# 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

- **Instant Download** After Purchase
- **100% Money Back** Guarantee
- **365 Days** Free Update
- **800,000+** Satisfied Customers

**QUESTION 1**

An API implementation is deployed to CloudHub.

What conditions can be alerted on using the default Anypoint Platform functionality, where the alert conditions depend on the end-to-end request processing of the API implementation?

A. When the API is invoked by an unrecognized API client

B. When a particular API client invokes the API too often within a given time period

C. When the response time of API invocations exceeds a threshold

D. When the API receives a very high number of API invocations

Correct Answer: C

When the response time of API invocations exceeds a threshold ********** >> Alerts can be setup for all the given options using the default Anypoint Platform functionality >> However, the question insists on an alert whose conditions depend on the end-to-end request processing of the API implementation. >> Alert w.r.t "Response Times" is the only one which requires end-to-end request processing of API implementation in order to determine if the threshold is exceeded or not. Reference: https://docs.mulesoft.com/api-manager/2.x/using-api- alerts

**QUESTION 2**

Version 3.0.1 of a REST API implementation represents time values in PST time using ISO 8601 hh:mm:ss format. The API implementation needs to be changed to instead represent time values in CEST time using ISO 8601 hh:mm:ss format. When following the semver.org semantic versioning specification, what version should be assigned to the updated API implementation?

A. 3.0.2

B. 4.0.0

C. 3.1.0

D. 3.0.1

Correct Answer: B

**********

As per semver.org semantic versioning specification:

Given a version number MAJOR.MINOR.PATCH, increment the:

-MAJOR version when you make incompatible API changes.

-

MINOR version when you add functionality in a backwards compatible manner.

-

PATCH version when you make backwards compatible bug fixes. As per the scenario given in the question, the API implementation is completely changing its behavior. Although the format of the time is still being maintained as hh:mm:ss and there is no change in schema w.r.t format, the API will start functioning different after this change as the times are going to come completely different. Example: Before the change, say, time is going as 09:00:00 representing the PST. Now on, after the change, the same time will go as 18:00:00 as Central European Summer Time is 9 hours ahead of Pacific Time. >> This may lead to some uncertain behavior on API clients depending on how they are handling the times in the API response. All the API clients need to be informed that the API functionality is going to change and will return in CEST format. So, this considered as a MAJOR change and the version of API for this new change would be 4.0.0

**QUESTION 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

Correct Answer: A

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 4**

An API implementation is deployed on a single worker on CloudHub and invoked by external API clients (outside of CloudHub). How can an alert be set up that is guaranteed to trigger AS SOON AS that API implementation stops responding to API invocations?

A. Implement a heartbeat/health check within the API and invoke it from outside the Anypoint Platform and alert when the heartbeat does not respond

B. Configure a "worker not responding" alert in Anypoint Runtime Manager

C. Handle API invocation exceptions within the calling API client and raise an alert from that API client when the API Is unavailable

D. Create an alert for when the API receives no requests within a specified time period

Correct Answer: B

Configure a "Worker not responding" alert in Anypoint Runtime Manager.

**********

>> All the options eventually helps to generate the alert required when the application stops responding.

>> However, handling exceptions within calling API and then raising alert from API client is inappropriate and silly. There could be many API clients invoking the API implementation and it is not ideal to have this setup consistently in all of

