Vendor: Mulesoft

Exam Code:MCPA-LEVEL-1-MAINTENANCE

Exam Name: MuleSoft Certified Platform Architect - Level 1 MAINTENANCE

Version: Demo

QUESTION 1

When using CloudHub with the Shared Load Balancer, what is managed EXCLUSIVELY by the API implementation (the Mule application) and NOT by Anypoint Platform?

A. The assignment of each HTTP request to a particular CloudHub worker

B. The logging configuration that enables log entries to be visible in Runtime Manager

C. The SSL certificates used by the API implementation to expose HTTPS endpoints

D. The number of DNS entries allocated to the API implementation

Correct Answer: C

The SSL certificates used by the API implementation to expose HTTPS endpoints

****

>> The assignment of each HTTP request to a particular CloudHub worker is taken care by Anypoint Platform itself. We need not manage it explicitly in the API implementation and in fact we CANNOT manage it in the API implementation. >>

The logging configuration that enables log entries to be visible in Runtime Manager is ALWAYS managed in the API implementation and NOT just for SLB. So this is not something we do EXCLUSIVELY when using SLB.

>> We DO NOT manage the number of DNS entries allocated to the API implementation inside the code. Anypoint Platform takes care of this.

It is the SSL certificates used by the API implementation to expose HTTPS endpoints that is to be managed EXCLUSIVELY by the API implementation. Anypoint Platform does NOT do this when using SLBs.

QUESTION 2

Once an API Implementation is ready and the API is registered on API Manager, who should request the access to the API on Anypoint Exchange?

A. None

B. Both

C. API Client

D. API Consumer

Correct Answer: D

API Consumer ********** >> API clients are piece of code or programs that use the client credentials of API consumer but does not directly interact with Anypoint Exchange to get the access >> API consumer is

the one who should get registered and request access to API and then API client needs to use those client credentials to hit the APIs So, API consumer is the one who needs to request access on the API from Anypoint Exchange

---

QUESTION 3

A company requires Mule applications deployed to CloudHub to be isolated between non- production and production environments. This is so Mule applications deployed to non- production environments can only access backend systems running in their customer- hosted non-production environment, and so Mule applications deployed to production environments can only access backend systems running in their customer-hosted production environment. How does MuleSoft recommend modifying Mule applications, configuring environments, or changing infrastructure to support this type of per- environment isolation between Mule applications and backend systems?

A. Modify properties of Mule applications deployed to the production Anypoint Platform environments to prevent access from non-production Mule applications

B. Configure firewall rules in the infrastructure inside each customer-hosted environment so that only IP addresses from the corresponding Anypoint Platform environments are allowed to communicate with corresponding backend systems

C. Create non-production and production environments in different Anypoint Platform business groups

D. Create separate Anypoint VPCs for non-production and production environments, then configure connections to the backend systems in the corresponding customer-hosted environments

Correct Answer: D

Create separate Anypoint VPCs for non-production and production environments, then configure connections to the backend systems in the corresponding customer-hosted environments. ********** >> Creating different Business Groups does NOT make any difference w.r.t accessing the non-prod and prod customer-hosted environments. Still they will be accessing from both Business Groups unless process network restrictions are put in place. >> We need to modify or couple the Mule Application Implementations with the environment. In fact, we should never implements application coupled with environments by binding them in the properties. Only basic things like endpoint URL etc should be bundled in properties but not environment level access restrictions. >> IP addresses on CloudHub are dynamic until unless a special static addresses are assigned. So it is not possible to setup firewall rules in customer-hosted infrastrcture. More over, even if static IP addresses are assigned, there could be 100s of applications running on cloudhub and setting up rules for all of them would be a hectic task, non-maintainable and definitely got a good practice. >> The best practice recommended by Mulesoft (In fact any cloud provider), is to have your Anypoint VPCs seperated for Prod and Non-Prod and perform the VPC peering or VPN tunneling for these Anypoint VPCs to respective Prod and Non-Prod customer-hosted environment networks. : https://docs.mulesoft.com/runtime-manager/virtual-private-cloud

---

QUESTION 4

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

Correct Answer: A

QUESTION 5

A system API has a guaranteed SLA of 100 ms per request. The system API is deployed to a primary environment as well as to a disaster recovery (DR) environment, with different DNS names in each environment. An upstream process API invokes the system API and the main goal of this process API is to respond to client requests in the least possible time. In what order should the system APIs be invoked, and what changes should be made in order to speed up the response time for requests from the process API?

