DUMPSBOSS.

MuleSoft Certified Platform Architect - Level 1

Mulesoft MCPA-Level-1

Version Demo

Total Demo Questions: 13

Total Premium Questions: 194

Buy Premium PDF

https://dumpsboss.co

support@dumpsboss.co

support@dumpsboss.co
dumpsboss.co

Topic Break Down

Topic	No. of Questions
Topic 1, Application Network	46
Topic 2, Architecture	56
Topic 3, Anypoint Platform	36
Topic 4, Operations	27
Topic 5, Security	29
Total	194

QUESTION NO: 1

A customer wants to monitor and gain insights about the number of requests coming in a given time period as well as to measure key performance indicators

(response times, CPU utilization, number of active APIs).

Which tool provides these data insights?

A. Anypoint Monitoring

B. APT Manager

C. Runtime Alerts

D. Functional Monitoring

ANSWER: A

Explanation:

Anypoint Monitoring is the appropriate tool because it provides operational visibility into Mule applications and their runtime behavior over selected time ranges. Its dashboards and metrics enable teams to examine incoming request volume, response-time trends, CPU utilization, memory use, error rates, and other runtime indicators needed to assess API and application health. This lets architects and operations teams identify demand patterns, establish performance baselines, investigate degradation, and make capacity or scaling decisions using observed data rather than isolated events.

For Mule applications deployed to supported runtime targets, Anypoint Monitoring collects and presents metrics in charts and dashboards, with filtering by environment, application, and time period. It also supports more detailed analysis through logs, tracing, and customizable dashboards, so the organization can correlate performance behavior with request activity. These capabilities directly satisfy the requirement for historical and near-real-time KPI insight rather than merely managing an API or issuing a single threshold notification. MuleSoft documents the available monitoring dashboards and application metrics in the Anypoint Monitoring documentation, including guidance on using application dashboards to inspect runtime performance and request-related metrics.

QUESTION NO: 2

A company has started to create an application network and is now planning to implement a Center for Enablement (C4E) organizational model. What key factor would lead the company to decide upon a federated rather than a centralized C4E?

A. When there are a large number of existing common assets shared by development teams

B. When various teams responsible for creating APIs are new to integration and hence need extensive training

C. When development is already organized into several independent initiatives or groups

D. When the majority of the applications in the application network are cloud based

ANSWER: C

Explanation:

When development is already organized into several independent initiatives or groups, a federated Center for Enablement is appropriate because it distributes enablement responsibility across those established teams while maintaining shared standards and governance. In a federated model, individual business domains, product groups, or delivery organizations can retain the autonomy needed to deliver APIs and integrations at their own pace. At the same time, the C4E provides common practices, reusable assets, platform guidance, and guardrails that keep the broader application network aligned. This structure is particularly effective in organizations that are already decentralized, since forcing all integration delivery through one central team can create bottlenecks and reduce ownership within the teams closest to the business capabilities they expose. MuleSoft describes the C4E as an operating model that enables teams to deliver integration capabilities at scale, with the model adapted to an organization's structure and maturity. A federated implementation supports distributed execution while preserving consistency through central enablement and governance. See MuleSoft's Center for Enablement overview and its API-led connectivity resources.

QUESTION NO: 3

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:

Guarding against Denial of Service attacks is the correct answer because an API policy runs at the Mule gateway only after traffic has already reached the runtime and consumed some network, connection, and processing capacity. Policies such as rate limiting and spike control are valuable for controlling the rate at which accepted requests are processed and for protecting downstream services from excessive demand. However, they are not a complete or effective defense against a true denial-of-service or distributed denial-of-service attack, where an attacker can exhaust bandwidth, connections, or gateway resources before policy evaluation can provide meaningful protection.

