DUMPSQUEEN

MuleSoft Certified Platform Architect - Level 1 MAINTENANCE

Mulesoft MCPA-Level-1-Maintenance

Version Demo

Total Demo Questions: 10

Total Premium Questions: 80

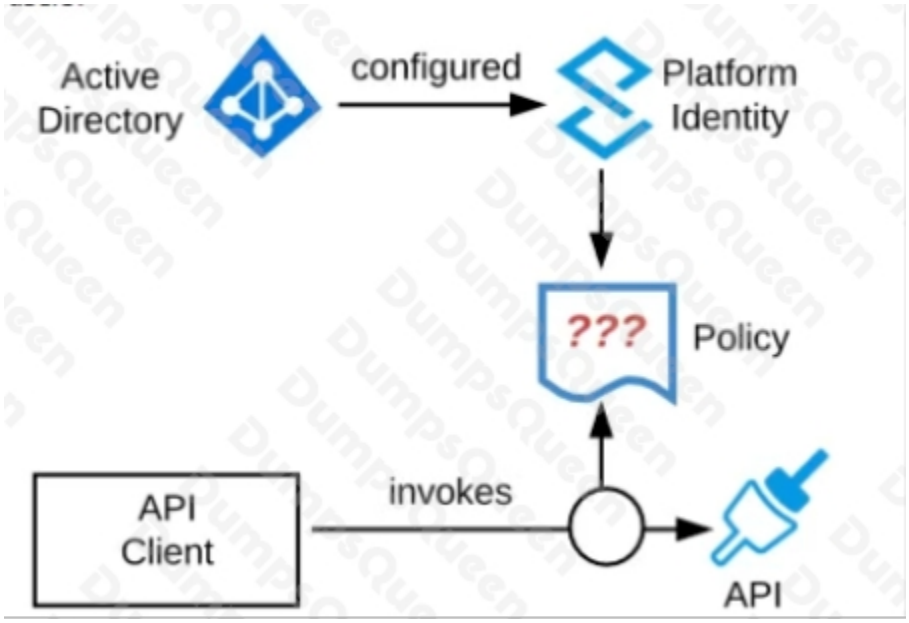
Buy Premium PDF

https://dumpsqueen.com

support@dumpsqueen.com

QUESTION NO: 1

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



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

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

B. Apply a client ID enforcement policy; the specific group of users will configure their client applications to use their specific client credentials

C. Apply an IP whitelist policy; only the specific users' workstations will be in the whitelist

D. Apply an OAuth 2.0 access token enforcement policy; the internal Active Directory will be configured as the OAuth server

ANSWER: A

Explanation:

Explanation

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

****

>> IP Whitelisting does NOT fit for this purpose. Moreover, the users workstations may not necessarily have static IPs in the network.

>> OAuth 2.0 enforcement requires a client provider which isn't in the organizations system components.

>> It is not an effective approach to let every user create separate client credentials and configure those for their usage.

The effective way it to apply a basic authentication - LDAP policy and the internal Active Directory will be configured as the LDAP source for authenticating users.