A. In parallel, invoke the system API deployed to the primary environment and the system API deployed to the DR environment, and ONLY use the first response

B. In parallel, invoke the system API deployed to the primary environment and the system API deployed to the DR environment using a scatter-gather configured with a timeout, and then merge the responses

C. Invoke the system API deployed to the primary environment, and if it fails, invoke the system API deployed to the DR environment

D. Invoke ONLY the system API deployed to the primary environment, and add timeout and retry logic to avoid intermittent failures

Correct Answer: A

In parallel, invoke the system API deployed to the primary environment and the system API deployed to the DR environment, and ONLY use the first response.

****

>> The API requirement in the given scenario is to respond in least possible time. >> The option that is suggesting to first try the API in primary environment and then fallback to API in DR environment would result in successful response but

NOT in least possible time. So, this is NOT a right choice of implementation for given requirement. >> Another option that is suggesting to ONLY invoke API in primary environment and to add timeout and retries may also result in successful

response upon retries but NOT in least possible time. So, this is also NOT a right choice of implementation for given requirement.

>> One more option that is suggesting to invoke API in primary environment and API in DR environment in parallel

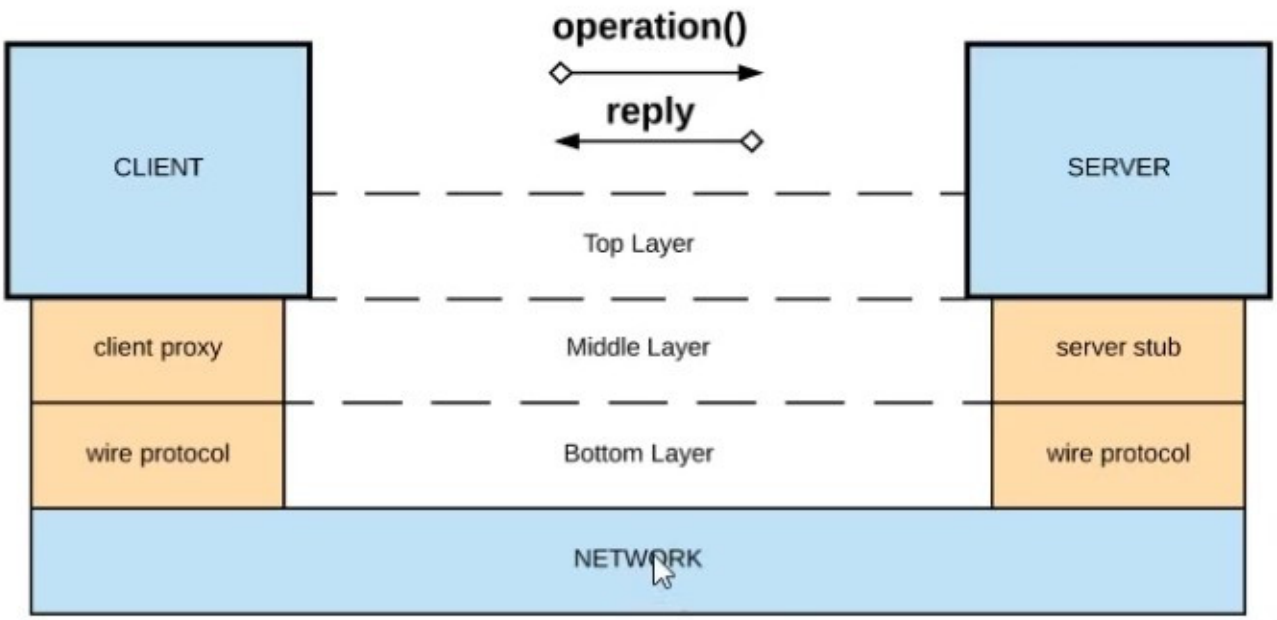
using Scatter-Gather would result in wrong API response as it would return merged results and moreover, Scatter-Gather