Anypoint API policies should therefore be part of a layered security and traffic-management design, rather than the perimeter defense for volumetric attacks. Effective DoS protection typically requires controls positioned upstream of Mule runtimes, such as an edge network, load balancer, web application firewall, DDoS-protection service, and network-level filtering. MuleSoft documents that rate-limiting policies control request consumption and can reject requests after configured limits are exceeded, which helps preserve backend capacity but does not eliminate inbound attack traffic. See MuleSoft Rate Limiting policy documentation and MuleSoft API policy documentation.

QUESTION NO: 4

A client has several applications running on the Salesforce service cloud. The business requirement for integration is to get daily data changes from Account and Case

Objects. Data needs to be moved to the client's private cloud AWS DynamoDB instance as a single JSON and the business foresees only wanting five attributes from the

Account object, which has 219 attributes (some custom) and eight attributes from the Case Object.

What design should be used to support the API/ Application data model?

A. Create separate entities for Account and Case Objects by mimicking all the attributes in SAPI, which are combined by the PAPI and filtered to provide JSON output containing 13 attributes.

B. transfer ISON data to DynamoD for respective Objects

C. Start implementing an Enterprise Data Model by defining enterprise Account and Case Objects and implement SAPI and DynamoDB tables based on the Enterprise Data Model,

D. Create separate entities for Account with five attributes and Case with eight attributes in SAPI, which are combined by the PAPI to provide JSON output containing 13 attributes.

ANSWER: D

Explanation:

Create separate entities for Account with five attributes and Case with eight attributes in SAPI, which are combined by the PAPI to provide JSON output containing 13 attributes. This design applies API-led connectivity by keeping the system-facing layer responsible for access to Salesforce Account and Case data while exposing only the fields required for the identified integration use case. The process layer then owns the cross-entity orchestration and composition: it obtains the daily

changes, combines the selected Account and Case attributes, and produces one JSON document with the target-oriented structure required by DynamoDB.

Using focused entity representations prevents an unnecessarily large payload and avoids coupling the downstream data store to the full Salesforce Account schema, including fields that have no business value for this flow. The composed JSON is also a clear application-level contract: DynamoDB receives a stable representation designed for its consumption, while Salesforce-specific retrieval and change-handling details remain behind the APIs. This separation supports independent evolution of Salesforce fields, the process logic, and the target persistence implementation. MuleSoft describes this division of responsibilities through its API-led approach, in which system APIs unlock underlying systems and process APIs compose and transform data for business processes. See MuleSoft API-led connectivity documentation and MuleSoft DataWeave documentation.

QUESTION NO: 5

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.

ANSWER: A

Explanation:

Implement (or generate) an API proxy Mule application for each of the APIs, then deploy the API proxies to the Mule runtimes. This creates a controlled partner-facing entry point in Partner-subnet while leaving each existing API implementation in its current, protected subnet. The proxy receives requests from authorized external partners and forwards them over the already available network path to the internal API endpoint. Consequently, external traffic does not require direct inbound access to the implementation subnet.

An API proxy is purpose-built for this pattern: it separates API exposure and policy enforcement from the backend implementation. The proxy can be associated with the API in API Manager and enforce applicable policies, such as client authentication, rate limiting, IP allowlists, and transport security, before forwarding requests. This enables the organization to apply a consistent partner-access boundary without changing the deployed implementation applications or disrupting existing consumers that continue to call their current endpoints.

Because only lightweight proxy applications are added to Mule runtimes that already exist in Partner-subnet, this approach avoids moving or replicating the implementation APIs and minimizes new runtime and operational resources. MuleSoft documents API proxies as gateways that apply API Manager policies to requests before routing them to an upstream API implementation. See Mule Gateway documentation and Deploying API proxies in API Manager.

QUESTION NO: 6

An organization requires several APIs to be secured with OAuth 2.0, and PingFederate has been identified as the identity provider for API client authorization, The

PingFederate Client Provider is configured in access management, and the PingFederate OAuth 2.0 Token Enforcement policy is configured for the API instances required by the

organization. The API instances reside in two business groups (Group A and Group B) within the Master Organization (Master Org).