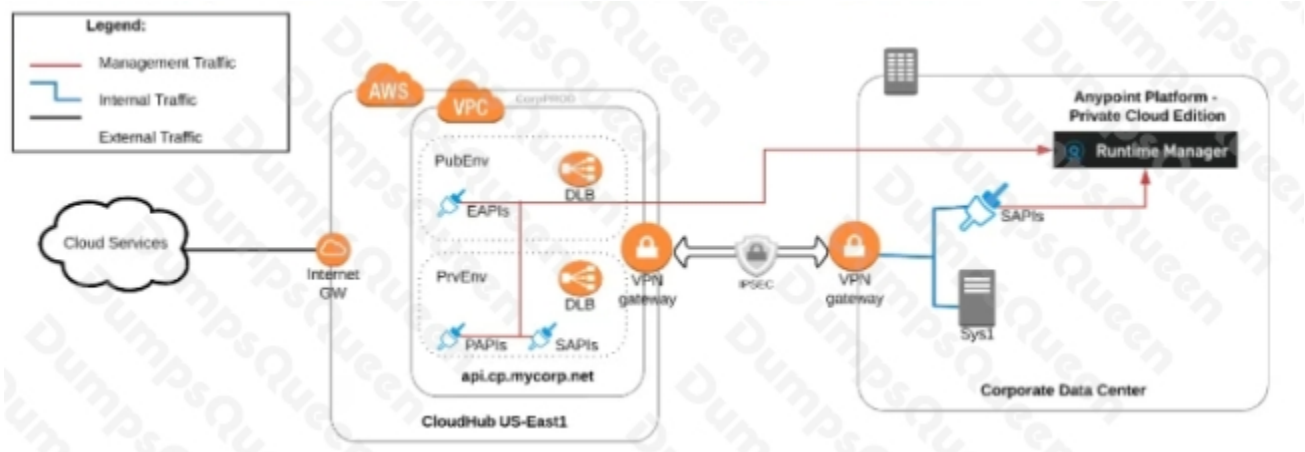
Reference: https://docs.mulesoft.com/api-manager/2.x/basic-authentication-ldap-concept

QUESTION NO: 2

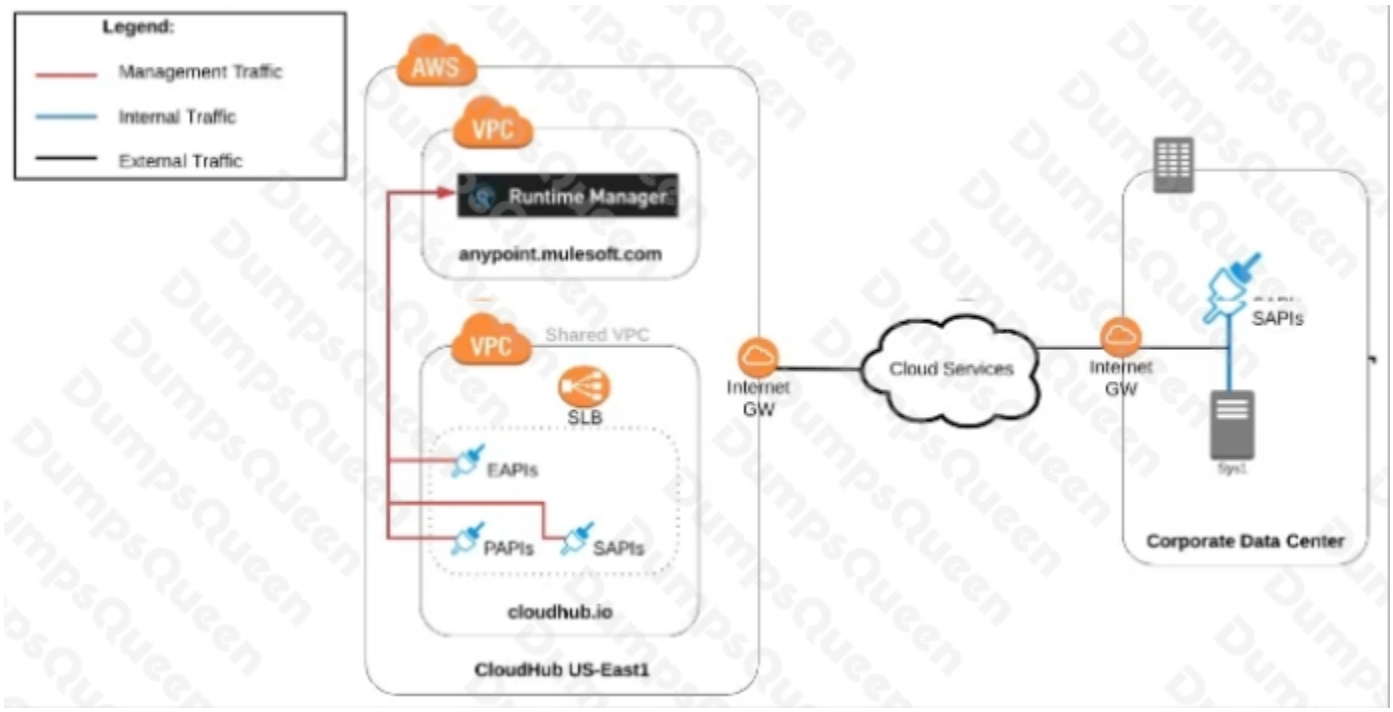
An organization uses various cloud-based SaaS systems and multiple on-premises systems. The on-premises systems are an important part of the organization's application network and can only be accessed from within the organization's intranet.

