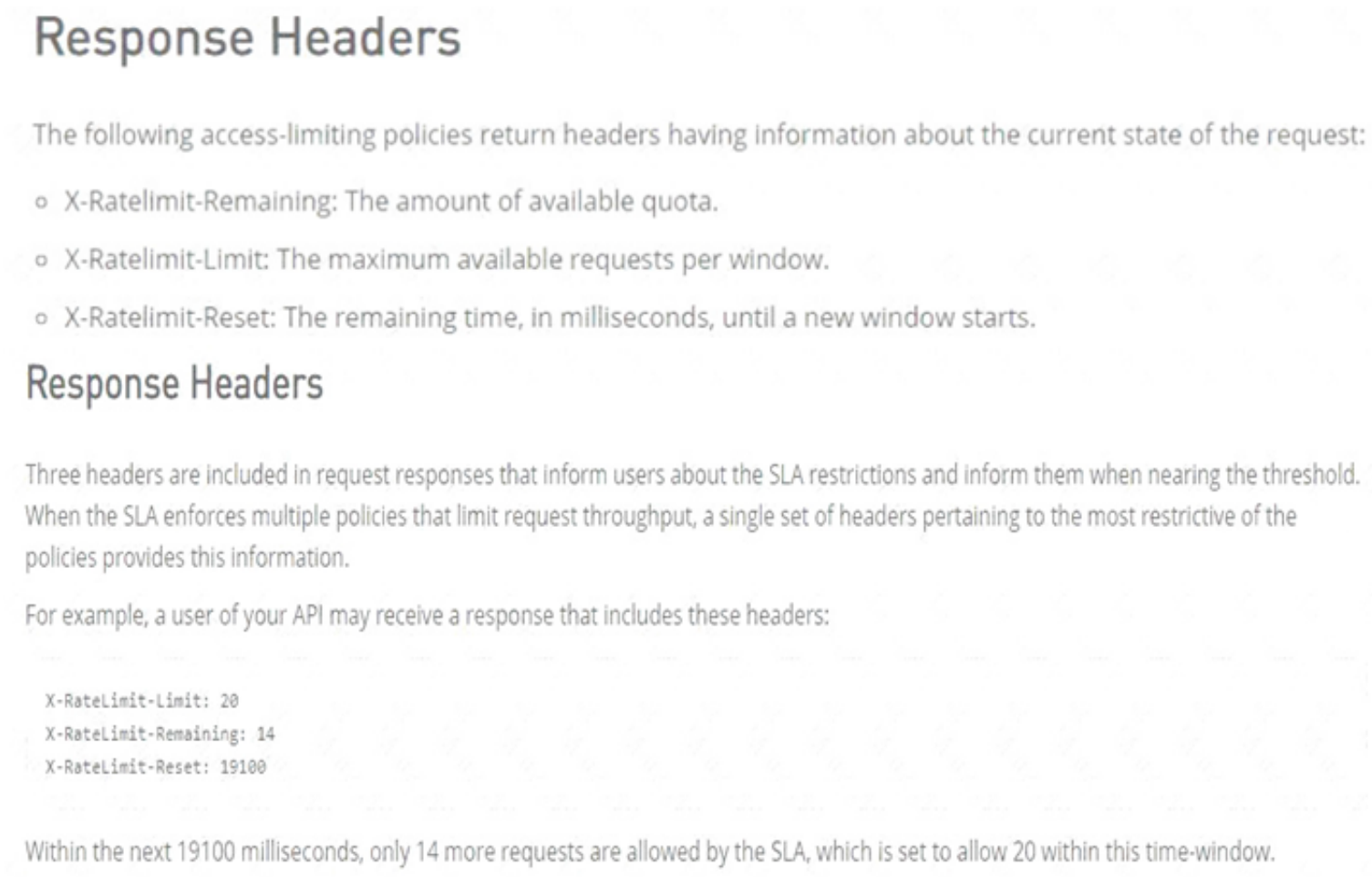
NEW QUESTION 11
How can the application of a rate limiting API policy be accurately reflected in the RAML definition of an API?