What should be done to allow API consumers to access the API instances?

A. The API administrator should configure the correct client discovery URL in both child business groups, and the API consumer should request access to the API in Ping Identity

B. The API administrator should grant access to the API consumers by creating contracts in the relevant API instances in API Manager

C. The API consumer should create a client application and request access to the API in Anypoint Exchange, and the API administrator should approve the request

D. The API consumer should create a client application and request access to the API in Ping Identity, and the organization's Ping Identity workflow will grant access

ANSWER: C

Explanation:

The API consumer should create a client application and request access to the API in Anypoint Exchange, and the API administrator should approve the request. This is the Anypoint Platform workflow for establishing a client application contract for a managed API instance. An API consumer discovers the API in Exchange, creates or selects a client application, and submits an access request for the relevant API instance. The API owner or administrator reviews that request and approves it, creating the contract that associates the client application with that API instance. This approval can be performed for API instances in the appropriate business groups, allowing the organization to govern access consistently while the PingFederate configuration supplies OAuth 2.0 authorization capabilities. After the contract is approved, the consumer can obtain and present the OAuth access token required by the PingFederate OAuth 2.0 Token Enforcement policy when calling the protected endpoint. MuleSoft documents the consumer request workflow in Request access to an API in Exchange and describes how API Manager manages client applications and contracts in Manage client applications. This separation lets Anypoint Platform govern API access while PingFederate remains the authorization server used to issue and validate OAuth tokens.

QUESTION NO: 7

Which two statements are true about the technology architecture of an Anypoint Virtual Private Cloud (VPC)?

Choose 2 answers

A. Ports 8081 and 8082 are used

B. CIDR blocks are used

C. Anypoint VPC is responsible for load balancing the applications

D. Round-robin load balancing is used to distribute client requests across different applications

E. By default, HTTP requests can be made from the public internet to workers at port 6091

ANSWER: B E

Explanation:

Anypoint Virtual Private Cloud (VPC) networking is defined using a Classless Inter-Domain Routing (CIDR) block. The CIDR block reserves the private IP-address range for the VPC and is a foundational design decision because it determines the addresses available to CloudHub resources and the ranges that can be routed through connectivity mechanisms such as VPNs or VPC peering. The selected CIDR range must therefore be planned to avoid overlap with the customer network and with other connected networks. MuleSoft documents the VPC CIDR block as part of VPC creation and network configuration; see Anypoint VPC documentation.

By default, HTTP requests from the public internet can reach CloudHub workers through port 6091. This default worker-access behavior is part of the VPC networking architecture and is relevant when defining perimeter controls, inbound firewall rules, and exposure requirements. Organizations can use supported networking and load-balancing capabilities to implement the desired public ingress pattern while keeping application traffic appropriately segmented. For the broader CloudHub networking model and inbound connectivity behavior, refer to the CloudHub networking guide.

QUESTION NO: 8

An organization wants to make sure only known partners can invoke the organization's APIs. To achieve this security goal, the organization wants to enforce a Client ID Enforcement policyin API Manager so that only registered partner applications can invoke the organization's APIs. In what type of API implementation does MuleSoft recommend adding an API proxy to enforce the Client ID Enforcement policy, rather than embedding the policy directly in the application's JVM?

A. A Mule 3 application using APIkit

B. A Mule 3 or Mule 4 application modified with custom Java code

C. A Mule 4 application with an API specification

D. A Non-Mule application

ANSWER: D

Explanation:

A Non-Mule application is correct because API Manager policies can be enforced in the runtime of a Mule application through API autodiscovery and embedded policy enforcement. In that deployment model, the Mule runtime retrieves the API configuration from API Manager and applies Client ID Enforcement before requests reach the implementation flow. This lets an organization validate the client application credentials registered in Anypoint Platform without placing equivalent security logic in each Mule flow or custom Java component.

