CHAPTER 8: CONNECT TO A MESSAGING PROVIDER

Theory

WebSphere Application Server supports asynchronous messaging that allows to create, send, receive and read requests. Asynchronous messaging support is based on Java Message Service (JMS) and Java EE Connector Architecture (JCA).

Synchronous messaging requires both applications to be available at the same time as in the example of phone talk. On the other hand, asynchronous messaging doesn't require sending and receiving applications available at the same time. We can use postal service as an example.

Java Message Service (JMS) is a Java API to access to message providers. JMS provider or in other words messaging provider gives the basis for the exchange of messages between applications.

A JMS message consist of three parts;

- Header, must exist in every JMS message and it is assigned automatically mostly by JMS provider when the message put to a JMS destination.
- Properties, are optional and can be application related, provider related and standard properties. Properties provide an efficient mechanism to filter application defined messages.
- Body, contains the main information which needs to be exchanged.

There are 6 different types of JMS messages based on the data it contains:

- Message, it is the base class and used for event notification.
- BytesMessage, stores the data as sequence of bytes and used for exchanging data in a format that is native to the application.
- TextMessage, stores the data as a string and used for simple text message exchanging.
- StreamMessage, stores the data as a sequence of primitive Java types.
- MapMessage, stores the data as a set of key-value pairs and used for delivering keyed data that can change in different messages.
- ObjectMessage, stores a serialized Java object and used for Java object exchanging.

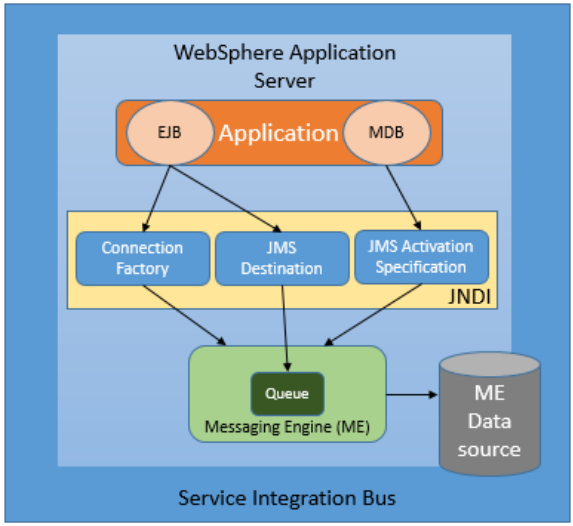
XML (Extensible Markup Language) is mostly used format in Java applications. It allows you to describe the data independent of the language that the application is written in or the platform which the application runs on.

A service integration bus is a group of servers or clusters that cooperate to provide asynchronous messaging service. An application server or a cluster can be a member of a bus. With service integration bus

- Any application can exchange messages with any application using destination.
- A message producing application can produce messages for a destination regardless of the messaging engine that the producer uses to connect to the bus.
- A message consuming application can consume messages from a destination regardless of which messaging engine the consumer uses to connect to the bus.

A connection factory is an object that is used by JMS client to establish the connection to a JMS provider. It is stored and managed object in a JNDI namespace. A connection factory can create connections to one messaging provider only. If you need to connect to a different provider, you need to create a new connection factory.

A JMS destination is an object that can be a JMS queue or a JMS topic. It represents the target of messages the client produces and the source of messages that the client consumes.



JMS activation specification is the standard way to manage the relation between Message-driven beans (MDB) which are a special class of Enterprise Java Beans (EJB) and a messaging engine.

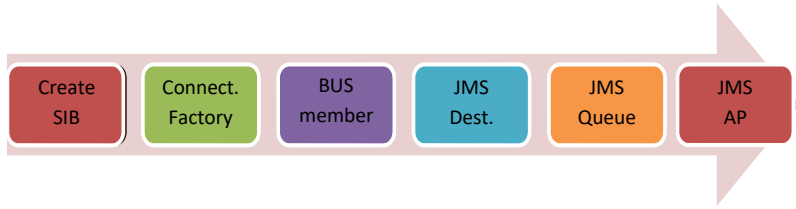
AIM

In this lab exercise, you will configure default messaging provider in WebSphere Application Server to enable the communication between applications that run on the application server and the other applications.

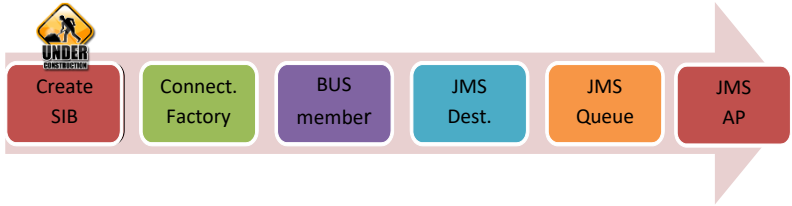
In order to achieve this, you need to complete following steps:

1. Create a Service Integration Bus
2. Configure JMS connection factory
3. Add a Bus member
4. Configure JMS destination
5. Configure JMS queue
6. Configure JMS Activation Specifications

Lab Exercise 8: CONNECT TO A MESSAGING PROVIDER

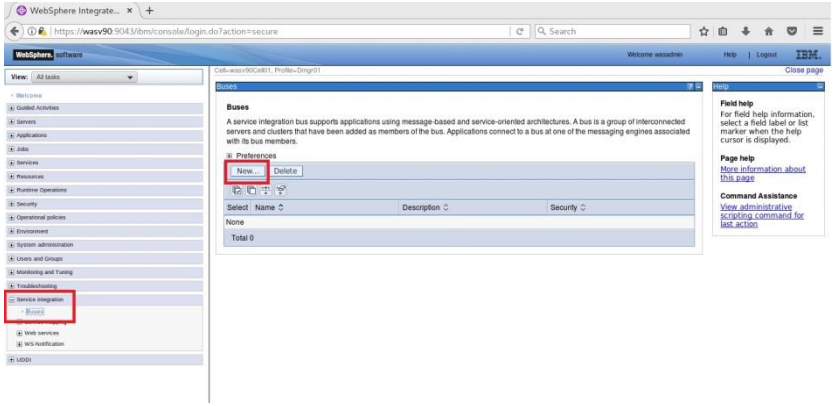


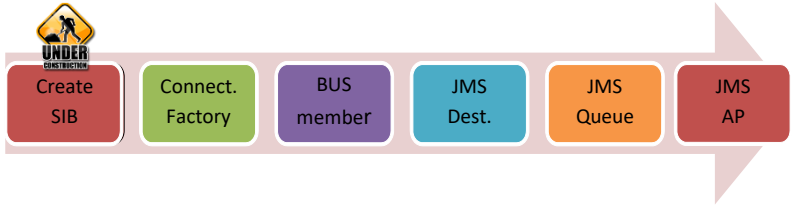
1. **Create a Service Integration Bus**
2. **Configure JMS connection factory**
3. **Add a Bus member**
4. **Configure JMS destination**
5. **Configure JMS queue**
6. **Configure JMS Activation Specifications**