A. By refining the resource definitions by adding a description of the rate limiting policy behavior
B. By refining the request definitions by adding a remaining Requests query parameter with description, type, and example

C. By refining the response definitions by adding the out-of-the-box Anypoint Platform rate-limit-enforcement securityScheme with description, type, and example
D. By refining the response definitions by adding the x-ratelimit-* response headers with description, type, and example

Answer: D

Explanation:
Correct Answer
By refining the response definitions by adding the x-ratelimit-* response headers with description, type, and example
****



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

NEW QUESTION 13
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.

NEW QUESTION 18
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.

NEW QUESTION 22
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.

NEW QUESTION 24
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.

NEW QUESTION 28
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.

NEW QUESTION 29
A Mule application exposes an HTTPS endpoint and is deployed to the CloudHub Shared Worker Cloud. All traffic to that Mule application must stay inside the AWS VPC.
To what TCP port do API invocations to that Mule application need to be sent?

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

Answer: D

Explanation:
Correct Answer 8082
****
>> 8091 and 8092 ports are to be used when keeping your HTTP and HTTPS app private to the LOCAL VPC respectively.
>> Above TWO ports are not for Shared AWS VPC/ Shared Worker Cloud.
>> 8081 is to be used when exposing your HTTP endpoint app to the internet through Shared LB
>> 8082 is to be used when exposing your HTTPS endpoint app to the internet through Shared LB So, API invocations should be sent to port 8082 when calling this HTTPS based app.
References:
https://docs.mulesoft.com/runtime-manager/cloudhub-networking-guide https://help.mulesoft.com/s/article/Configure-Cloudhub-Application-to-Send-a-HTTPS-Request-Directly-to-An
https://help.mulesoft.com/s/question/0D52T00004mXXULSA4/multiple-http-listerners-on-cloudhub-one-with-p

NEW QUESTION 30
An API client calls one method from an existing API implementation. The API implementation is later updated. What change to the API implementation would require the API client's invocation logic to also be updated?

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

Answer: C

Explanation:

Correct Answer
When a new required field is added to the method called by the API client
****
>> Generally, the logic on API clients need to be updated when the API contract breaks.
>> When a new method or a child method is added to an API , the API client does not break as it can still continue to use its existing method. So these two options are out.
>> We are left for two more where "datatype of the response if changed" and "a new required field is added".
>> Changing the datatype of the response does break the API contract. However, the question is insisting on the "invocation" logic and not about the response handling logic. The API client can still invoke the API successfully and receive the response but the response will have a different datatype for some field.
>> Adding a new required field will break the API's invocation contract. When adding a new required field, the API contract breaks the RAML or API spec agreement that the API client/API consumer and API provider has between them. So this requires the API client invocation logic to also be updated.

NEW QUESTION 31
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 deployedon-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..

NEW QUESTION 34
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.

NEW QUESTION 37
A set of tests must be performed prior to deploying API implementations to a staging environment. Due to data security and access restrictions, untested APIs cannot be granted access to the backend systems, so instead mocked data must be used for these tests. The amount of available mocked data and its contents is sufficient to entirely test the API implementations with no active connections to the backend systems. What type of tests should be used to incorporate this mocked data?

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

Answer: D

Explanation:

Correct Answer
Unit tests (Whitebox)
****

NEW QUESTION 38
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.

NEW QUESTION 40
An API has been updated in Anypoint Exchange by its API producer from version 3.1.1 to 3.2.0 following accepted semantic versioning practices and the changes have been communicated via the API's public portal.
The API endpoint does NOT change in the new version.
How should the developer of an API client respond to this change?

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

Answer: D

NEW QUESTION 44
An organization wants MuleSoft-hosted runtime plane features (such as HTTP load balancing, zero downtime, and horizontal and vertical scaling) in its Azure environment. What runtime plane minimizes the organization's effort to achieve these features?

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

Answer: A

Explanation:

Correct Answer
Anypoint Runtime Fabric
****
>> When a customer is already having an Azure environment, It is not at all an ideal approach to go with hybrid model having some Mule Runtimes hosted on Azure and some on MuleSoft. This is unnecessary and useless.
>> CloudHub is a Mulesoft-hosted Runtime plane and is on AWS. We cannot customize to point CloudHub to customer's Azure environment.
>> Anypoint Platform for Pivotal Cloud Foundry is specifically for infrastructure provided by Pivotal Cloud Foundry
>> Anypoint Runtime Fabric is right answer as it is a container service that automates the deployment and orchestration of Mule applications and API gateways. Runtime Fabric runs within a customer-managed infrastructure on AWS, Azure, virtual machines (VMs), and bare-metal servers.
-Some of the capabilities of Anypoint Runtime Fabric include:
-Isolation between applications by running a separate Mule runtime per application.
-Ability to run multiple versions of Mule runtime on the same set of resources.
-Scaling applications across multiple replicas.
-Automated application fail-over.
-Application management with Anypoint Runtime Manager.

