MCPA-LEVEL-1-MAINTENANCE Q&As

MuleSoft Certified Platform Architect - Level 1 MAINTENANCE

Pass Mulesoft MCPA-LEVEL-1-MAINTENANCE Exam with 100% Guarantee

Free Download Real Questions & Answers **PDF** and **VCE** file from:

https://www.geekcert.com/mcpa-level-1-maintenance.html

100% Passing Guarantee
100% Money Back Assurance

Following Questions and Answers are all new published by Mulesoft Official Exam Center

- **Instant Download** After Purchase
- **100% Money Back** Guarantee
- **365 Days** Free Update
- **800,000+** Satisfied Customers

QUESTION 1

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

Correct Answer: C

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

****

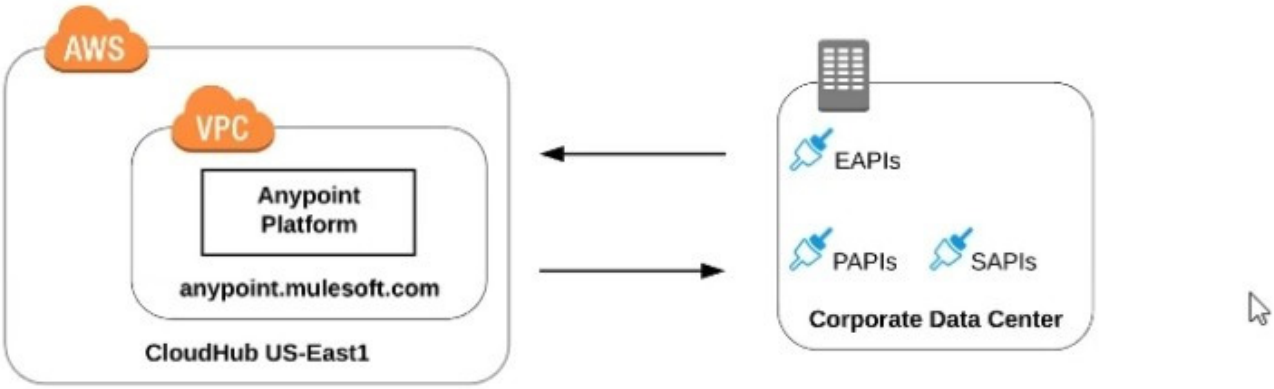
Key details to note from the given scenario:

>> 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 2

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

Correct Answer: C

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

- Load balancing

  Load balancing is not provided for hybrid deployments. You can manage load balancing with the tools connected to your on-premises resources.

>> For Hybrid deployment models, the on-premises are first connected to Runtime Manager using Runtime Manager agent. So, the connection is initiated first from On- premises to Runtime Manager. Then all control can be done from Runtime Manager. >> Anypoint Runtime Manager CANNOT ensure automatic HA. Clusters/Server Groups etc should be configured before hand.

Only TRUE statement in the given choices is, API implementations can run successfully in customer-hosted Mule runtimes, even when they are unable to communicate with the control plane. There are several references below to justify this statement.

References: https://docs.mulesoft.com/runtime-manager/deployment-strategies#hybrid-deployments https://help.mulesoft.com/s/article/On-Premise-Runtimes-Disconnected-From-US-Control- Plane-June-18th-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-

RuntimeManager-May-29th-2018

On-Premise Runtimes Disconnected From US Control Plane - June 18th 2018

Jun 19, 2018 . RCA

Content

Impacted Platforms	Impacted Duration
Anypoint Runtime Manager / On-Prem Runtimes	During this time frame, on-prem runtimes appeared disconnected from the US Anypoint Control Plane: June 18, 2018 10:35 AM PST to June 18, 2018 11:12 AM PST

Incident Description

On-premises applications weren't able to connect to Anypoint Runtime Manager during the length of the incident, which made on-premises runtimes to threw errors in their logs because they received network disconnect messages from the control plane. Other than generating the log as mentioned above entries, on-premises runtimes and applications were not impacted.