does things in parallel which is true but still completes its scope only on finishing all routes inside it. So again, NOT a right choice of implementation for given requirement

The Correct choice is to invoke the API in primary environment and the API in DR environment parallelly, and using ONLY the first response received from one of them.

QUESTION 6

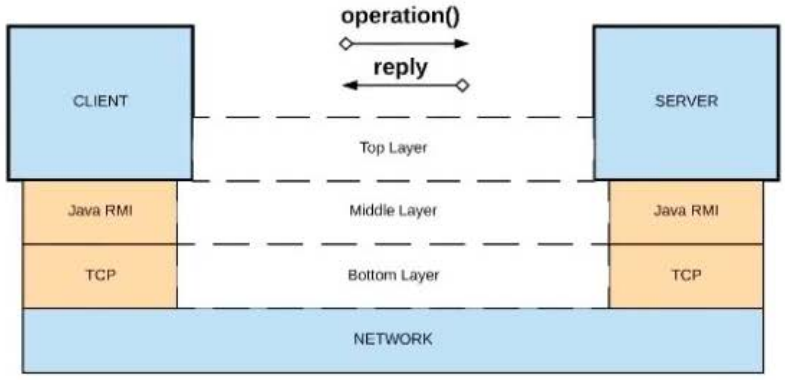
Refer to the exhibit.



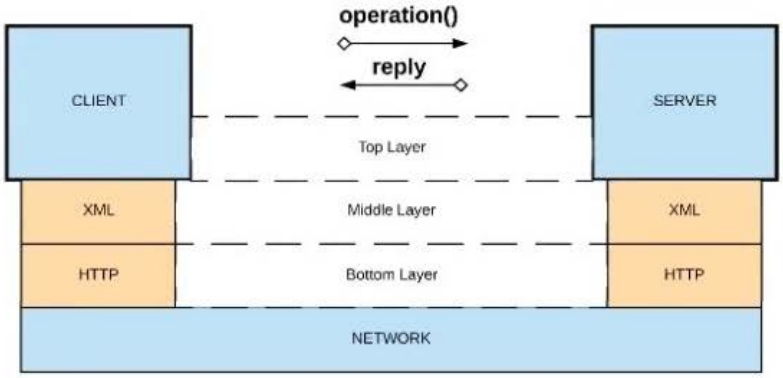
Generic RPC Architecture

What is a valid API in the sense of API-led connectivity and application networks?

