MCIA-LEVEL-1$^{\text{Q\&As}}$

MuleSoft Certified Integration Architect - Level 1

Pass Mulesoft MCIA-LEVEL-1 Exam with 100% Guarantee

QUESTION 1

An organization has various integrations implemented as Mule applications. Some of these Mule applications are deployed to custom hosted Mule runtimes (on-premises) while others execute in the MuleSoft-hosted runtime plane (CloudHub). To perform the Integra functionality, these Mule applications connect to various backend systems, with multiple applications typically needing to access the backend systems.

How can the organization most effectively avoid creating duplicates in each Mule application of the credentials required to access the backend systems?

A. Create a Mule domain project that maintains the credentials as Mule domain-shared resources Deploy the Mule applications to the Mule domain, so the credentials are available to the Mule applications

B. Store the credentials in properties files in a shared folder within the organization\\'s data center Have the Mule applications load properties files from this shared location at startup

C. Segregate the credentials for each backend system into environment-specific properties files Package these properties files in each Mule application, from where they are loaded at startup

D. Configure or create a credentials service that returns the credentials for each backend system, and that is accessible from customer-hosted and MuleSoft-hosted Mule runtimes Have the Mule applications toad the properties at startup by invoking that credentials service

Correct Answer: D

Explanation:

* "Create a Mule domain project that maintains the credentials as Mule domain-shared resources" is wrong as domain project is not supported in Cloudhub * We should Avoid Creating duplicates in each Mule application but below two options cause duplication of credentials - Store the credentials in properties files in a shared folder within the organization\\'s data center. Have the Mule applications load properties files from this shared location at startup - Segregate the credentials for each backend system into environment- specific properties files. Package these properties files in each Mule application, from where they are loaded at startup So these are also wrong choices * Credentials service is the best approach in this scenario. Mule domain projects are not supported on CloudHub. Also its is not recommended to have multiple copies of configuration values as this makes difficult to maintain Use the Mule Credentials Vault to encrypt data in a .properties file. (In the context of this document, we refer to the .properties file simply as the properties file.) The properties file in Mule stores data as key-value pairs which may contain information such as usernames, first and last names, and credit card numbers. A Mule application may access this data as it processes messages, for example, to acquire login credentials for an external Web service. However, though this sensitive, private data must be stored in a properties file for Mule to access, it must also be protected against unauthorized ?and potentially malicious ?use by anyone with access to the Mule application

QUESTION 2

An XA transaction Is being configured that involves a JMS connector listening for Incoming JMS messages. What is the meaning of the timeout attribute of the XA transaction, and what happens after the timeout expires?

A. The time that is allowed to pass between committing the transaction and the completion of the Mule flow After the timeout, flow processing triggers an error

B. The time that Is allowed to pass between receiving JMS messages on the same JMS connection After the timeout, a new JMS connection Is established

C. The time that Is allowed to pass without the transaction being ended explicitly After the timeout, the transaction Is forcefully rolled-back

D. The time that Is allowed to pass for state JMS consumer threads to be destroyed After the timeout, a new JMS consumer thread is created

Correct Answer: C

Explanation:

*

Setting a transaction timeout for the Bitronix transaction manager Set the transaction timeout either

In wrapper.conf

In CloudHub in the Properties tab of the Mule application deployment The default is 60 secs. It is defined as

