MCPA-LEVEL1 Q&As

MuleSoft Certified Platform Architect - Level 1

Pass Mulesoft MCPA-LEVEL1 Exam with 100% Guarantee

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

https://www.pass4itsure.com/mulesoft-certified-platform-architect-level-1.html

100% Passing Guarantee
100% Money Back Assurance

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

QUESTION 1

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

Correct Answer: B

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.

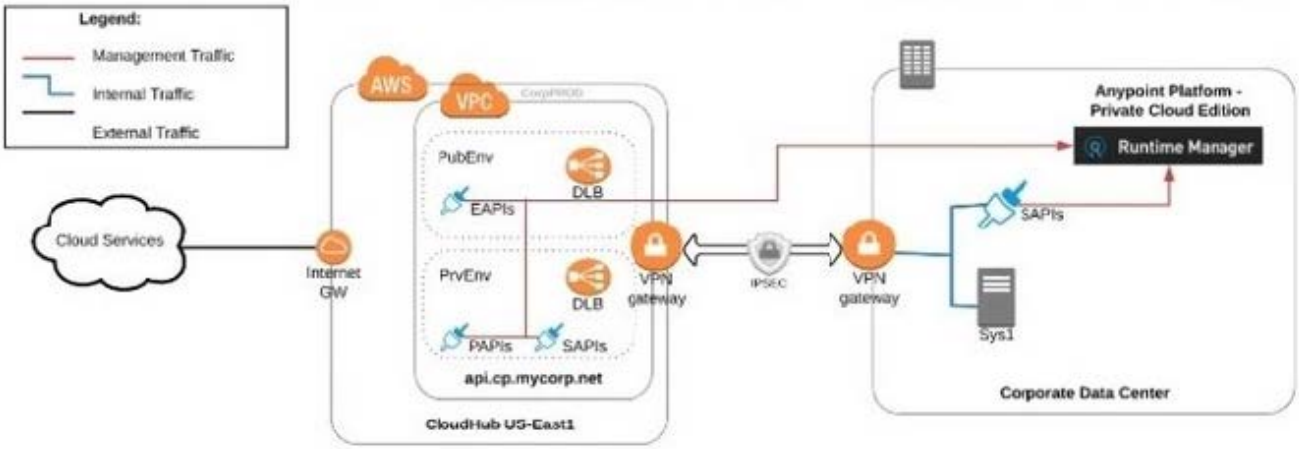
So, the only right option to go with and fits the purpose of scenario being most cost- effective is - using a CloudHub Shared Load Balancer.

QUESTION 2

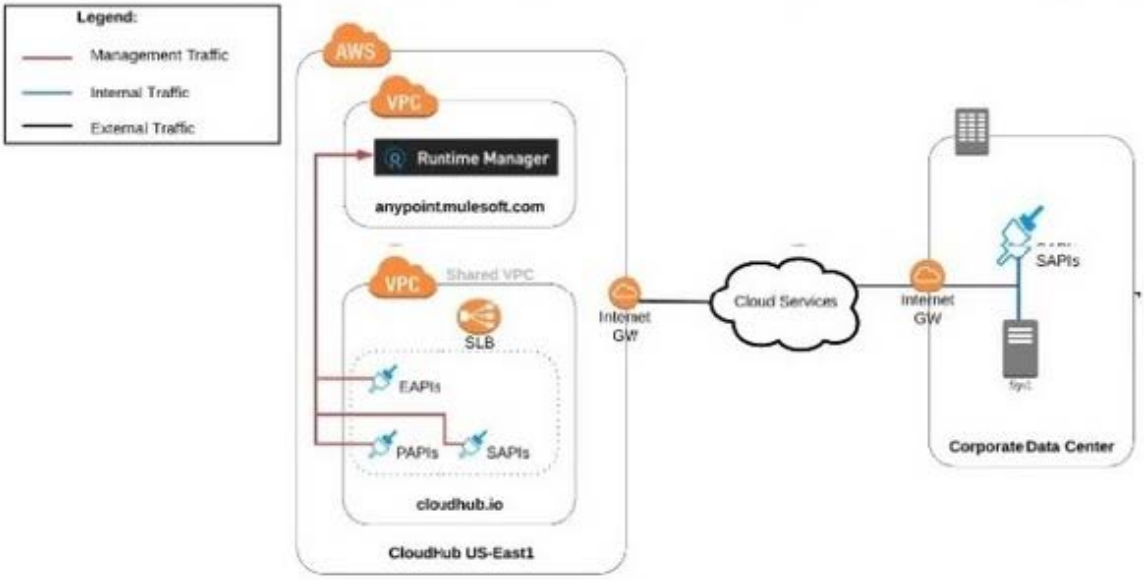
An organization uses various cloud-based SaaS systems and multiple on-premises systems. The on-premises systems are an important part of the organization\\'s application network and can only be accessed from within the organization\\'s intranet.