==========

Runtime Manager cannot manage On-Prem Applications and Servers from US Control Plane - June 25th 2019

Jul 3, 2019 . RCA

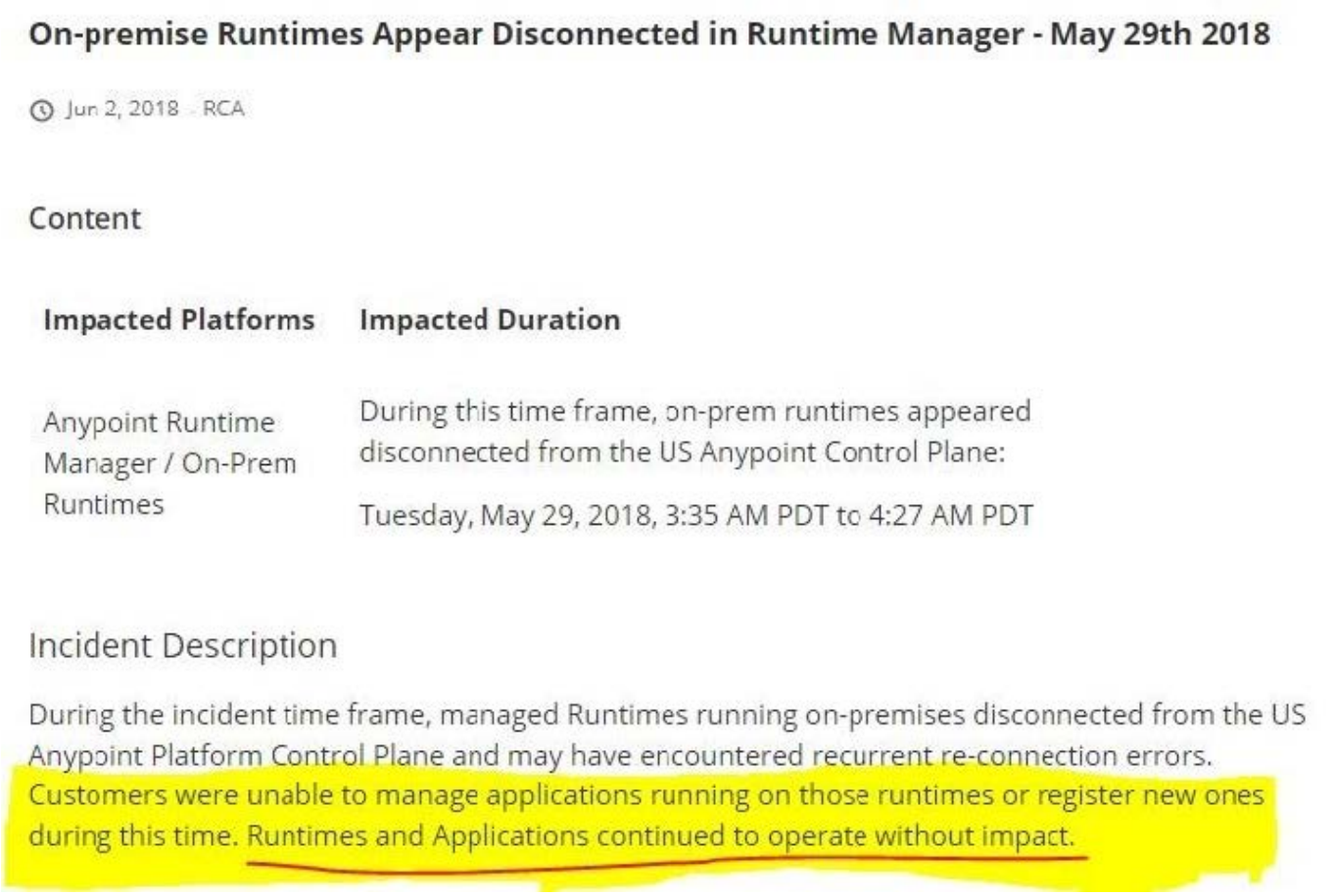
Content

Incident Summary

Between 2:51 p.m. PT June 25th and 12:41 a.m. PT June 26th, customers were not able to manage their On-Prem applications and servers. The availability of running applications and runtimes were not impacted.

Impacted Platforms	Impact Duration
US-Prod	9 hours and 50 minutes

==========


On-premise Runtimes Appear Disconnected in Runtime Manager - May 29th 2018

Jun 2, 2018 · RCA

Content

Impacted Platforms	Impacted Duration
Anypoint Runtime Manager / On-Prem Runtimes	During this time frame, on-prem runtimes appeared disconnected from the US Anypoint Control Plane: Tuesday, May 29, 2018, 3:35 AM PDT to 4:27 AM PDT

Incident Description

During the incident time frame, managed Runtimes running on-premises disconnected from the US Anypoint Platform Control Plane and may have encountered recurrent re-connection errors. Customers were unable to manage applications running on those runtimes or register new ones during this time. Runtimes and Applications continued to operate without impact.

QUESTION 3

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

A. TRUE

B. FALSE

Correct Answer: A

********** >> 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 4

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

Correct Answer: B

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 5

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

Correct Answer: 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

****

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.