NEW QUESTION 45
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

NEW QUESTION 50
An organization wants to make sure only known partners can invoke the organization's APIs. To achieve this security goal, the organization wants to enforce a Client ID Enforcement policy in API Manager so that only registered partner applications can invoke the organization's APIs. In what type of API implementation does MuleSoft recommend adding an API proxy to enforce the Client ID Enforcement policy, rather than embedding the policy directly in the application's JVM?

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

Answer: D

Explanation:
Correct Answer
A Non-Mule application
****
>> All type of Mule applications (Mule 3/ Mule 4/ with APIkit/ with Custom Java Code etc) running on Mule Runtimes support the Embedded Policy Enforcement on them.
>> The only option that cannot have or does not support embedded policy enforcement and must have API Proxy is for Non-Mule Applications.
So, Non-Mule application is the right answer.

NEW QUESTION 54
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

APIs use HTTP-based protocols: cached HTTP responses from previous HTTP requests may potentially be returned if the same HTTP request is seen again.

Safe HTTP methods are ones that do not alter the state of the underlying resource. That is, the *HTTP responses to requests using safe HTTP methods may be cached*.

The HTTP standard requires the following HTTP methods on any resource to be safe:

- GET
- HEAD
- OPTIONS

Safety must be honored by REST APIs (but not by non-REST APIs like SOAP APIs): It is the *responsibility of every API implementation* to implement GET, HEAD or OPTIONS methods such that they never change the state of a resource.

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

NEW QUESTION 59
A new upstream API Is being designed to offer an SLA of 500 ms median and 800 ms maximum (99th percentile) response time. The corresponding API implementation needs to sequentially invoke 3 downstream APIs of very similar complexity.
The first of these downstream APIs offers the following SLA for its response time: median: 100 ms, 80th percentile: 500 ms, 95th percentile: 1000 ms.
If possible, how can a timeout be set in the upstream API for the invocation of the first downstream API to meet the new upstream API's desired SLA?

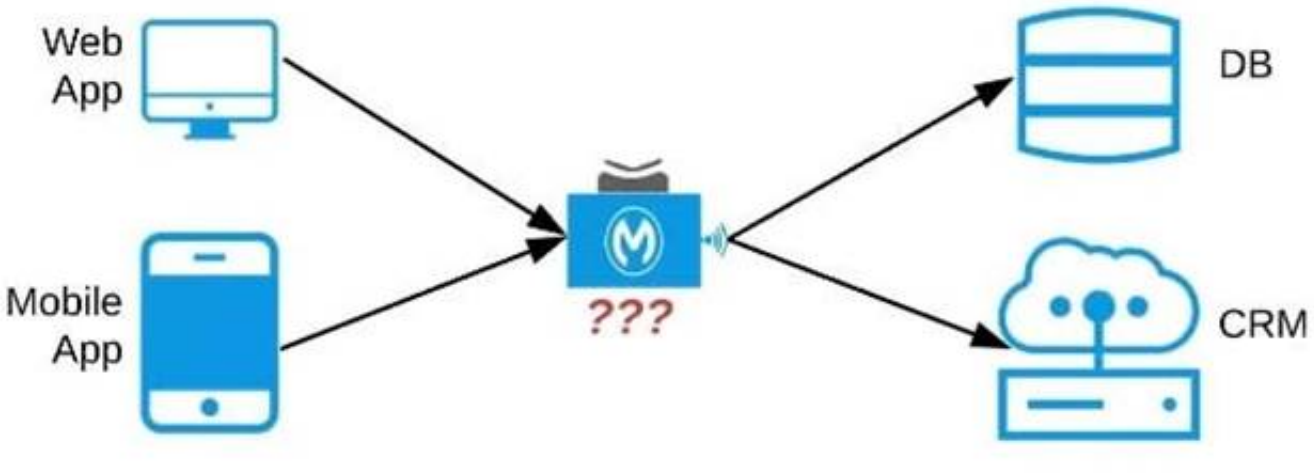
A. Set a timeout of 50 ms; this times out more invocations of that API but gives additional room for retries
B. Set a timeout of 100 ms; that leaves 400 ms for the other two downstream APIs to complete
C. No timeout is possible to meet the upstream API's desired SLA; a different SLA must be negotiated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