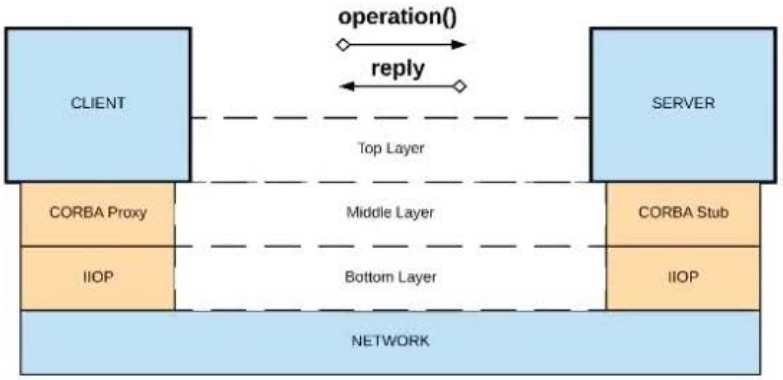
A. Java RMI over TCP



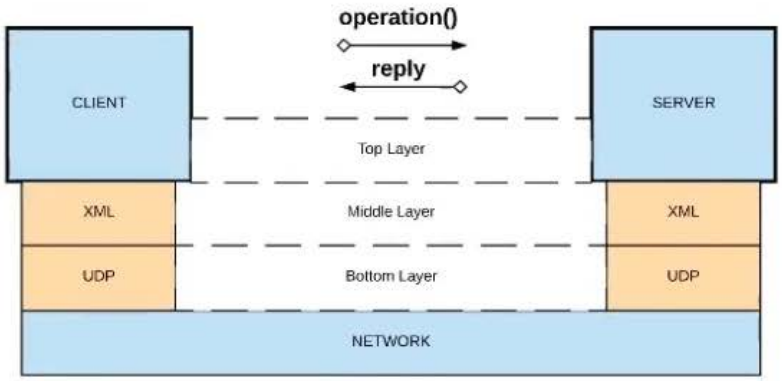
B. Java RMI over TCP



C. CORBA over HOP



D. XML over UDP



A. Option A

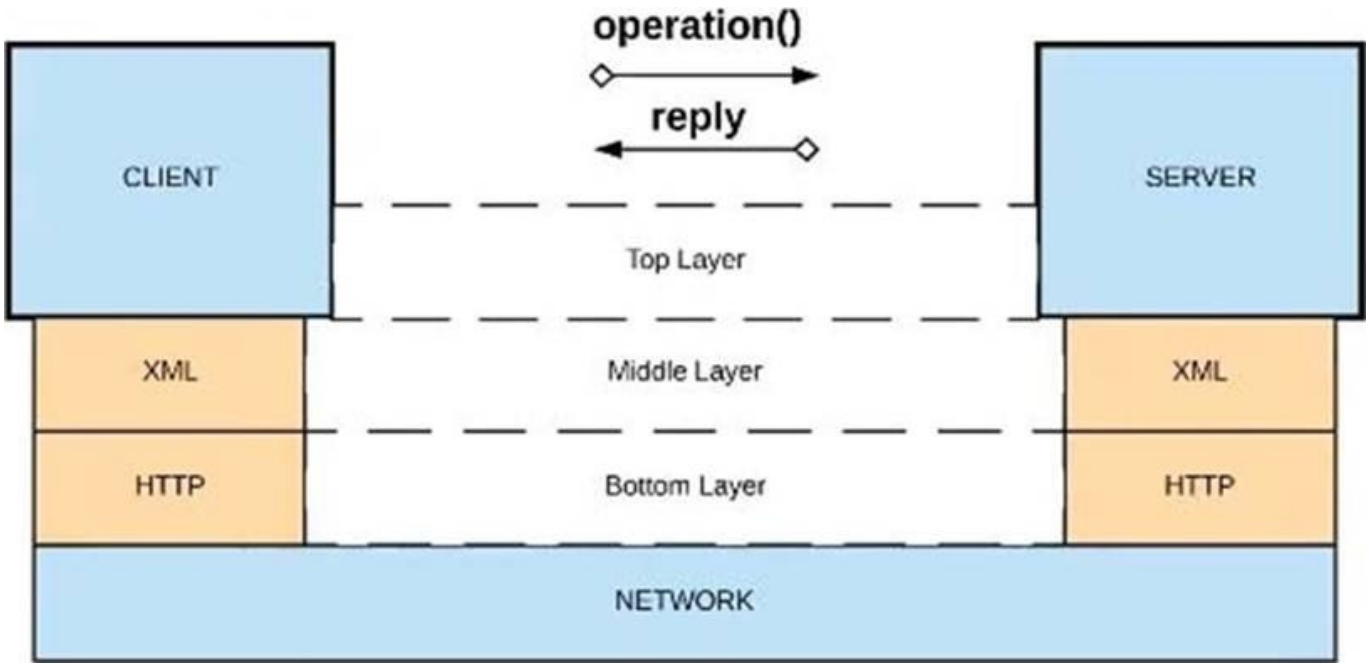
B. Option B

C. Option C

D. Option D

Correct Answer: D

XML over HTTP ** >> API-led connectivity and Application Networks urge to have the APIs on HTTP based protocols for building most effective APIs and networks on top of them. >> The HTTP based APIs allow the platform to apply various varities of policies to address many NFRs >> The HTTP based APIs also allow to implement many standard and effective implementation patterns that adhere to HTTP based w3c rules.