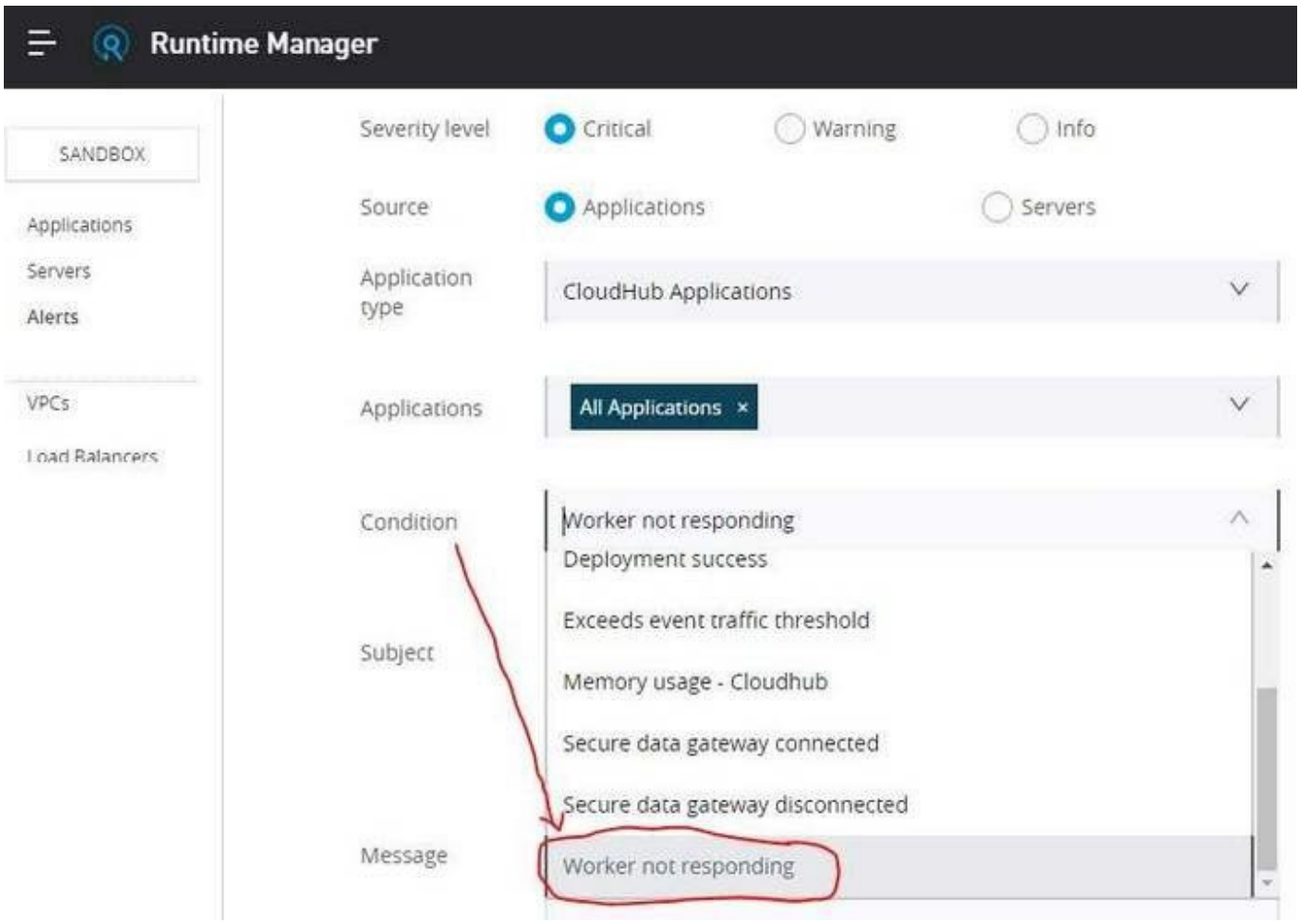
them. Not a realistic way to do. >> Implementing a health check/ heartbeat with in the API and calling from outside to detmine the health sounds OK but needs extra setup for it and same time there are very good chances of generating false

alarms when there are any intermittent network issues between external tool calling the health check API on API implementation. The API implementation itself may not have any issues but due to some other factors some false alarms may go

out.

>> Creating an alert in API Manager when the API receives no requests within a specified time period would actually generate realistic alerts but even here some false alarms may go out when there are genuinely no requests from API clients.

The best and right way to achieve this requirement is to setup an alert on Runtime Manager with a condition "Worker not responding". This would generate an alert AS SOON AS the workers become unresponsive.



**QUESTION 5**

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

Correct Answer: D

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