Answer: B

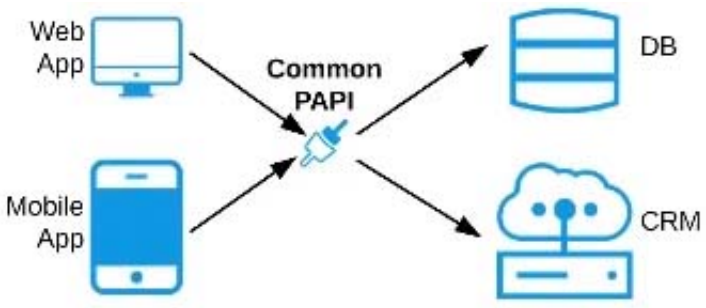
Explanation:

Correct Answer
Set a timeout of 100ms; that leaves 400ms for other two downstream APIs to complete
********** Key details to take from the given scenario:
>> Upstream API's designed SLA is 500ms (median). Lets ignore maximum SLA response times.
>> This API calls 3 downstream APIs sequentially and all these are of similar complexity.
>> The first downstream API is offering median SLA of 100ms, 80th percentile: 500ms; 95th percentile: 1000ms.
Based on the above details:
>> We can rule out the option which is suggesting to set 50ms timeout. Because, if the median SLA itself being offered is 100ms then most of the calls are going to timeout and time gets wasted in retried them and eventually gets exhausted with all retries. Even if some retries gets successful, the remaining time wont leave enough room for 2nd and 3rd downstream APIs to respond within time.
>> The option suggesting to NOT set a timeout as the invocation of this API is mandatory and so we must wait until it responds is silly. As not setting time out would go against the good implementation pattern and moreover if the first API is not responding within its offered median SLA 100ms then most probably it would either respond in 500ms (80th percentile) or 1000ms (95th percentile). In BOTH cases, getting a successful response from 1st downstream API does NO GOOD because already by this time the Upstream API SLA of 500 ms is breached. There is no time left to call 2nd and 3rd downstream APIs.
>> It is NOT true that no timeout is possible to meet the upstream APIs desired SLA.
As 1st downstream API is offering its median SLA of 100ms, it means MOST of the time we would get the responses within that time. So, setting a timeout of 100ms would be ideal for MOST calls as it leaves enough room of 400ms for remaining 2 downstream API calls.

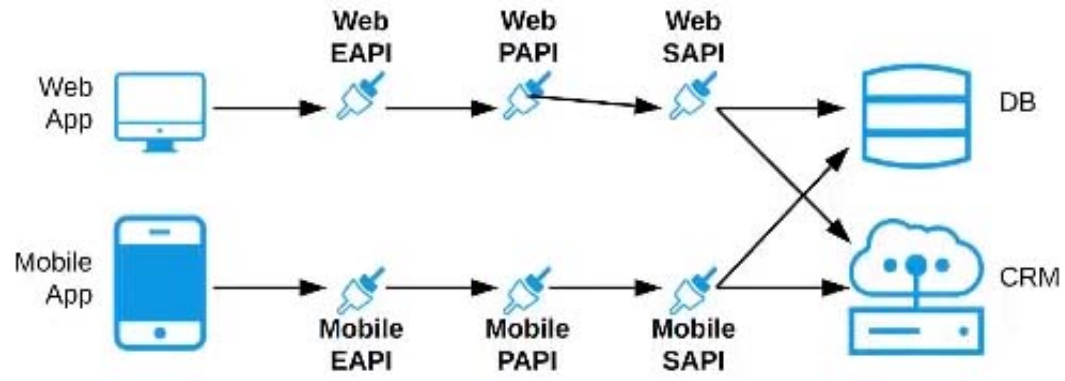
NEW QUESTION 60
Refer to the exhibit. An organization needs to enable access to their customer data from both a mobile app and a web application, which each need access to common fields as well as certain unique fields.
The data is available partially in a database and partially in a 3rd-party CRM system.
What APIs should be created to best fit these design requirements?