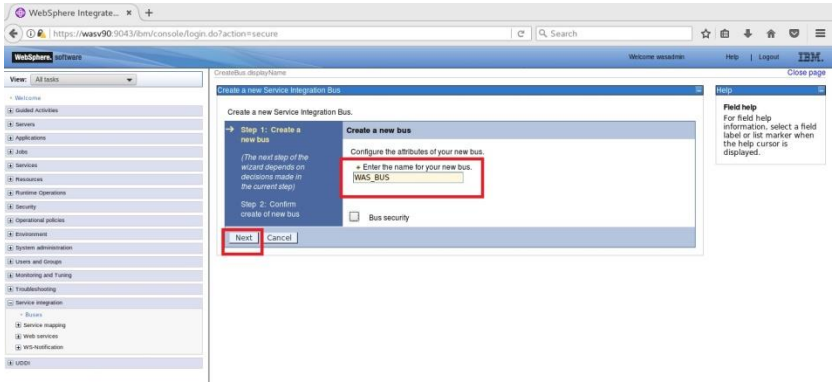
Task 1: Create a Service Integration Bus

Step 1: Navigate to “Service integration>Buses” and then click on “New” to add a new bus.

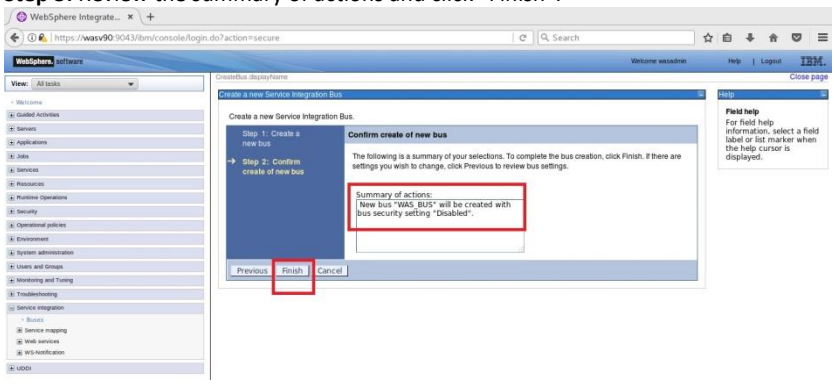


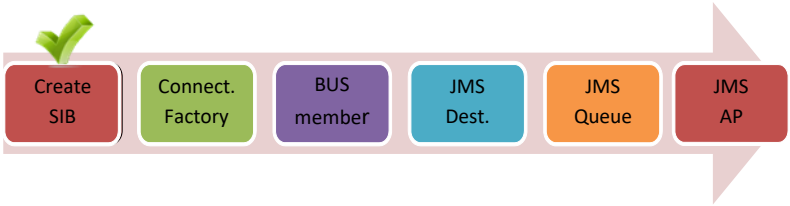


Step 2: You need to enter the name for the new bus, eg. “FC_SampleBus” and then click “Next”.

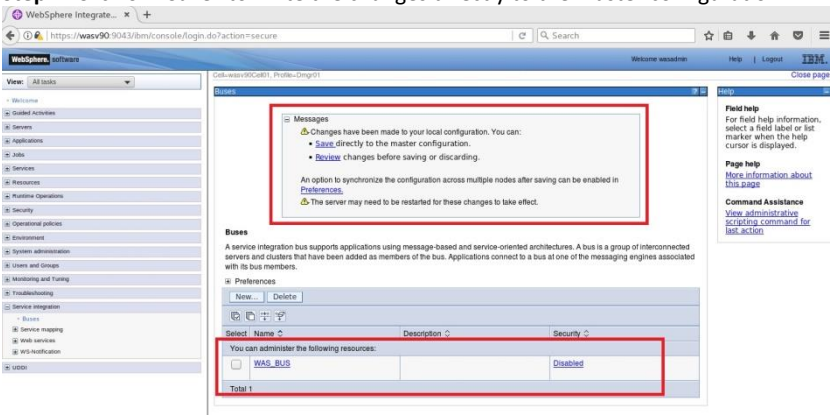


Step 3: Review the summary of actions and click “Finish”.

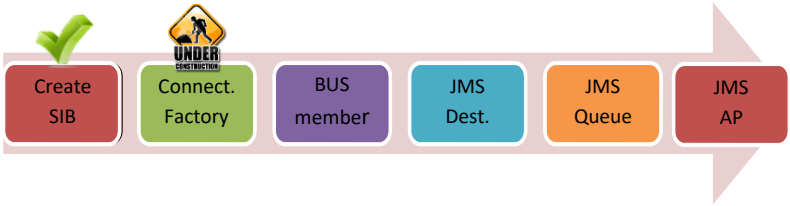




Step 4: Click on “Save” to write the changes directly to the master configuration.