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?

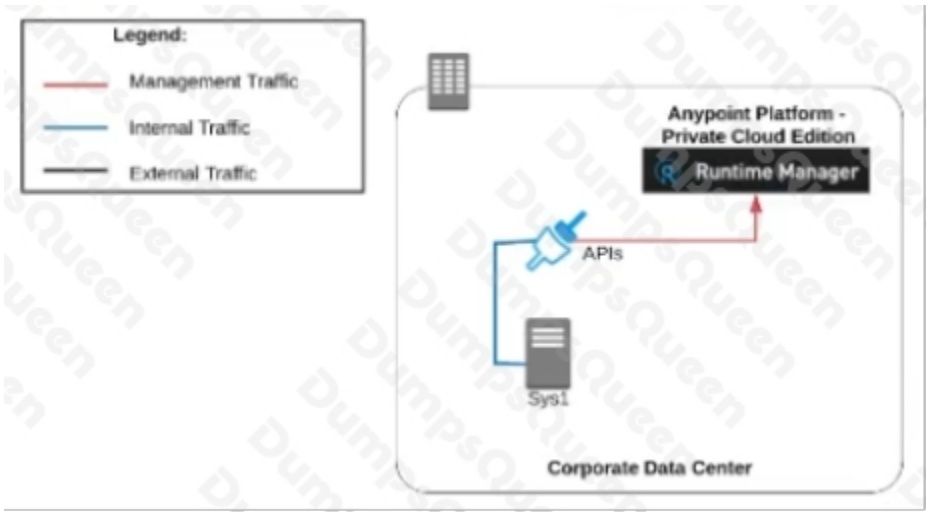
A) Use CloudHub-deployed Mule runtimes in an Anypoint VPC managed by Anypoint Platform Private Cloud Edition control plane