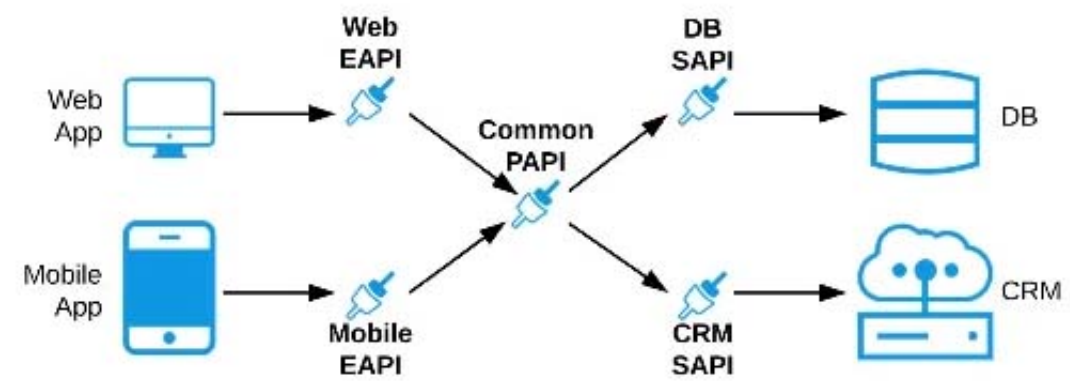
A) A Process API that contains the data required by both the web and mobile apps, allowing these applications to invoke it directly and access the data they need thereby providing the flexibility to add more fields in the future without needing API changes



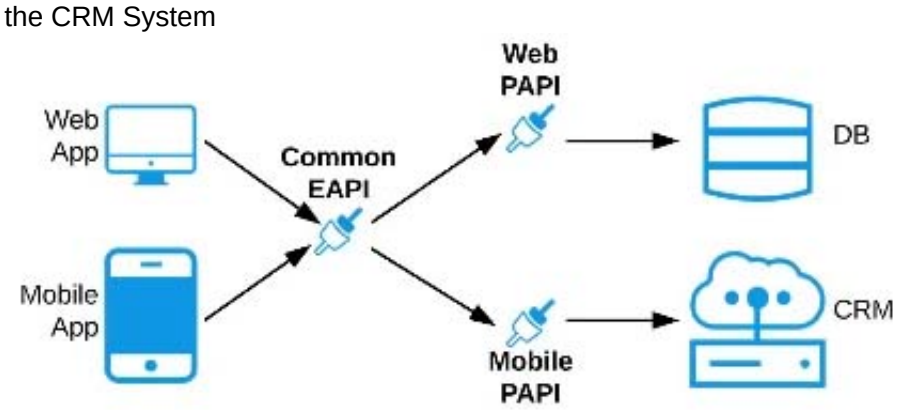
B) One set of APIs (Experience API, Process API, and System API) for the web app, and another set for the mobile app



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



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



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

Answer: C

Explanation:

