DUMPSARENA

Salesforce Certified MuleSoft Integration Foundations

Salesforce Mule-101

Version Demo

Total Demo Questions: 7

Total Premium Questions: 76

Buy Premium PDF

https://dumpsarena.co

sales@dumpsarena.co

sales@dumpsarena.co
dumpsarena.co

Topic Break Down

Topic	No. of Questions
Topic 1, Application Networks and API-Led Connectivity	27
Topic 2, Anypoint Platform	27
Topic 3, APIs	22
Total	76

QUESTION NO: 1

A support team is investigating a failed order request that passed through several Mule applications and downstream microservices. The team must reconstruct the request path and inspect the detailed failure events associated with that specific request.

Which two observability approaches are most useful? Select two.

A. Tracing

B. Logging

C. Metrics

D. Analytics

E. Data mining

ANSWER: A B

Explanation:

Tracing is useful because it follows a request across multiple services and identifies the path, dependencies, timing, and point of failure. **Logging** provides detailed event information, including error messages and contextual details generated by the participating applications.

Metrics provide aggregate measures such as request counts, error rates, and response times, but do not by themselves reconstruct one request's path. Analytics can reveal broader trends, while data mining is not the primary operational approach for investigating a specific distributed transaction.

QUESTION NO: 2

According to MuleSoft, which major benefit does a Center for Enablement (C4E) provide for an enterprise and its lines of business?

A. Centrally managing return on investment (ROI) reporting from lines of business to leadership

B. Accelerating self-service by the lines of business

C. Enabling Edge security between the lines of business and public devices

D. Centralizing project management across the lines of business

ANSWER: B

Explanation:

Accelerating self-service by the lines of business is correct because a Center for Enablement is designed to help an organization scale integration and API delivery beyond a centralized technical team. The C4E establishes reusable standards, governance, architecture guidance, training, and shared assets that enable teams throughout the business to discover, consume, and build integrations more independently. Rather than becoming a delivery bottleneck, the central enablement function equips distributed teams with the capabilities and guardrails needed to solve business problems quickly and consistently.

In MuleSoft's operating model, this approach supports composable enterprise practices: teams can reuse APIs and integration assets that have already been created, reducing duplicate effort and shortening delivery time. Self-service does not mean an absence of oversight; it is enabled through a governed platform, documented reusable assets, and clear practices that allow lines of business to innovate within agreed enterprise standards. This combination of autonomy, reuse, and governance is the central value of a C4E and helps organizations increase the speed and scale of digital initiatives. See MuleSoft's overview of the Center for Enablement and its guidance on API-led connectivity.

QUESTION NO: 3

An integration architect is designing an API that must accept requests from API clients for both XML and JSON content over HTTP/1.1 by default.

A. REST

B. GraphQL

C. gRPC

D. SOAP

ANSWER: A

Explanation:

REST is correct because it is an architectural style for HTTP-based APIs that can expose the same resource in multiple media-type representations, including JSON and XML. An API client identifies the representation it is sending with the `Content-Type` header, such as `application/json` or `application/xml`. It can also indicate its preferred response representation through the HTTP `Accept` header. This is HTTP content negotiation, a standard mechanism that works naturally with RESTful resource-oriented APIs over HTTP/1.1.

In Mule applications, HTTP listeners receive the request headers and payload, allowing flows to inspect or route based on the inbound media type. DataWeave can then transform the payload from JSON or XML into a common internal model and generate an appropriate response representation. This supports clients with differing representation requirements while maintaining one consistent API resource model and HTTP interface. MuleSoft's documentation describes the HTTP Listener's handling of inbound HTTP requests and headers at HTTP Listener Reference. The HTTP semantics and the role of `Accept` and `Content-Type` in selecting representations are defined by the IETF in RFC 9110: HTTP Semantics.

QUESTION NO: 4

A platform architect needs to secure, route, and apply policies to requests entering an organization's APIs from mobile applications and external partners.

Which architectural component is primarily responsible for this north-south API traffic?