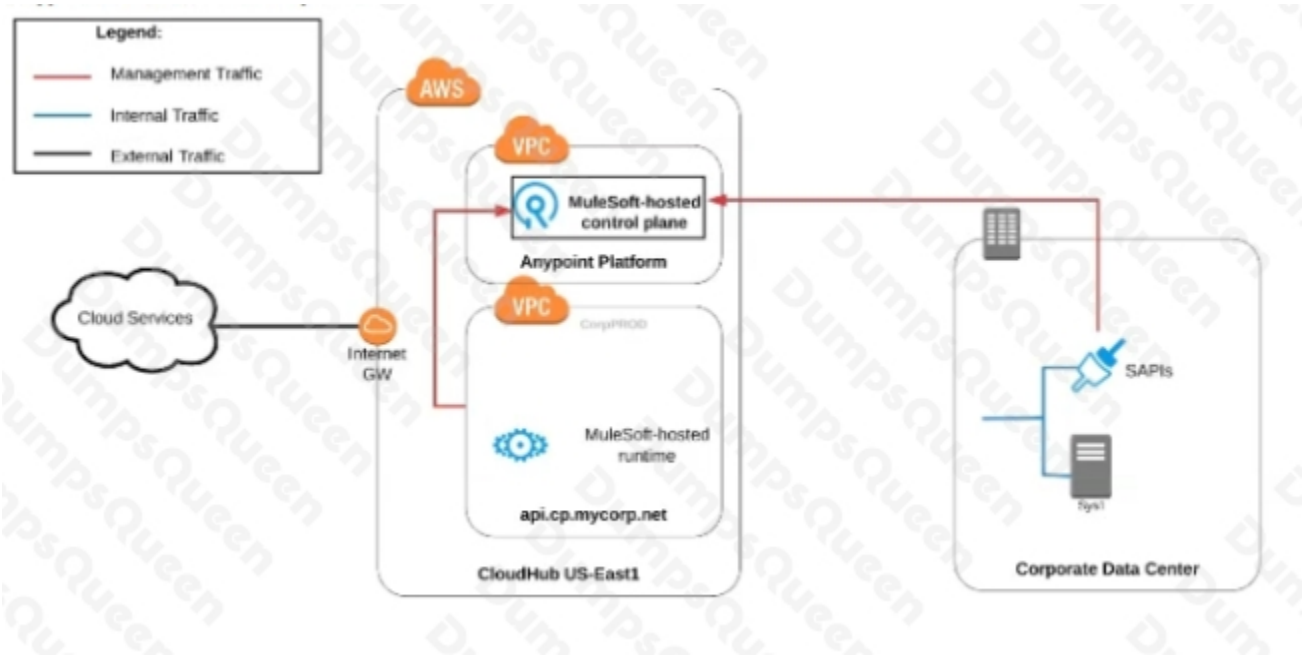
B) Use CloudHub-deployed Mule runtimes in the shared worker cloud managed by the MuleSoft-hosted Anypoint Platform control plane



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

ANSWER: B

Explanation:

Explanation

(Correct Answer): Use a combination of CloudHub-deployed and manually provisioned on-premises Mule runtimes managed by the MuleSoft-hosted Platform control plane.

****

Key details to be taken from the given scenario:

>> Organization uses BOTH cloud-based and on-premises systems

>> On-premises systems can only be accessed from within the organization's intranet

Let us evaluate the given choices based on above key details:

>> CloudHub-deployed Mule runtimes can ONLY be controlled using MuleSoft-hosted control plane. We CANNOT use Private Cloud Edition's control plane to control CloudHub Mule Runtimes. So, option suggesting this is INVALID

>> Using CloudHub-deployed Mule runtimes in the shared worker cloud managed by the MuleSoft-hosted Anypoint Platform is completely IRRELEVANT to given scenario and silly choice. So, option suggesting this is INVALID

>> Using an on-premises installation of Mule runtimes that are completely isolated with NO external network access, managed by the Anypoint Platform Private Cloud Edition control plane would work for On-premises integrations. However,

with NO external access, integrations cannot be done to SaaS-based apps. Moreover CloudHub-hosted apps are best-fit for integrating with SaaS-based applications. So, option suggesting this is BEST WAY.

The best way to configure and use Anypoint Platform to support these mixed/hybrid integrations is to use a combination of CloudHub-deployed and manually provisioned on-premises Mule runtimes managed by the MuleSoft-hosted Platform control plane.

QUESTION NO: 3

An API experiences a high rate of client requests (TPS) vwth small message paytoads. How can usage limits be imposed on the API based on the type of client application?

A. Use an SLA-based rate limiting policy and assign a client application to a matching SLA tier based on its type

B. Use a spike control policy that limits the number of requests for each client application type

C. Use a cross-origin resource sharing (CORS) policy to limit resource sharing between client applications, configured by the client application type

D. Use a rate limiting policy and a client ID enforcement policy, each configured by the client application type

ANSWER: A

Explanation:

(Correct Answer): Use an SLA-based rate limiting policy and assign a client application to a matching SLA tier based on its type.

****

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

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

QUESTION NO: 4

Which of the following sequence is correct?