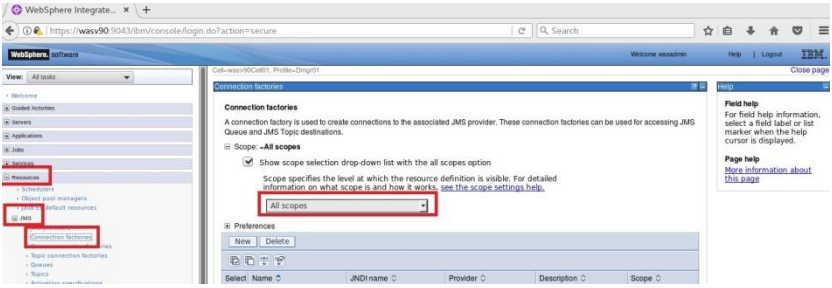


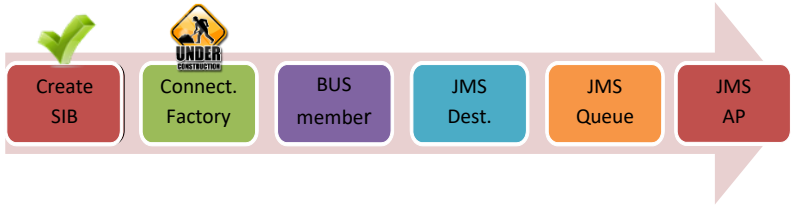
Task 1 is complete!



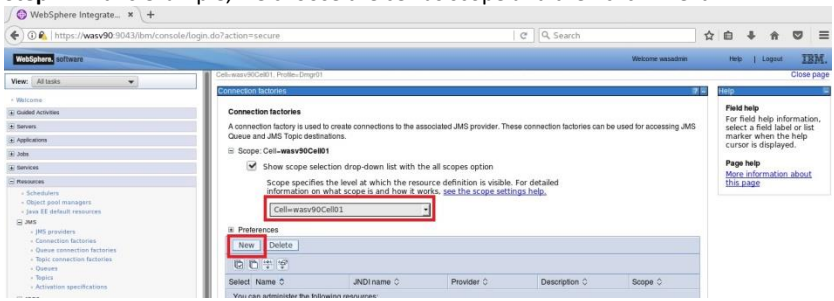
Task 2: Create JMS Factory

Step 1: Navigate to “Resources>JMS>Connection factories” and then select the scope you want to add a connection factory.

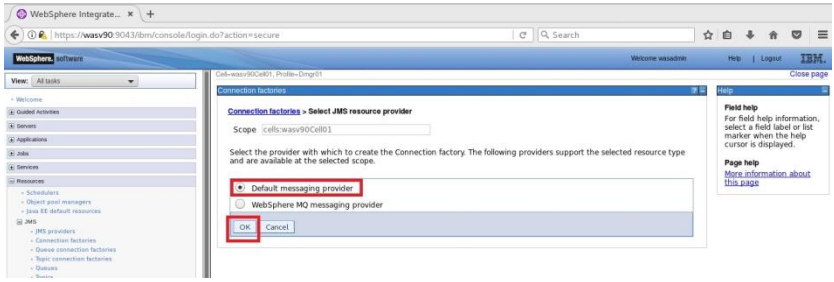


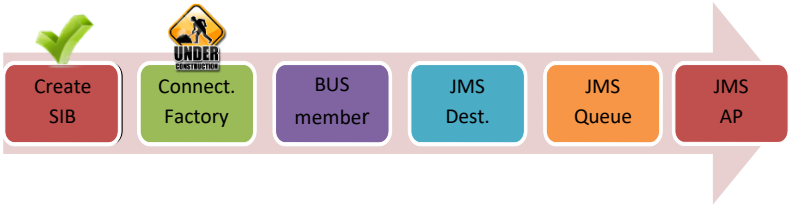


Step 2: In this example, we choose the cell as scope and then click “Next”.



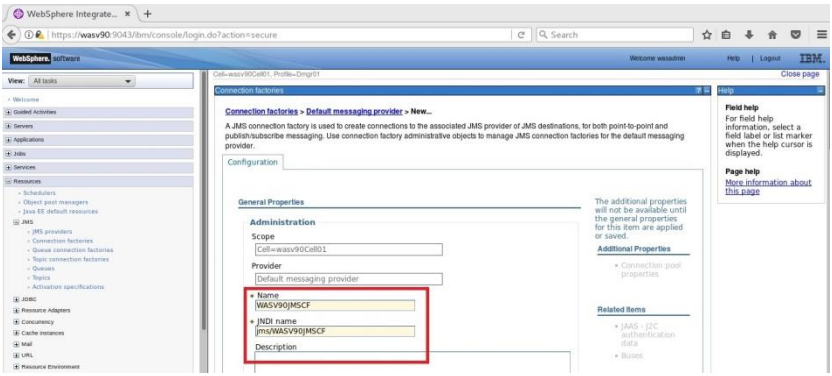
Step 3: Select “Default messaging provider” and then click “OK”. You can choose “WebSphere MQ messaging provider” for WebSphere MQ configuration.

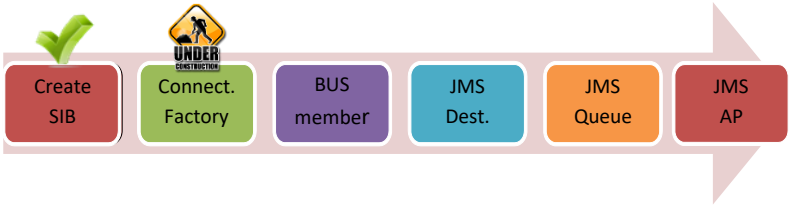




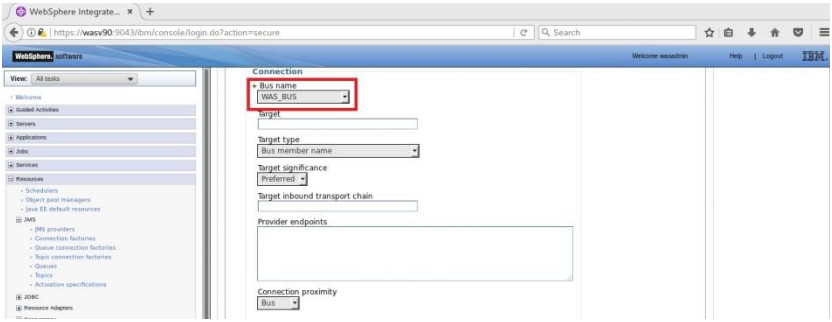
Step 4: You need to define name and JNDI name for the default messaging provider connection factory.