A non-Mule implementation does not run inside a Mule runtime and therefore cannot use the Mule application's embedded policy-enforcement capability. MuleSoft recommends placing an API proxy in front of that implementation. The proxy runs on a Mule runtime, is associated with the managed API in API Manager, and applies Client ID Enforcement at the gateway boundary before forwarding an authorized request to the non-Mule backend. This preserves a consistent, centrally managed partner-access control model while allowing the backend application to remain unchanged. MuleSoft documents API proxies as the mechanism for applying API Manager policies to services not implemented in Mule, and documents Client ID Enforcement as a policy that validates client credentials on incoming requests. See MuleSoft API proxy deployments and Client ID-based policies.

QUESTION NO: 9

What is true about automating interactions with Anypoint Platform using tools such as Anypoint Platform REST APIs, Anypoint CU, or the Mule Maven plugin?

A. Access to Anypoint Platform APIs and Anypoint CU can be controlled separately through the roles and permissions in Anypoint Platform, so that specific users can get access to Anypoint CLI white others get access to the platform APIs

B. Anypoint Platform APIs can ONLY automate interactions with CloudHub, while the Mule Maven plugin is required for deployment to customer-hosted Mule runtimes

C. By default, the Anypoint CLI and Mule Maven plugin are NOT included in the Mule runtime, so are NOT available to be used by deployed Mule applications

D. API policies can be applied to the Anypoint Platform APIs so that ONLY certain LOBs have access to specific functions

ANSWER: C

Explanation:

By default, the Anypoint CLI and Mule Maven plugin are NOT included in the Mule runtime, so are NOT available to be used by deployed Mule applications. This is correct because these are external administration and build/deployment tools, rather than runtime capabilities packaged within a Mule application or supplied by the Mule runtime engine. Anypoint CLI is installed independently on a workstation or automation agent and is used to execute commands against Anypoint Platform services. Similarly, the Mule Maven plugin is configured in a project's Maven build and is executed by Maven in a development or CI/CD environment to package, deploy, and manage applications. A deployed application runs its application code, configuration, connectors, and dependencies in Mule runtime; it does not gain the ability to invoke CLI commands or Maven deployment goals simply because it is running. Keeping these tools outside the runtime supports repeatable delivery automation, where a CI/CD process authenticates to Anypoint Platform and performs deployments or environment-management actions before or after application execution. MuleSoft documents the separate installation and use of the CLI in its Anypoint CLI documentation, and documents Maven-based deployment separately in the Mule Maven Plugin documentation.

QUESTION NO: 10

A company uses a hybrid Anypoint Platform deployment model that combines the EU control plane with customer-hosted Mule runtimes. After successfully testing a Mule API implementation in the Staging environment, the Mule API implementation is set with environment-specific properties and must be promoted to the Production environment. What is a way that MuleSoft recommends to configure the Mule API implementation and automate its promotion to the Production environment?

A. Bundle properties files for each environment into the Mule API implementation's deployable archive, then promote the Mule API implementation to the Production environment using Anypoint CLI or the Anypoint Platform REST APIs

B. Modify the Mule API implementation's properties in the API Manager Properties tab, then promote the Mule API implementation to the Production environment using API Manager

C. Modify the Mule API implementation's properties in Anypoint Exchange, then promote the Mule API implementation to the Production environment using Runtime Manager

D. Use an API policy to change properties in the Mule API implementation deployed to the Staging environment and another API policy to deploy the Mule API implementation to the Production environment