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

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

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

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

ANSWER: B

Explanation:

Explanation

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

****

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

>> API Implementation does not route any requests. It is a final piece of logic where functionality of target systems is exposed. So, the requests should be routed to the API implementation by some other entity. So, the options stating "API Implementation routes the request to >> API" is INVALID

>> The statements in one of the options are correct but sequence is wrong. The sequence is given as "API Client implementes logic to call an API >> API Consumer requests access to API >> API routes the request to >> API Implementation". Here, the statements in the options are VALID but sequence is WRONG.

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

QUESTION NO: 5

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:

Explanation

(Correct Answer): The API policy is defined in API Manager for a specific API instance, and then ONLY applied to the specific API instance.

****

>> Once our API specifications are ready and published to Exchange, we need to visit API Manager and register an API instance for each API.

>> API Manager is the place where management of API aspects takes place like addressing NFRs by enforcing policies on them.

>> We can create multiple instances for a same API and manage them differently for different purposes.

>> One instance can have a set of API policies applied and another instance of same API can have different set of policies applied for some other purpose.

>> These APIs and their instances are defined PER environment basis. So, one need to manage them seperately in each environment.

>> We can ensure that same configuration of API instances (SLAs, Policies etc..) gets promoted when promoting to higher environments using platform feature. But this is optional only. Still one can change them per environment basis if they have to.

>> Runtime Manager is the place to manage API Implementations and their Mule Runtimes but NOT APIs itself. Though API policies gets executed in Mule Runtimes, We CANNOT enforce API policies in Runtime Manager. We would need to do that via API Manager only for a cherry picked instance in an environment.

So, based on these facts, right statement in the given choices is - "The API policy is defined in API Manager for a specific API instance, and then ONLY applied to the specific API instance".

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

QUESTION NO: 6

What is most likely NOT a characteristic of an integration test for a REST API implementation?

A. The test needs all source and/or target systems configured and accessible

B. The test runs immediately after the Mule application has been compiled and packaged

C. The test is triggered by an external HTTP request

D. The test prepares a known request payload and validates the response payload

ANSWER: B

Explanation:

Explanation

(Correct Answer): The test runs immediately after the Mule application has been compiled and packaged

****

>> Integration tests are the last layer of tests we need to add to be fully covered.

>> These tests actually run against Mule running with your full configuration in place and are tested from external source as they work in PROD.

>> These tests exercise the application as a whole with actual transports enabled. So, external systems are affected when these tests run.

So, these tests do NOT run immediately after the Mule application has been compiled and packaged.

FYI... Unit Tests are the one that run immediately after the Mule application has been compiled and packaged.

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

QUESTION NO: 7

What CANNOT be effectively enforced using an API policy in Anypoint Platform?

A. Guarding against Denial of Service attacks

B. Maintaining tamper-proof credentials between APIs

C. Logging HTTP requests and responses

D. Backend system overloading

ANSWER: A

Explanation:

Explanation

(Correct Answer): Guarding against Denial of Service attacks

****

>> Backend system overloading can be handled by enforcing "Spike Control Policy"

>> Logging HTTP requests and responses can be done by enforcing "Message Logging Policy"

>> Credentials can be tamper-proofed using "Security" and "Compliance" Policies

However, unfortunately, there is no proper way currently on Anypoint Platform to guard against DOS attacks.