Name: WASV90JMSCF
JDNI name: jms/WASV90JMSCF

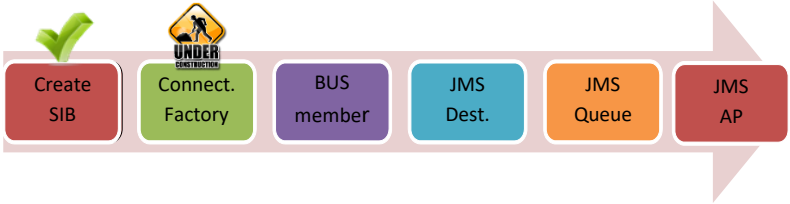




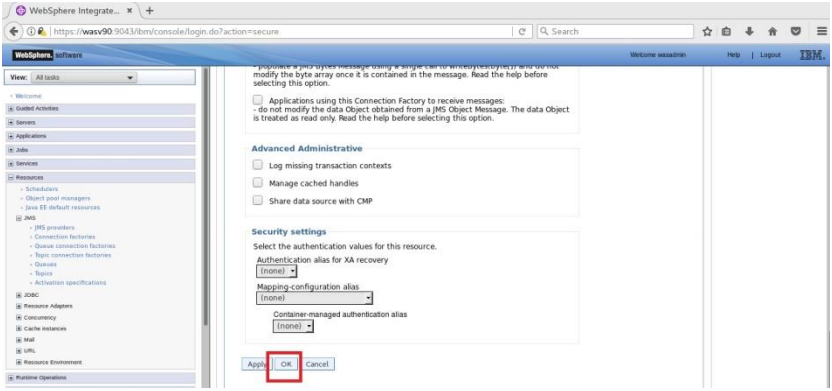
Scroll down to select “Bus name” that we added in the first task.

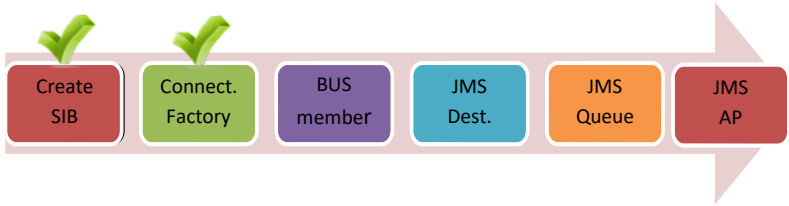


ScrollS

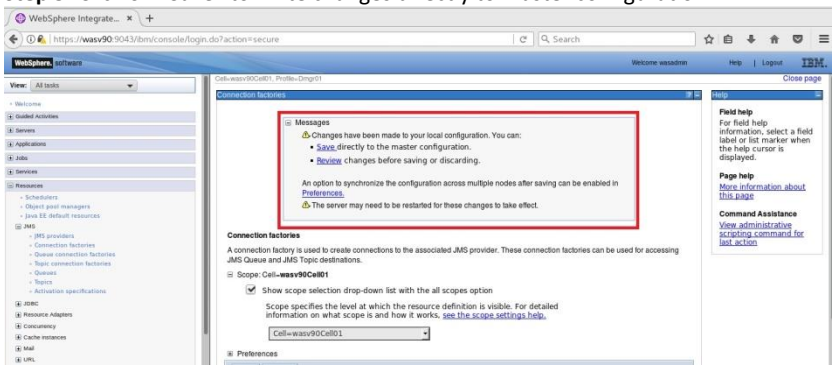


Scroll down and click “OK”.

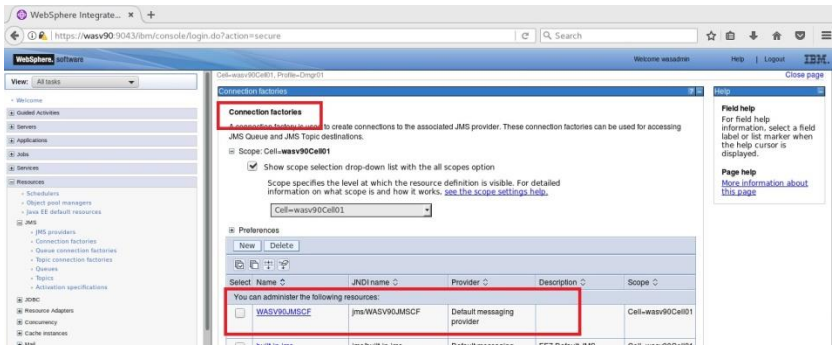




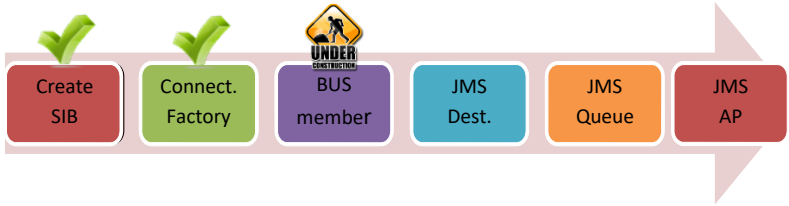
Step 5: Click on “Save” to write changes directly to master configuration.



You should see the newly added connection factory.

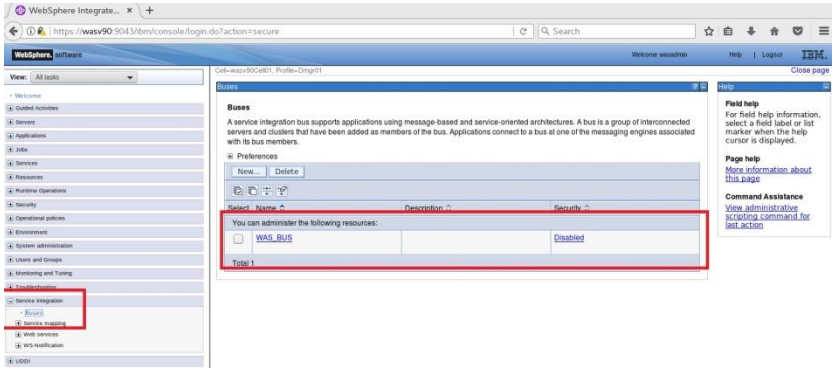


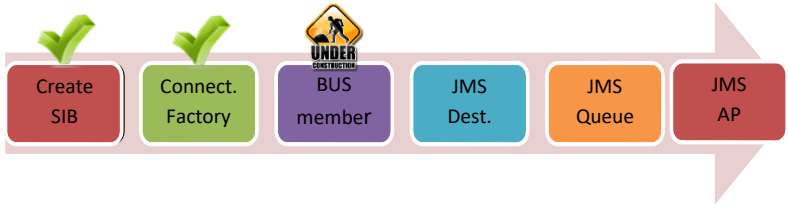
Task 2 is complete!