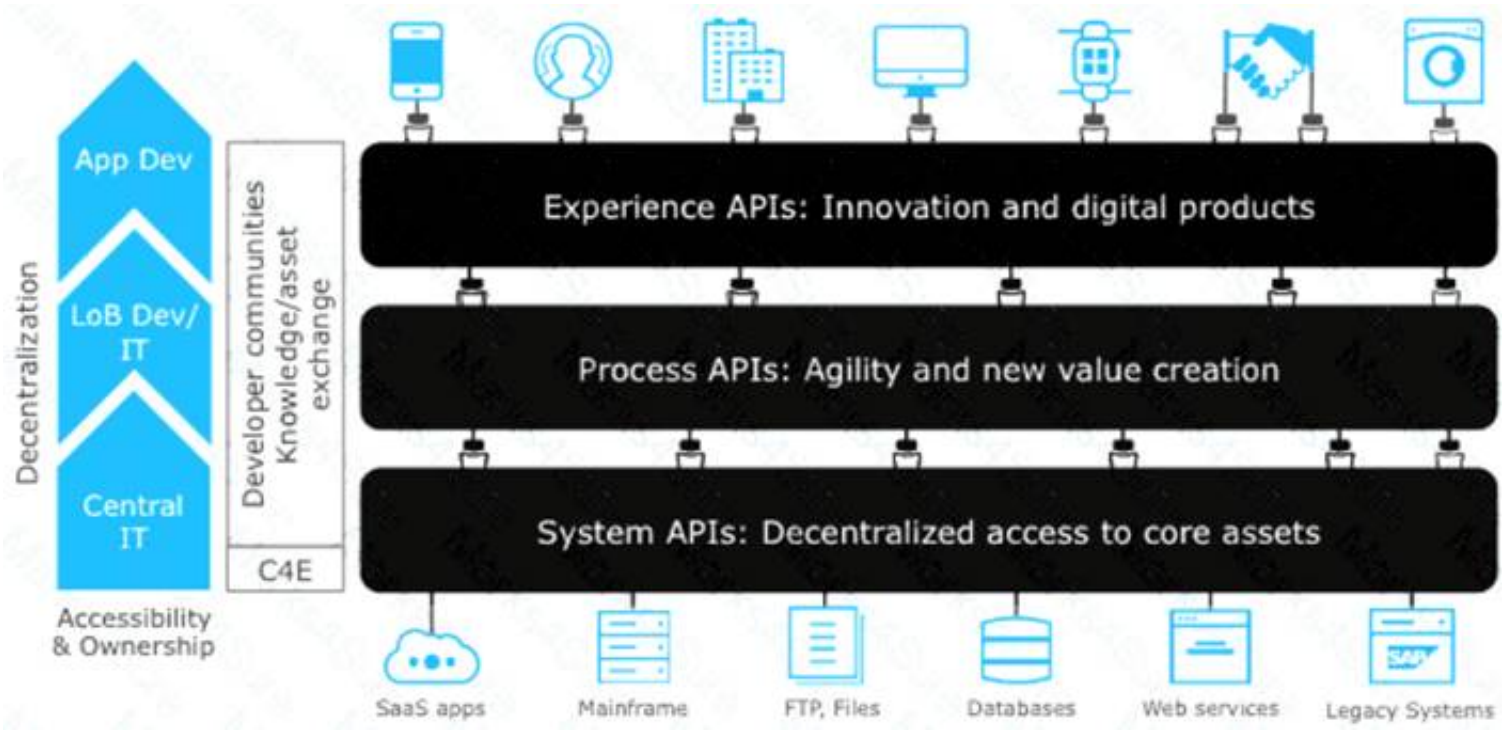
Correct Answer
Separate Experience APIs for the mobile and web app, but a common Process API that invokes separate System APIs created for the database and CRM system
********** As per MuleSoft's API-led connectivity:
>> Experience APIs should be built as per each consumer needs and their experience.
>> Process APIs should contain all the orchestration logic to achieve the business functionality.
>> System APIs should be built for each backend system to unlock their data.

NEW QUESTION 62
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.

NEW QUESTION 63
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

NEW 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

NEW QUESTION 68
An organization has several APIs that accept JSON data over HTTP POST. The APIs are all publicly available and are associated with several mobile applications and web applications.
The organization does NOT want to use any authentication or compliance policies for these APIs, but at the same time, is worried that some bad actor could send payloads that could somehow compromise the applications or servers running the API implementations.
What out-of-the-box Anypoint Platform policy can address exposure to this threat?

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

Answer: D

Explanation:
Correct Answer
Apply a JSON threat protection policy to all APIs to detect potential threat vectors
****
>> Usually, if the APIs are designed and developed for specific consumers (known consumers/customers) then we would IP Whitelist the same to ensure that traffic only comes from them.
>> However, as this scenario states that the APIs are publicly available and being used by so many mobile and web applications, it is NOT possible to identify and blacklist all possible bad actors.
>> So, JSON threat protection policy is the best chance to prevent any bad JSON payloads from such bad actors.

NEW QUESTION 73
A company has started to create an application network and is now planning to implement a Center for Enablement (C4E) organizational model. What key factor would lead the company to decide upon a federated rather than a centralized C4E?

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

Answer: C

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

NEW QUESTION 75
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

NEW QUESTION 76
A company wants to move its Mule API implementations into production as quickly as possible. To protect access to all Mule application data and metadata, the company requires that all Mule applications be deployed to the company's customer-hosted infrastructure within the corporate firewall. What combination of runtime plane and control plane options meets these project lifecycle goals?