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

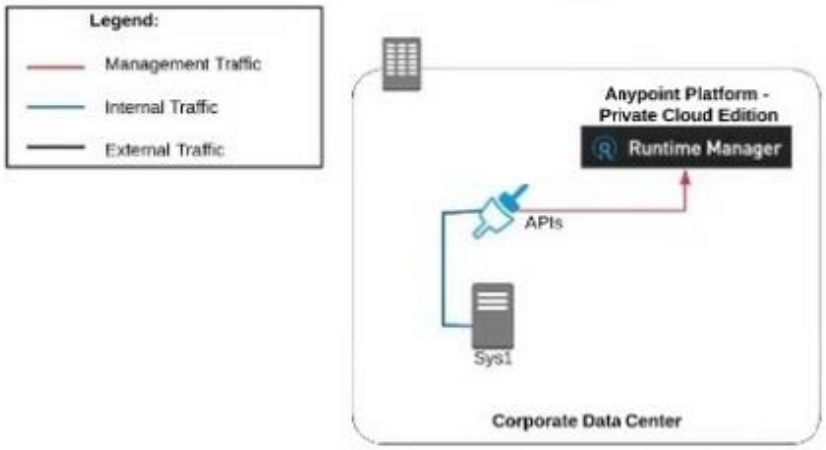
A. Use CloudHub-deployed Mule runtimes in an Anypoint VPC managed by Anypoint Platform Private Cloud Edition control plane



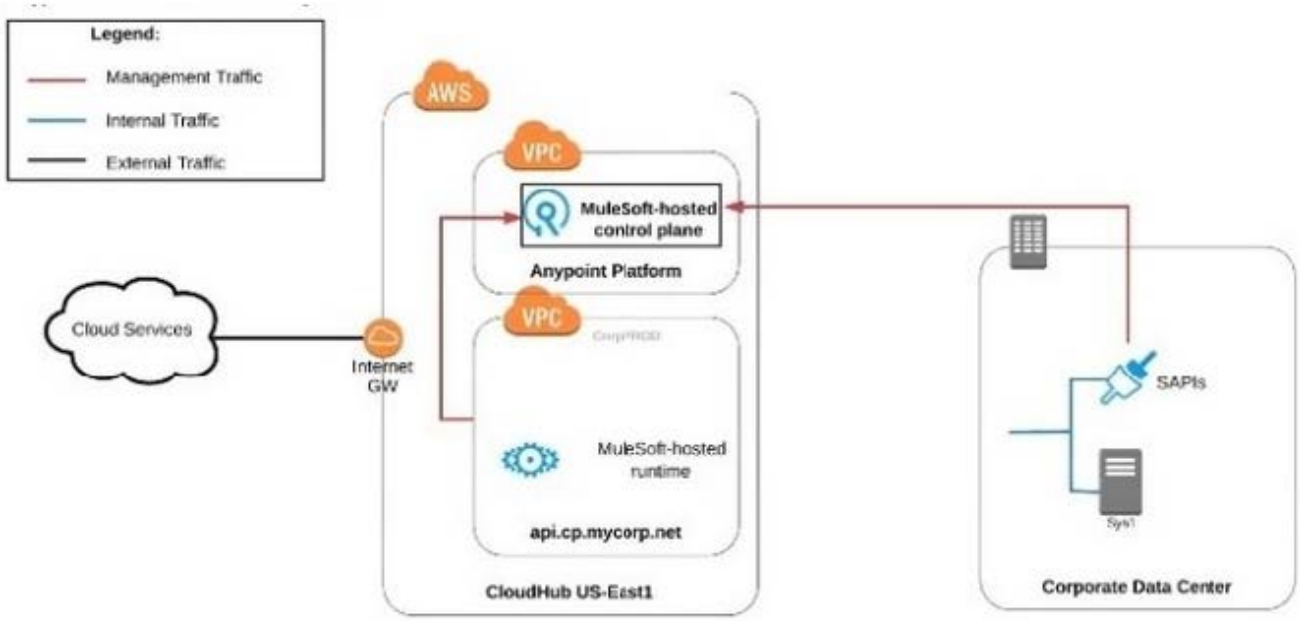
B. Use CloudHub-deployed Mule runtimes in the shared worker cloud managed by the MuleSoft-hosted Anypoint Platform control plane