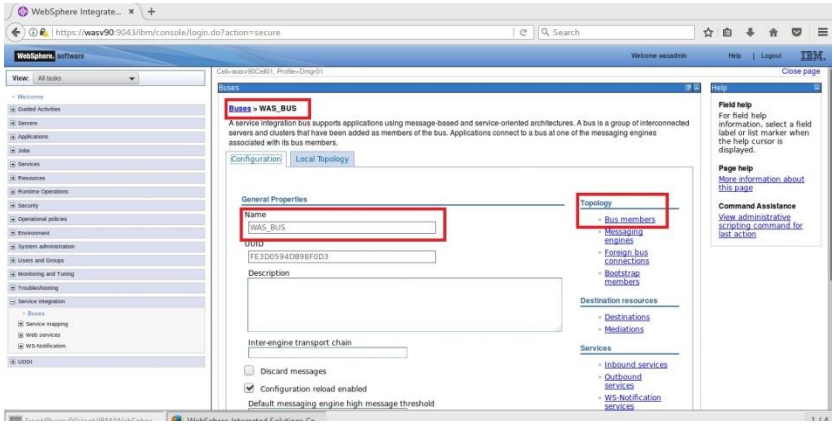
Task 3: Add a Bus member

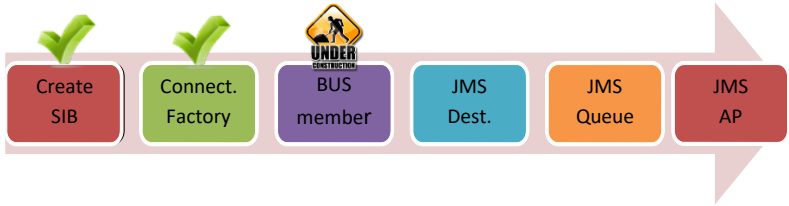
Step 1: Navigate to “Service integration>Buses” and click on the bus name (WAS_Bus) to change parameters.



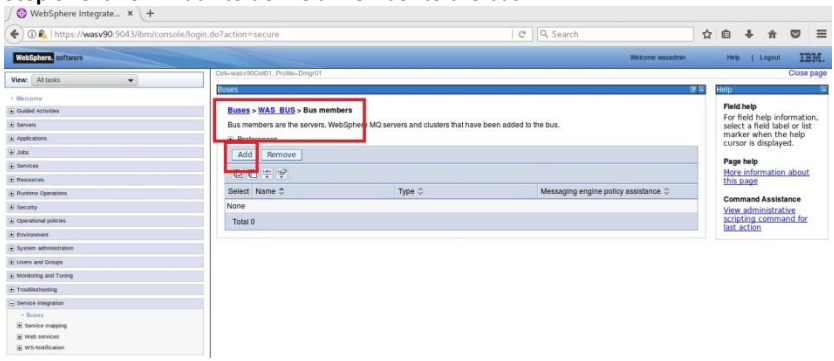


Step 2: Click on “Bus members” under “Topology” menu located on the right.

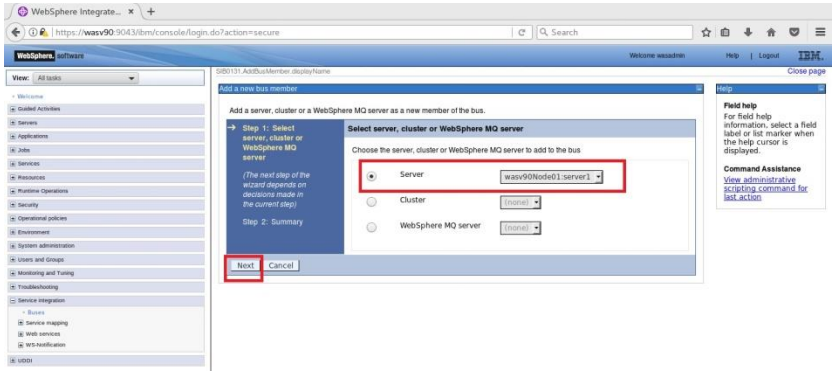


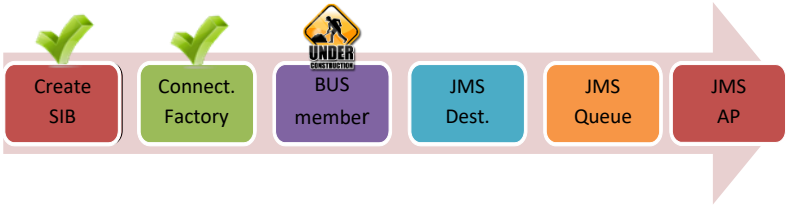


Step 3: Click on “Add” to define a member to the bus.