A. Service mesh

B. API gateway

C. API client

D. System API

ANSWER: B

Explanation:

API gateway is correct because it manages traffic between external API clients and backend API implementations. An API gateway provides a central enforcement point for policies such as authentication, authorization, rate limiting, and traffic routing.

A service mesh primarily manages service-to-service communication inside a distributed application. An API client consumes an API rather than governing inbound traffic, and a System API exposes system-of-record capabilities but is not itself the architectural gateway for external traffic management.

QUESTION NO: 5

An organization has deployed an existing customer API and wants to govern access consistently without modifying the API implementation. The organization also wants visibility into how the API is being used.

Which two API Manager capabilities meet these requirements? Select two.

A. Apply runtime policies to the managed API

B. Review API analytics and usage information

C. Build the Mule application implementation

D. Deploy the Mule application to a runtime target

E. Publish reusable API assets for discovery

ANSWER: A B

Explanation:

API Manager can apply runtime policies, such as authentication, client access, and rate-limiting policies, to a managed API instance without requiring the API implementation to contain equivalent code. API Manager also provides API analytics that help teams review usage and operational behavior.

Anypoint Studio is used to build Mule applications, Runtime Manager is used to deploy and operate them, and Anypoint Exchange is used to discover and share reusable assets. Those capabilities do not provide the central runtime API governance described in the scenario.

QUESTION NO: 6

What are two reasons why a typical Mulesoft customer favors a Mulesoft-hosted Anypoint platform runtime plane over a customer-hosted runtime for its Mule application deployments?

A. Reduced IT operations effort

B. Increased application isolation

C. Increased application throughput

D. Reduced time-to-market for the first application

E. Reduced application latency

ANSWER: A D

Explanation:

“Reduced IT operations effort” is correct because a MuleSoft-hosted runtime plane, such as CloudHub, is operated as a managed cloud service. MuleSoft manages the underlying runtime infrastructure and its operational responsibilities, including platform availability, infrastructure maintenance, scaling capabilities, and runtime-plane updates. This lets customer teams focus more of their effort on designing, deploying, monitoring, and improving integrations rather than provisioning and operating the supporting hosting environment.

“Reduced time-to-market for the first application” is also correct because customers can begin deploying to a ready-to-use managed runtime plane without first building their own hosting foundation. A customer-hosted model requires upfront work to select, provision, secure, configure, and operate infrastructure, as well as establish networking and deployment practices. CloudHub removes much of that initial platform setup, enabling teams to deploy an application sooner and use Anypoint Platform’s centralized deployment and management capabilities from the outset. These advantages are especially valuable when an organization needs to deliver its first integrations quickly while minimizing the operational burden of maintaining the runtime environment. MuleSoft describes CloudHub as a fully managed cloud service for deploying and running Mule applications. See the CloudHub documentation and the Runtime Manager documentation.

QUESTION NO: 7

According to MuleSoft's API development best practices, which type of API development approach starts with writing and approving an API contract?

A. Implement-first

B. Agile

C. Design-first

D. Catalyst

ANSWER: C

Explanation:

Design-first is correct because it establishes the API contract before implementation begins. The contract is an API specification, commonly written in RAML or OpenAPI, that defines the API's resources, operations, request and response structures, data types, security expectations, and error behavior. Reviewing and approving this specification early gives API consumers, architects, and implementation teams a shared, unambiguous agreement about how the API will behave.

This approach supports parallel work: consumer teams can understand and mock the planned interface while implementation teams build the underlying Mule application. It also makes governance, consistency, and reuse easier because the API design can be assessed against organizational standards before code is created. MuleSoft's Design Center supports this workflow by enabling teams to create and manage API specifications as a central design artifact. Once the contract is agreed upon, implementation can proceed with clear requirements and can be tested for conformance to the defined interface.

For MuleSoft guidance on creating and managing API specifications, see MuleSoft Design Center documentation. Additional background on the role of API specifications in the API lifecycle is available in MuleSoft's API overview.