C. Use an on-premises installation of Mule runtimes that are completely isolated with NO external network access, managed by the Anypoint Platform Private Cloud Edition control plane



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



A. Option A

B. Option B

C. Option C

D. Option D

Correct Answer: B

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 3

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

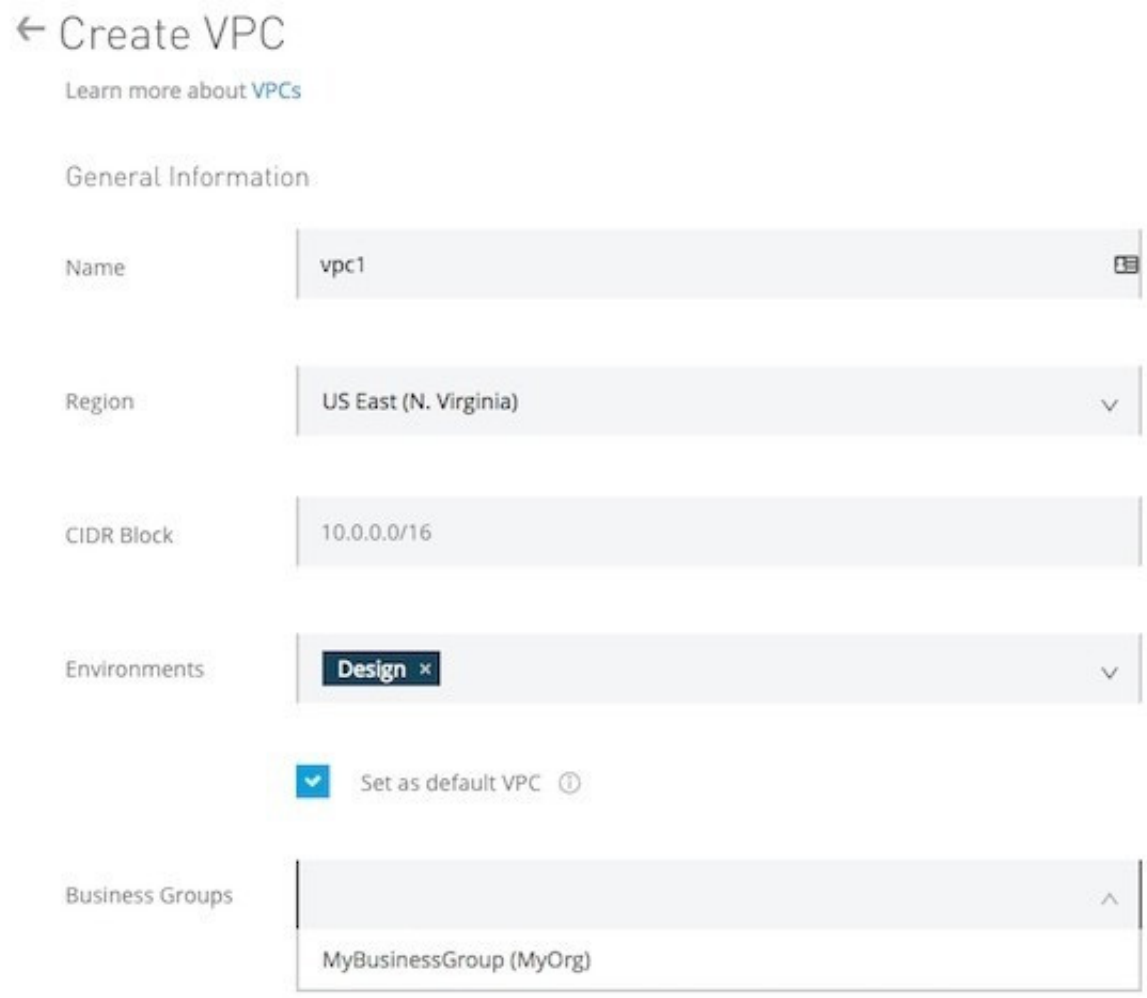
Correct Answer: B

Traffic between Mule applications deployed to an Anypoint VPC and on- premises systems can stay within a private network ********** >> 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 4

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)