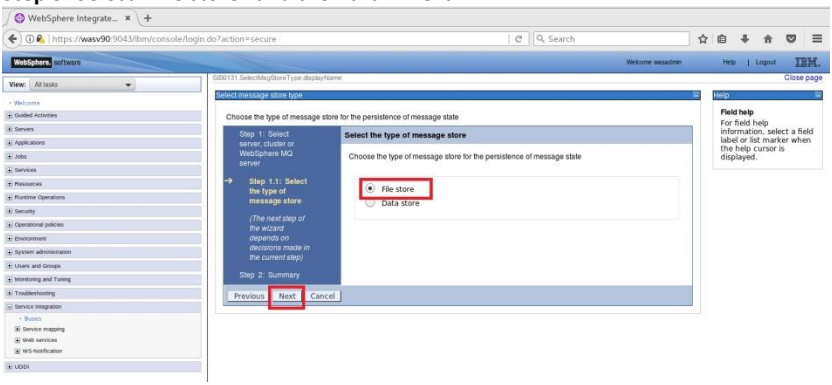


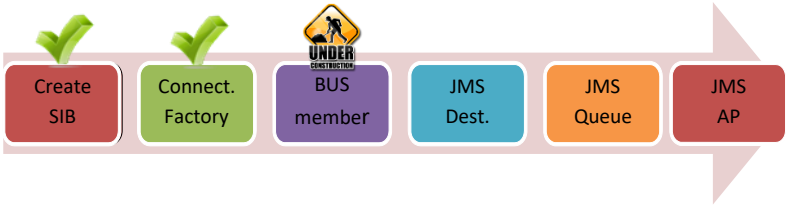
Step 4: Select “Server” or “Cluster” according to your needs and then click on “Next”.



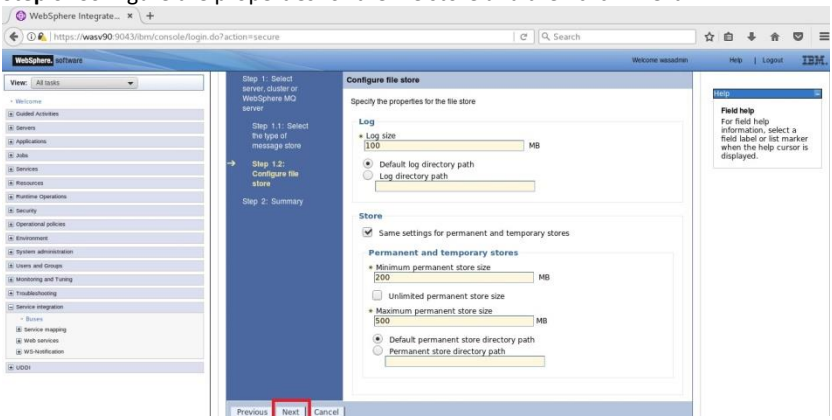


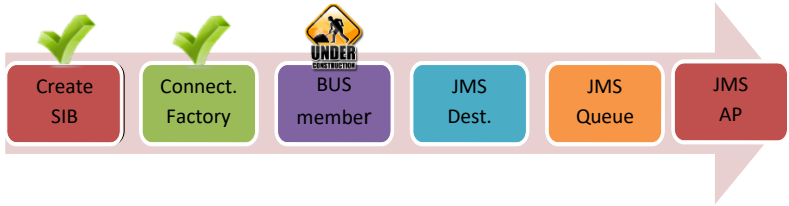
Step 5: Select “File store” and then click “Next”.



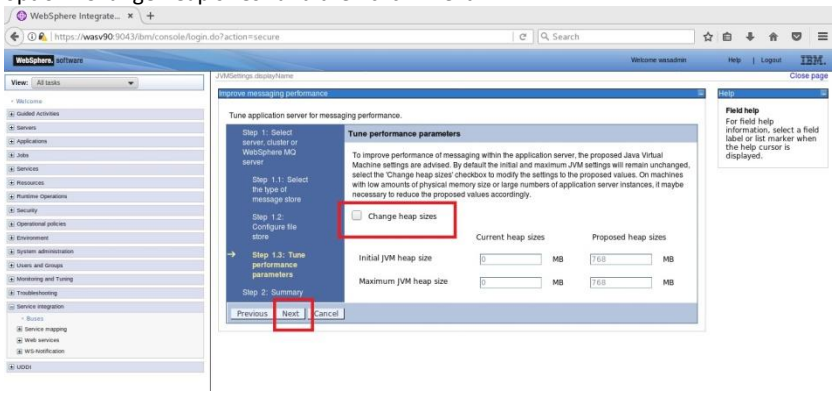


Step 6: Configure the properties for the file store and then click “Next”.

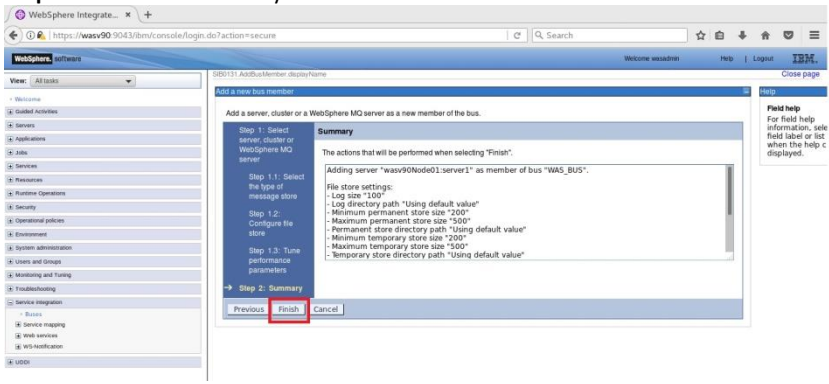


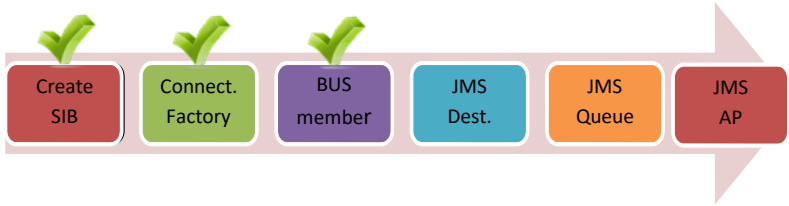


Step 7: You can tune the application server for messaging performance selecting the option “Change heap sizes” and then click “Next”.