QUESTION 7

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

Correct Answer: B

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.

QUESTION 8

What is the main change to the IT operating model that MuleSoft recommends to organizations to improve innovation and clock speed?

A. Drive consumption as much as production of assets; this enables developers to discover and reuse assets from other projects and encourages standardization

B. Expose assets using a Master Data Management (MDM) system; this standardizes projects and enables developers to quickly discover and reuse assets from other projects

C. Implement SOA for reusable APIs to focus on production over consumption; this standardizes on XML and WSDL formats to speed up decision making

D. Create a lean and agile organization that makes many small decisions everyday; this speeds up decision making and enables each line of business to take ownership of its projects

Correct Answer: A

Drive consumption as much as production of assets; this enables developers to discover and reuse assets from other projects and encourages standardization

****

>> The main motto of the new IT Operating Model that MuleSoft recommends and made popular is to change the way that they are delivered from a production model to a production + consumption model, which is done through an API

strategy called API-led connectivity.

>> The assets built should also be discoverable and self-serveable for reusablity across LOBs and organization.

>> MuleSoft\\'s IT operating model does not talk about SDLC model (Agile/ Lean etc) or MDM at all. So, options suggesting these are not valid.

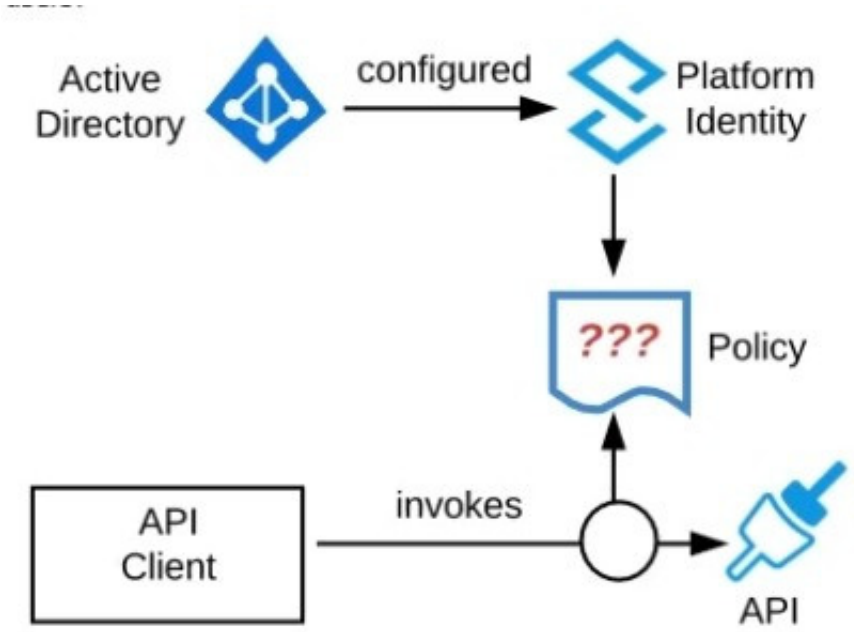
References:

https://blogs.mulesoft.com/biz/connectivity/what-is-a-center-for-enablement-c4e/

https://www.mulesoft.com/resources/api/secret-to-managing-it-projects

QUESTION 9

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

Correct Answer: A

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 10

A code-centric API documentation environment should allow API consumers to investigate and execute API client source code that demonstrates invoking one or more APIs as part of representative scenarios.