mule.bitronix.transactiontimeout = 120

*

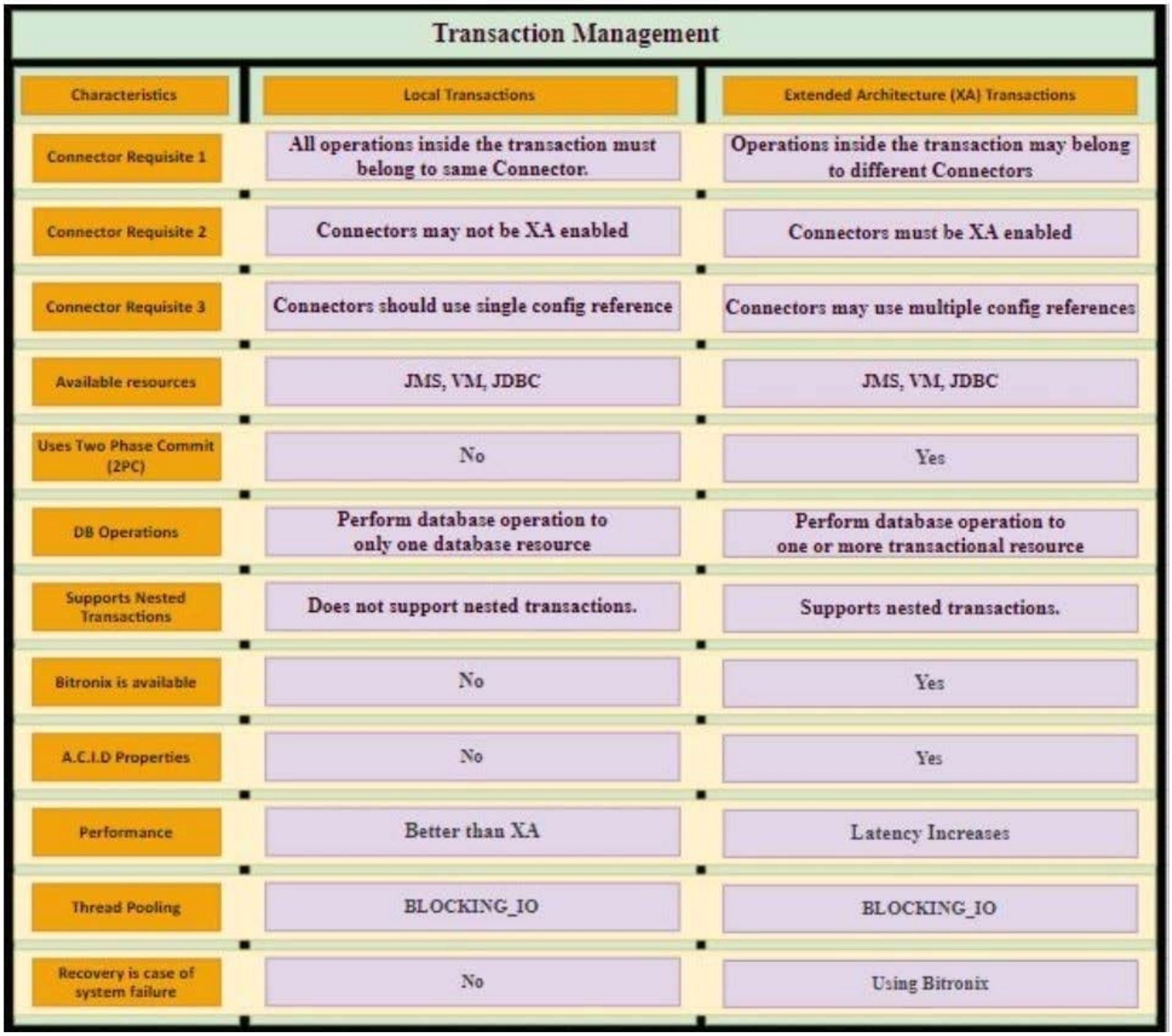
This property defines the timeout for each transaction created for this manager. If the transaction has not terminated before the timeout expires it will be automatically rolled back.

-- Additional Info around Transaction Management:

Bitronix is available as the XA transaction manager for Mule applications To use Bitronix, declare it as a global configuration element in the Mule application

Each Mule runtime can have only one instance of a Bitronix transaction manager, which is shared by all Mule applications

For customer-hosted deployments, define the XA transaction manager in a Mule domain ?Then share this global element among all Mule applications in the Mule runtime



Transcation Management

Characteristics	Local Transactions	Extended Architecture (XA) Transactions
Connector Requisite 1	All operations inside the transaction must belong to same Connector.	Operations inside the transaction may belong to different Connectors
Connector Requisite 2	Connectors may not be XA enabled	Connectors must be XA enabled
Connector Requisite 3	Connectors should use single config reference	Connectors may use multiple config references
Available resources	JMS, VM, JDBC	JMS, VM, JDBC
Uses Two Phase Commit (2PC)	No	Yes
DB Operations	Perform database operation to only one database resource	Perform database operation to one or more transactional resource
Supports Nested Transactions	Does not support nested transactions.	Supports nested transactions.
Bitronix is available	No	Yes
A.C.I.D Properties	No	Yes
Performance	Better than XA	Latency Increases
Thread Pooling	BLOCKING_IO	BLOCKING_IO
Recovery is case of system failure	No	Using Bitronix

QUESTION 3

An organization has strict unit test requirement that mandate every mule application must have an MUnit test suit with a test case defined for each flow and a minimum test coverage of 80%.

A developer is building Munit test suit for a newly developed mule application that sends API request to an external rest API.

What is the effective approach for successfully executing the Munit tests of this new application while still achieving the required test coverage for the Munit tests?

A. Invoke the external endpoint of the rest API from the mule floors

B. Mark the rest API invocations in the Munits and then call the mocking service flow that simulates standard responses from the REST API

C. Mock the rest API invocation in the Munits and return a mock response for those invocations

D. Create a mocking service flow to simulate standard responses from the rest API and then configure the mule flows to call the marking service flow

Correct Answer: C

QUESTION 4

What is a core pillar of the MuleSoft Catalyst delivery approach?

A. Business outcomes

B. Technology centralization

C. Process thinking

D. Scope reduction

Correct Answer: A

QUESTION 5

According to MuleSoft, a synchronous invocation of a RESTful API using HTTP to get an individual customer record from a single system is an example of which system integration interaction pattern?

A. Request-Reply

B. Multicast

C. Batch

D. One-way

Correct Answer: A