MCPA-LEVEL1 Q&As

MuleSoft Certified Platform Architect - Level 1

QUESTION 1

What Anypoint Connectors support transactions?

A. Database, JMS, VM

B. Database, 3MS, HTTP

C. Database, JMS, VM, SFTP

D. Database, VM, File

Correct Answer: A

QUESTION 2

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

Correct Answer: A

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 3

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

Correct Answer: B

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.

QUESTION 4

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

Correct Answer: 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 *** >> 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 5

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

Correct Answer: C

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