Correct Answer: D

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

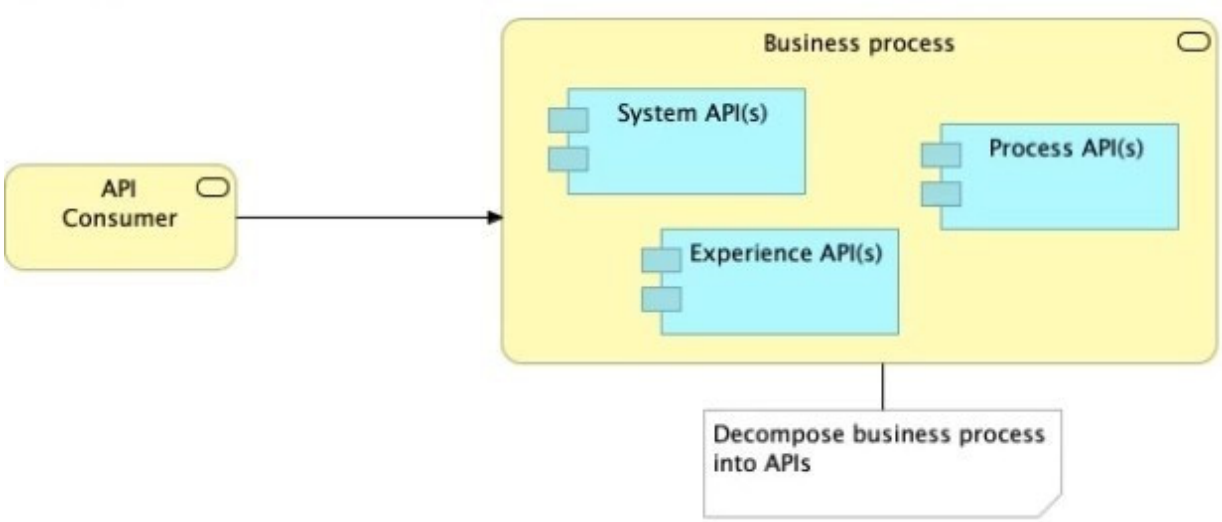
and make them success or fail to easily simulate happy and unhappy paths respectively.

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.

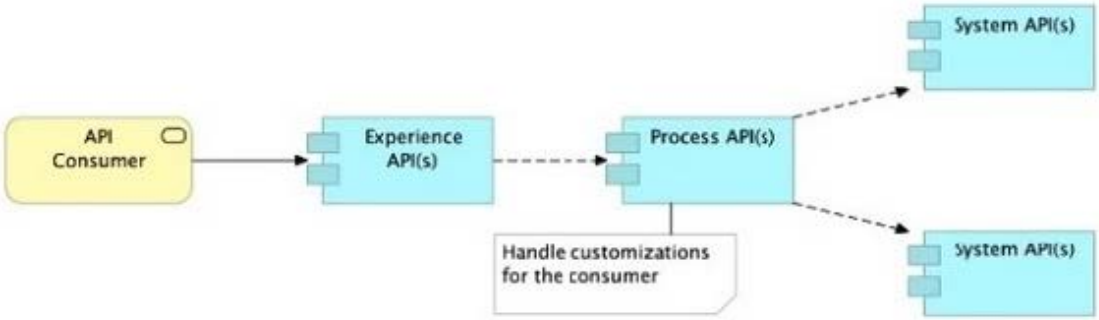
QUESTION 5

Refer to the exhibit.

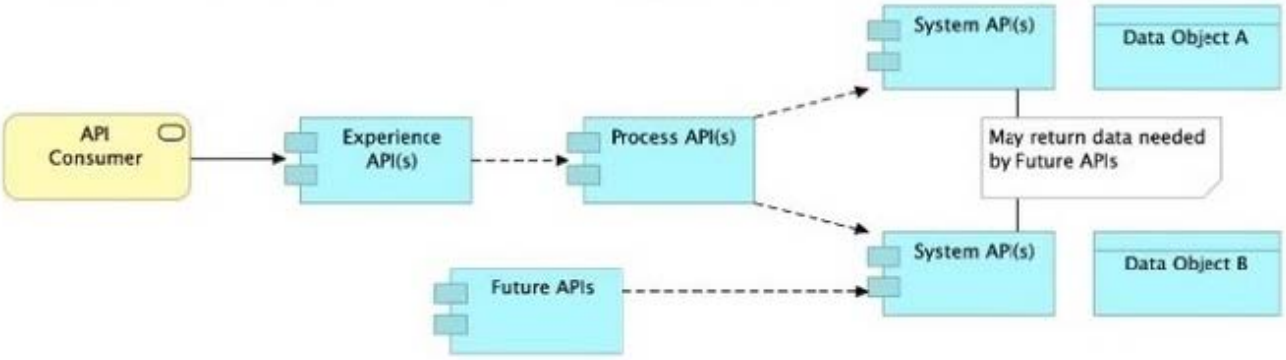


What is the best way to decompose one end-to-end business process into a collaboration of Experience, Process, and System APIs?