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

QUESTION NO: 8

Select the correct Owner-Layer combinations from below options

A. 1. App Developers owns and focuses on Experience Layer APIs
2. Central IT owns and focuses on Process Layer APIs
3. LOB IT owns and focuses on System Layer APIs

B. 1. Central IT owns and focuses on Experience Layer APIs
2. LOB IT owns and focuses on Process Layer APIs
3. App Developers owns and focuses on System Layer APIs

C. 1. App Developers owns and focuses on Experience Layer APIs
2. LOB IT owns and focuses on Process Layer APIs
3. Central IT owns and focuses on System Layer APIs

ANSWER: C

Explanation:

Explanation

(Correct Answer):

1. App Developers owns and focuses on Experience Layer APIs

2. LOB IT owns and focuses on Process Layer APIs

3. Central IT owns and focuses on System Layer APIs

References:

https://blogs.mulesoft.com/biz/api/experience-api-ownership/

https://blogs.mulesoft.com/biz/api/process-api-ownership/

https://blogs.mulesoft.com/biz/api/system-api-ownership/

QUESTION NO: 9

What correctly characterizes unit tests of Mule applications?

A. They test the validity of input and output of source and target systems

B. They must be run in a unit testing environment with dedicated Mule runtimes for the environment

C. They must be triggered by an external client tool or event source

D. They are typically written using MUnit to run in an embedded Mule runtime that does not require external connectivity

ANSWER: D

Explanation:

Explanation

(Correct Answer): They are typically written using MUnit to run in an embedded Mule runtime that does not require external connectivity.

****

Below TWO are characteristics of Integration Tests but NOT unit tests:

>> They test the validity of input and output of source and target systems.

>> They must be triggered by an external client tool or event source.

It is NOT TRUE that Unit Tests must be run in a unit testing environment with dedicated Mule runtimes for the environment.

MuleSoft offers MUnit for writing Unit Tests and they run in an embedded Mule Runtime without needing any separate/ dedicated Runtimes to execute them. They also do NOT need any external connectivity as MUnit supports mocking via stubs.

https://dzone.com/articles/munit-framework

QUESTION NO: 10

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

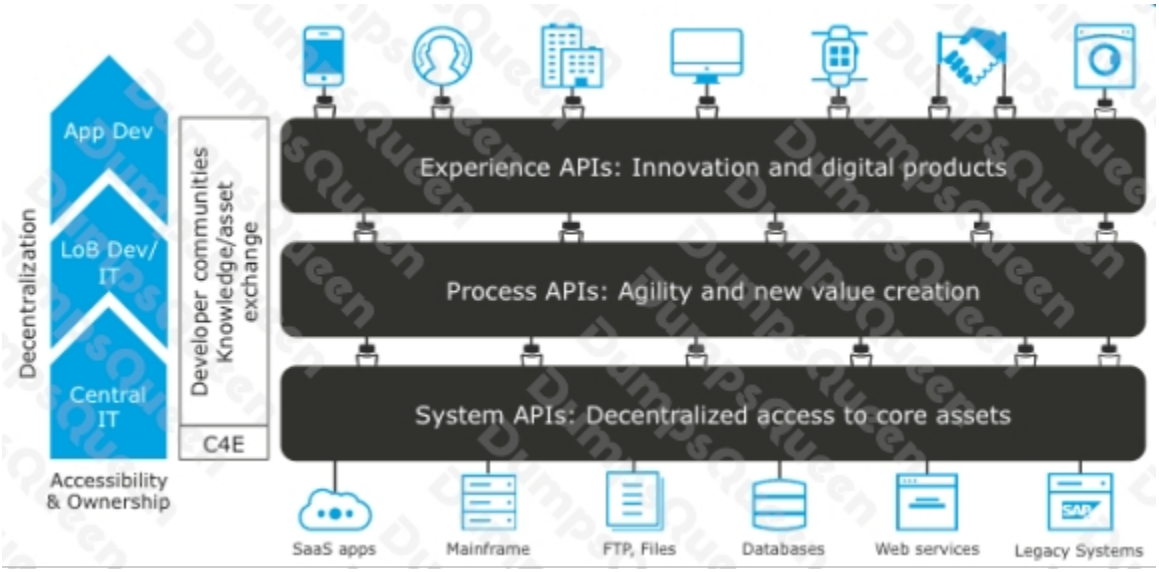
C. System Layer

ANSWER: C

Explanation:

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.