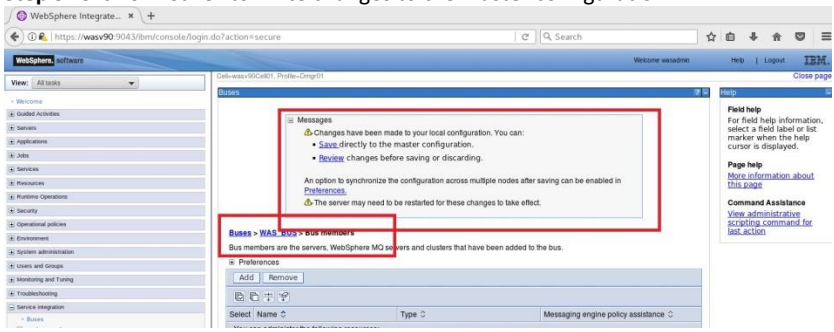


Step 8: Review the summary of actions and then click “Finish”.

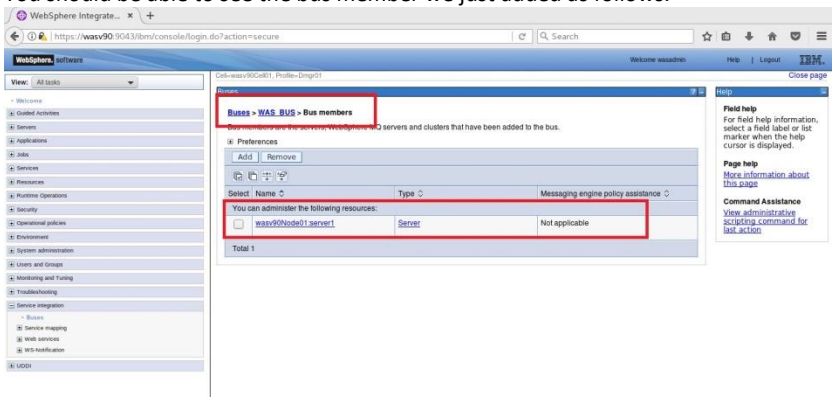




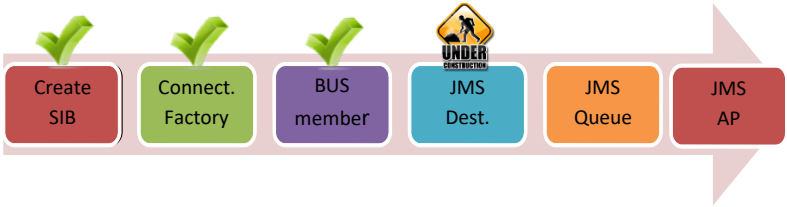
Step 9: Click on “Save” to write changes to the master configuration.



You should be able to see the bus member we just added as follows.

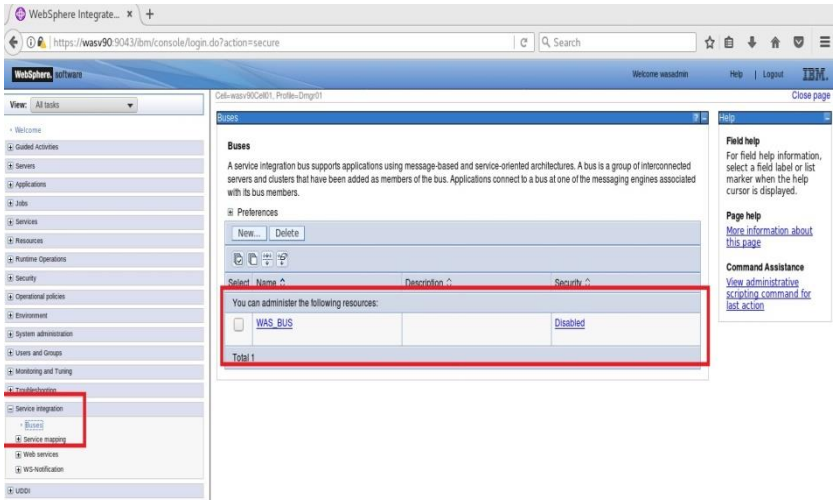


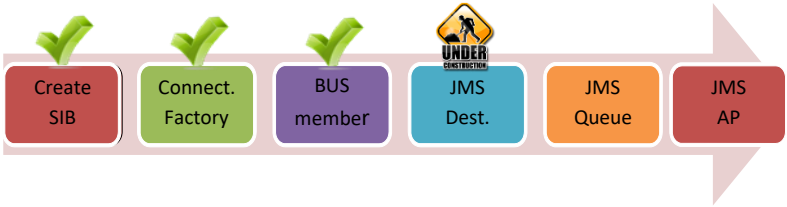
Task 3 is complete!



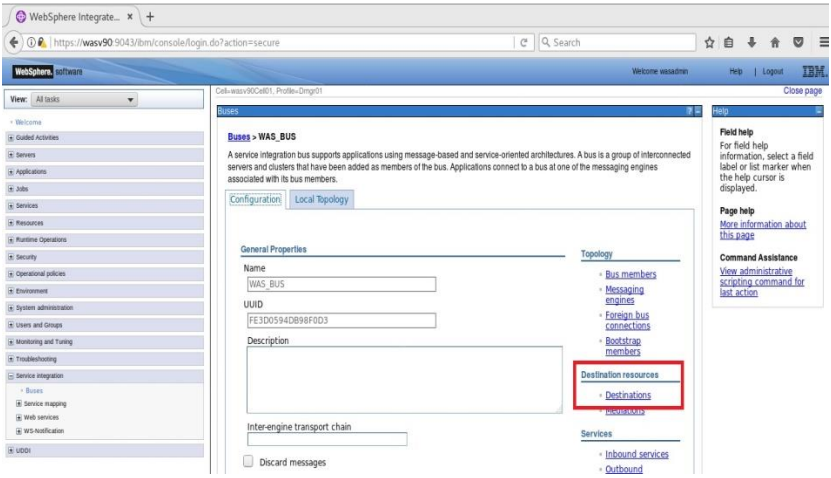
Task 4: Configure JMS Destination

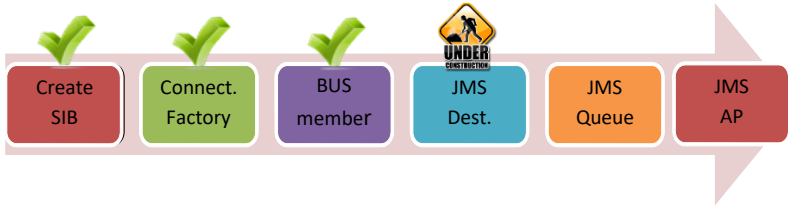
Step 1: Navigate to “Service integration>Buses” and click on the bus name.