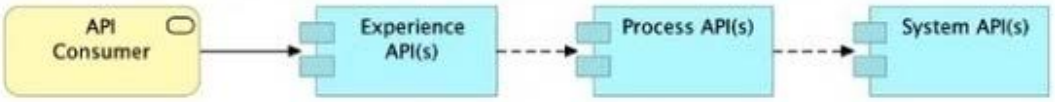
A. Handle customizations for the end-user application at the Process API level rather than the Experience API level



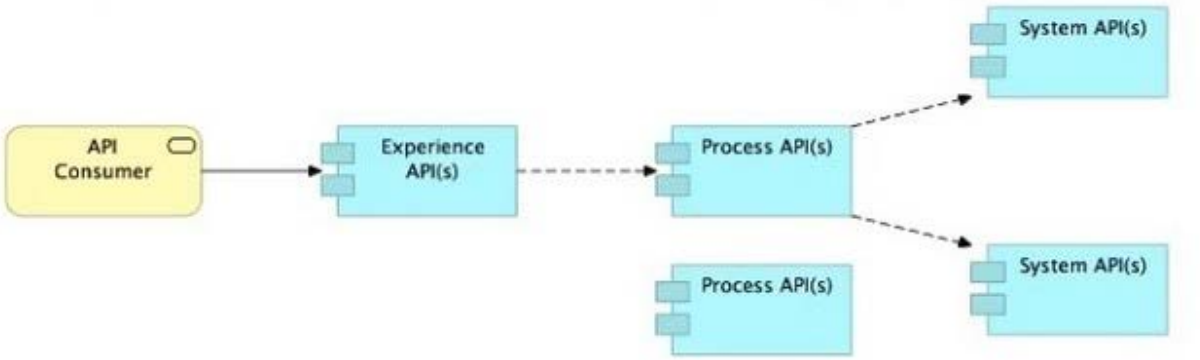
B. Allow System APIs to return data that is NOT currently required by the identified Process or Experience APIs



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

Correct Answer: B

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

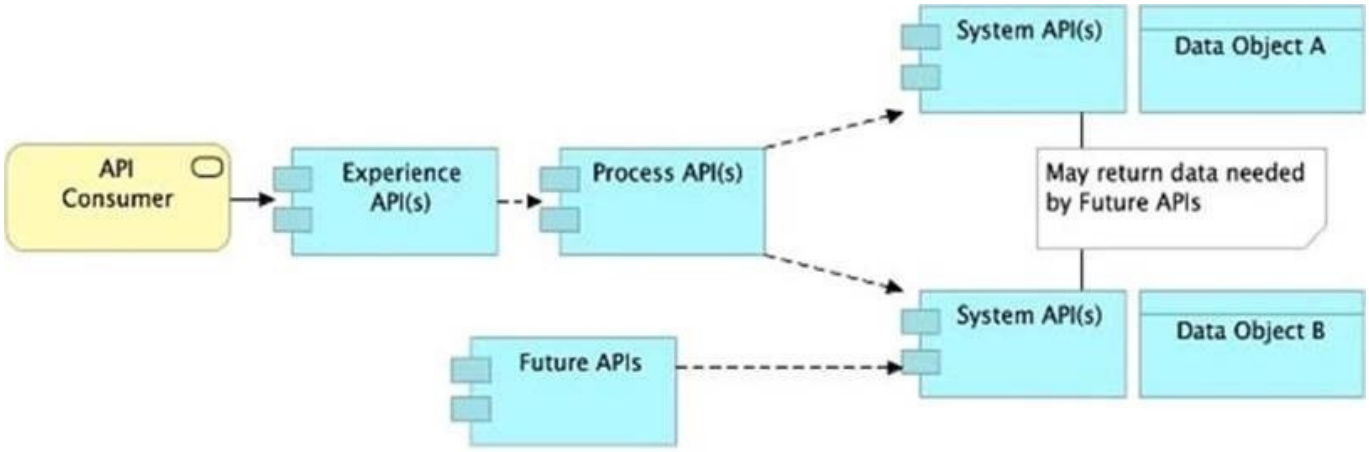
>> We should use tiered approach but NOT always by creating exactly one API for each of the 3 layers. Experience APIs might be one but Process APIs and System APIs are often more than one. System APIs for sure will be more than one

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.



[MCPA-LEVEL1 VCE Dumps](#) [MCPA-LEVEL1 Practice Test](#) [MCPA-LEVEL1 Study Guide](#)