ANSWER: A

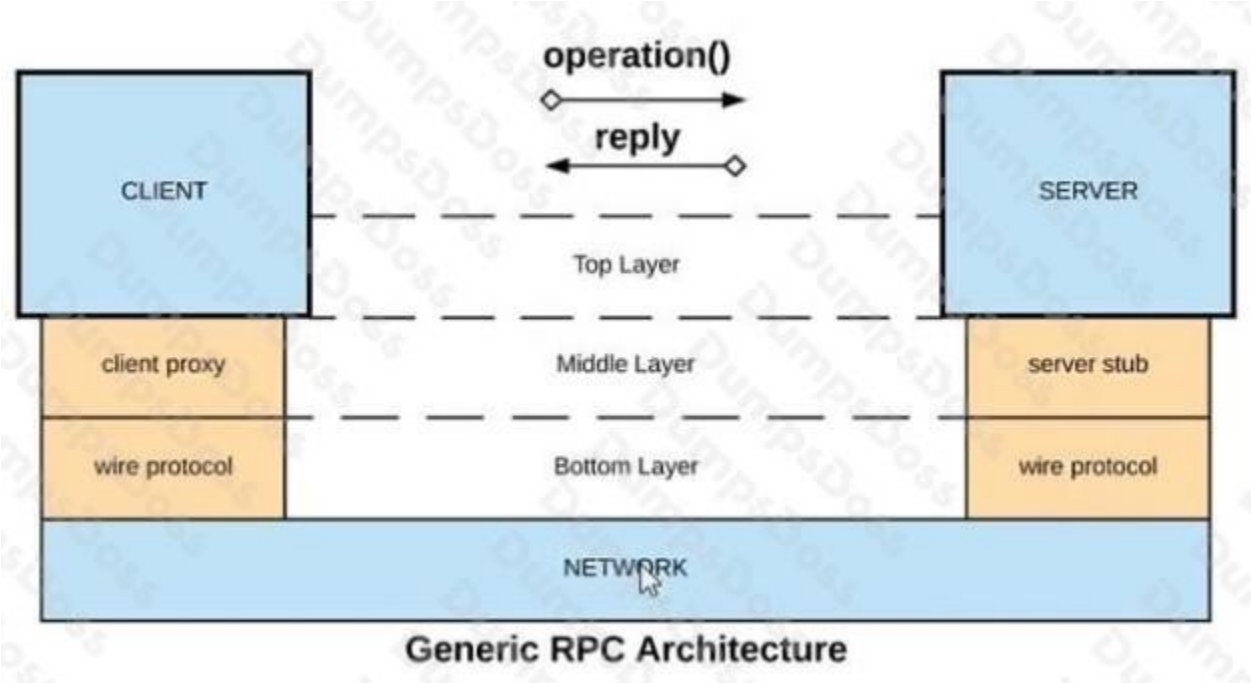
Explanation:

Bundle properties files for each environment into the Mule API implementation's deployable archive, then promote the Mule API implementation to the Production environment using Anypoint CLI or the Anypoint Platform REST APIs. This approach keeps configuration under source control with the application and enables a repeatable release process: the same tested application artifact can contain configuration files for the applicable environments, while the deployment process selects the Production-specific configuration. Mule applications support externalized configuration through property files, including YAML and properties formats, so values such as endpoint URLs, credentials references, and operational settings can be separated from flow logic. See MuleSoft's Configuring Properties documentation.

Automation through Anypoint CLI or Anypoint Platform REST APIs is appropriate for a hybrid deployment because it makes promotion scriptable and consistent across environments. A CI/CD pipeline can deploy the approved artifact to the customer-hosted Production runtime target associated with the EU control plane, apply the intended environment configuration, and avoid manual release steps. MuleSoft documents the available command-line capabilities and authentication patterns in the Anypoint CLI documentation. This promotes controlled, auditable deployments while preserving environment-specific behavior.

QUESTION NO: 11

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. XML over HTTP

C. CORBA over IIOP

D. XML over UDP

ANSWER: B

Explanation:

XML over HTTP is a valid API because it exposes a capability through a broadly accessible, network-based interface using HTTP, with a defined data representation that clients can consume independently of the implementation. In API-led connectivity, an API is a managed and reusable contract for accessing a capability or data, rather than merely a point-to-point integration mechanism. HTTP provides the standard web transport used by API clients and gateways, while XML can provide a structured request and response payload format. The API contract can describe resources or operations, request and response messages, security requirements, and expected behavior, allowing the interface to be discovered, governed, reused, and evolved without requiring consumers to be coupled to the underlying application technology.