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

Answer: A

Explanation:
Correct Answer
Manually provisioned customer-hosted runtime plane and customer-hosted control plane
****
There are two key factors that are to be taken into consideration from the scenario given in the question.
>> Company requires both data and metadata to be resided within the corporate firewall
>> Company would like to go with customer-hosted infrastructure.
Any deployment model that is to deal with the cloud directly or indirectly (Mulesoft-hosted or Customer's own cloud like Azure, AWS) will have to share atleast the

metadata.
Application data can be controlled inside firewall by having Mule Runtimes on customer hosted runtime plane. But if we go with Mulsoft-hosted/ Cloud-based control plane, the control plane required atleast some minimum level of metadata to be sent outside the corporate firewall.
As the customer requirement is pretty clear about the data and metadata both to be within the corporate firewall, even though customer wants to move to production as quickly as possible, unfortunately due to the nature of their security requirements, they have no other option but to go with manually provisioned customer-hosted runtime plane and customer-hosted control plane.

NEW QUESTION 79
The implementation of a Process API must change.
What is a valid approach that minimizes the impact of this change on API clients?

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

Answer: C

Explanation:
Correct Answer
Implement required changes to the Process API implementation so that, whenever possible, the Process API’s RAML definition remains unchanged.
********** Key requirement in the question is:
>> Approach that minimizes the impact of this change on API clients Based on above:
>> Updating the RAML definition would possibly impact the API clients if the changes require any thing mandatory from client side. So, one should try to avoid doing that until really necessary.
>> Implementing the changes as a completely different API and then redirectly the clients with 3xx status code is really upsetting design and heavily impacts the API clients.
>> Organisations and IT cannot simply postpone the changes required until all API consumers acknowledge they are ready to migrate to a new Process API or API version. This is unrealistic and not possible.
The best way to handle the changes always is to implement required changes to the API implementations so that, whenever possible, the API’s RAML definition remains unchanged.

NEW QUESTION 80
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 coarse-grained 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.

NEW QUESTION 83
What is true about where an API policy is defined in Anypoint Platform and how it is then applied to API instances?

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

Answer: B

Explanation:

Correct Answer
The API policy is defined in API Manager for a specific API instance, and then ONLY applied to the specific API instance.
****
>> Once our API specifications are ready and published to Exchange, we need to visit API Manager and register an API instance for each API.
>> API Manager is the place where management of API aspects takes place like addressing NFRs by enforcing policies on them.
>> We can create multiple instances for a same API and manage them differently for different purposes.
>> One instance can have a set of API policies applied and another instance of same API can have different set of policies applied for some other purpose.

>> These APIs and their instances are defined PER environment basis. So, one need to manage them seperately in each environment.
>> We can ensure that same configuration of API instances (SLAs, Policies etc..) gets promoted when promoting to higher environments using platform feature. But this is optional only. Still one can change them per environment basis if they have to.
>> Runtime Manager is the place to manage API Implementations and their Mule Runtimes but NOT APIs itself. Though API policies gets executed in Mule Runtimes, We CANNOT enforce API policies in Runtime Manager. We would need to do that via API Manager only for a cherry picked instance in an environment.
So, based on these facts, right statement in the given choices is - "The API policy is defined in API Manager for a specific API instance, and then ONLY applied to the specific API instance".

NEW QUESTION 86
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.

NEW QUESTION 87
.....

THANKS FOR TRYING THE DEMO OF OUR PRODUCT

Visit Our Site to Purchase the Full Set of Actual MCPA-Level-1 Exam Questions With Answers.

We Also Provide Practice Exam Software That Simulates Real Exam Environment And Has Many Self-Assessment Features. Order the MCPA-Level-1 Product From:

https://www.2passeasy.com/dumps/MCPA-Level-1/

Money Back Guarantee

MCPA-Level-1 Practice Exam Features:

* MCPA-Level-1 Questions and Answers Updated Frequently

* MCPA-Level-1 Practice Questions Verified by Expert Senior Certified Staff

* MCPA-Level-1 Most Realistic Questions that Guarantee you a Pass on Your FirstTry

* MCPA-Level-1 Practice Test Questions in Multiple Choice Formats and Updatesfor 1 Year