What is the most effective way to provide this type of code-centric API documentation environment using Anypoint Platform?

A. Enable mocking services for each of the relevant APIs and expose them via their Anypoint Exchange entry

B. Ensure the APIs are well documented through their Anypoint Exchange entries and API Consoles and share these pages with all API consumers

C. Create API Notebooks and include them in the relevant Anypoint Exchange entries

D. Make relevant APIs discoverable via an Anypoint Exchange entry

Correct Answer: C

Create API Notebooks and Include them in the relevant Anypoint exchange entries ********** >> API Notebooks are the one on Anypoint Platform that enable us to provide code-centric API documentation : https://docs.mulesoft.com/exchange/to-use-api-notebook

---

QUESTION 11

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

Correct Answer: A

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 12

Which layer in the API-led connectivity focuses on unlocking key systems, legacy systems, data sources etc and exposes the functionality?

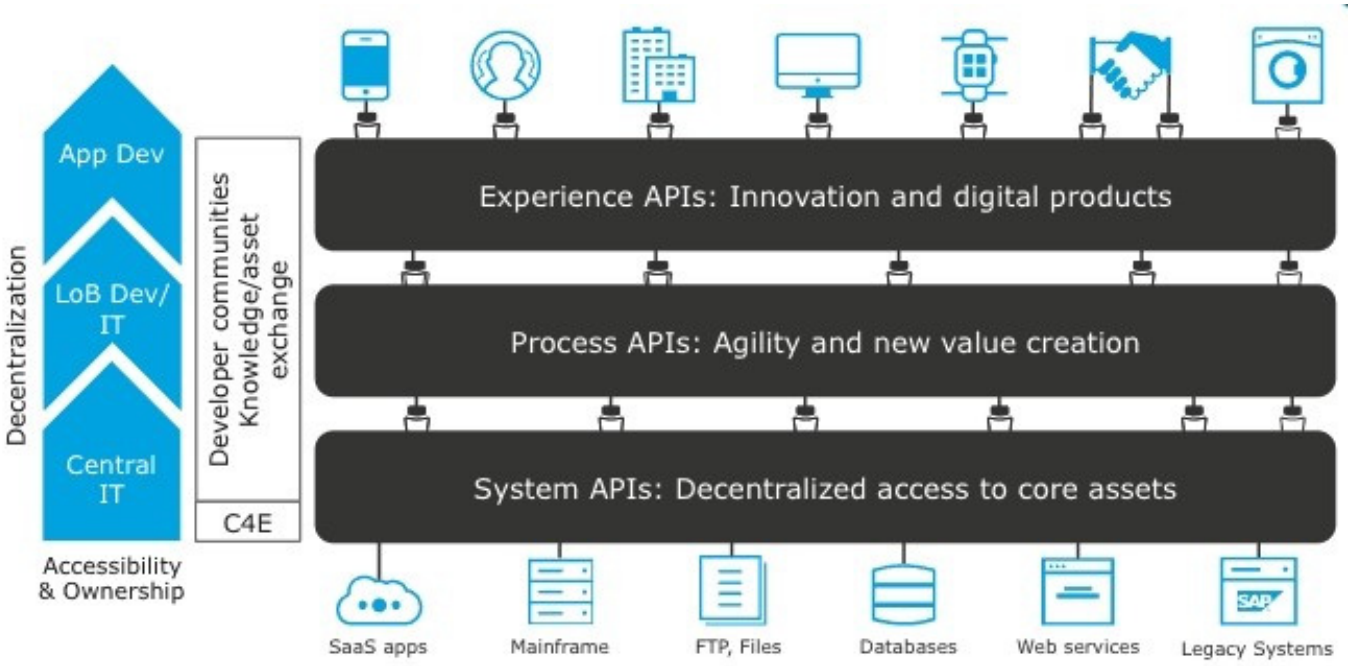
A. Experience Layer

B. Process Layer

C. System Layer

Correct Answer: C

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.