MuleSoft's API-led approach organizes these reusable interfaces into experience, process, and system APIs. The protocol and payload format do not by themselves determine the layer; instead, the interface's purpose and its contract determine how it participates in the application network. An XML-over-HTTP interface can therefore be implemented, managed, secured, and published as an API within Anypoint Platform. See MuleSoft's API-led connectivity overview and the API Manager documentation.

QUESTION NO: 12

An IT Security Compliance Auditor is assessing which nonfunctional requirements (NFRs) are already being implemented to meet security measures.

* The Web API has Rate-Limiting SLA

* Basic Authentication - LDAP

* JSON Threat Protection

* TP Allowlist policies applied

Which two NFRs-are enforced?

A. The API invocations are coming from a known subnet range

B. Username/password supported to validate login credentials

C. Sensitive data is masked to prevent compromising critical information

D. The API is protected against XML invocation attacks

E. Performance expectations are to be allowed up to 1,000 requests per second

ANSWER: A B

Explanation:

The API invocations are coming from a known subnet range is enforced by the applied allowlist policy. An IP allowlist restricts access based on the source IP address or CIDR subnet of the calling client. This implements a network-access security requirement: only traffic originating from approved, known network ranges can reach the API through the governed endpoint. MuleSoft API Manager provides an IP allowlist policy for enforcing this kind of source-address restriction.

Username/password supported to validate login credentials is enforced by Basic Authentication with LDAP. Basic authentication requires a client to provide credentials, while LDAP supplies the identity store against which those credentials are validated. Together, these controls implement an authentication requirement by verifying the identity associated with each API request before allowing access to the protected resource. The policy therefore directly supports the stated security NFR for credential-based login validation.

These controls address distinct security layers: the allowlist limits where requests may originate, and LDAP-backed basic authentication validates who is making the request. MuleSoft policy documentation describes applying policies to API proxies and managed API endpoints to enforce such runtime controls. See MuleSoft gateway policies and Basic Authentication policy documentation.

QUESTION NO: 13

4 Production environment is running on a dedicated Virtual Private Cloud (VPC) on CloudHub 1,0, and the security team guidelines clearly state no traffic on HTTP.

Which two options support these security guidelines?

Choose 2 answers

A. Configure the HTTPS protocol in HTTP listener in the Mule application

B. Create a custom policy to apply to outgoing and incoming HTTP requests to control access to a configured API endpoint

C. Remove the entry from the VPC firewall rule

D. Configure the IP Blocklist policy to control access to a configured API endpoint from either a single IP address or a range of IP addresses.

E. Add the entry in the VPC firewall rule.

ANSWER: A C

Explanation:

Configuring the HTTPS protocol in HTTP listener in the Mule application ensures that the application endpoint is configured to accept transport-layer-secured HTTPS connections. HTTPS protects data in transit by using TLS, which provides encryption and server authentication for requests sent to the deployed Mule application. For a production workload, the listener configuration should use an HTTPS listener configuration and the required certificate and key material, rather than exposing an HTTP listener for client traffic.

Removing the entry from the VPC firewall rule is also required when that entry permits the CloudHub HTTP listener port. A dedicated VPC firewall controls which inbound ports can reach workers in the VPC. Removing the HTTP allowance prevents HTTP requests from reaching the application at the network boundary, while a separate rule can allow only the required HTTPS port and trusted source ranges. Applying both controls provides defense in depth: the Mule application is configured for TLS, and the VPC network policy does not expose HTTP connectivity.

MuleSoft documents CloudHub VPC firewall configuration and its inbound-rule behavior in the CloudHub Networking Guide. For configuring TLS-enabled application endpoints, see the HTTP Listener documentation.