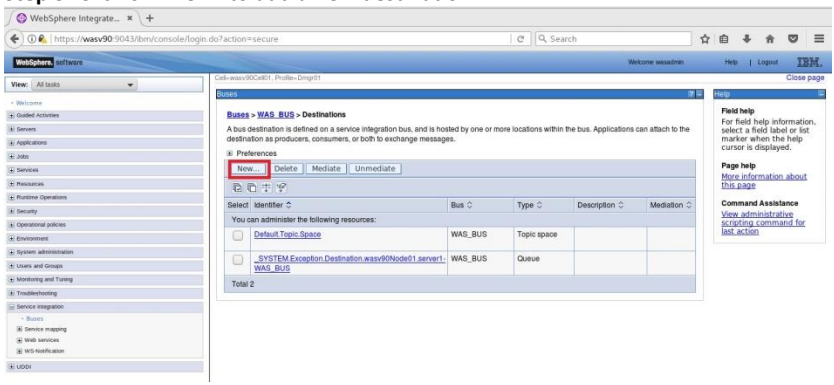


Step 2: Click on “Destinations” under “Destination resources” menu located on the right.

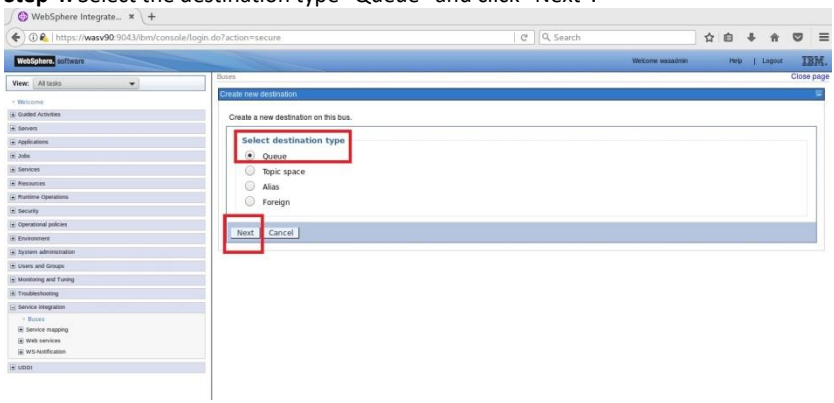


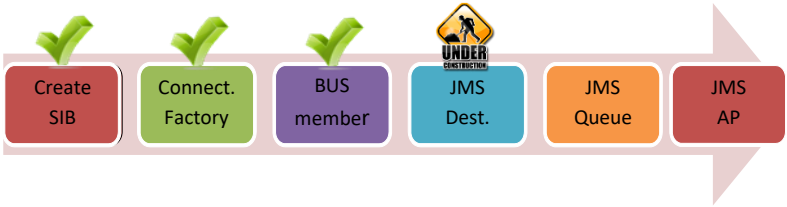


Step 3: Click on “New” to add a new destination.

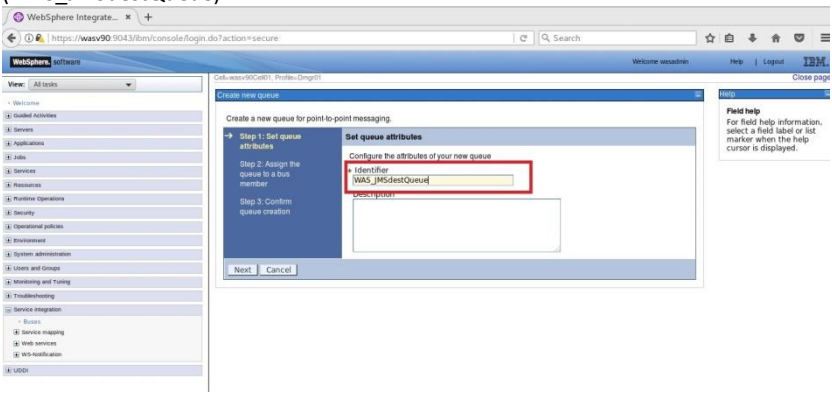


Step 4: Select the destination type “Queue” and click “Next”.

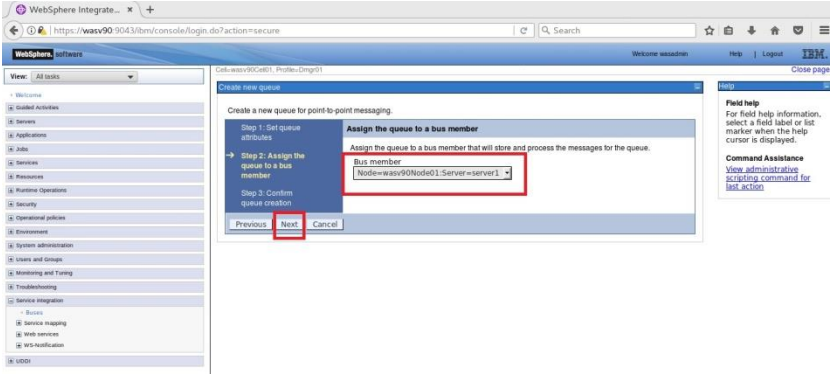


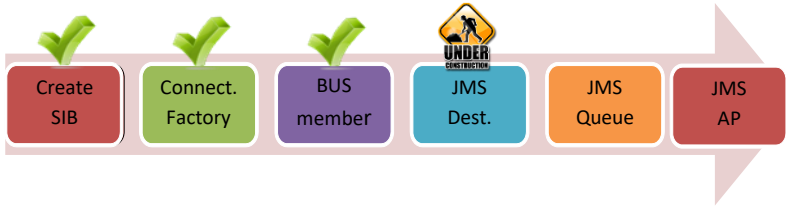


Step 5: Enter an identifier for the new queue and click “Next”. (WAS_JMSdestQueue)

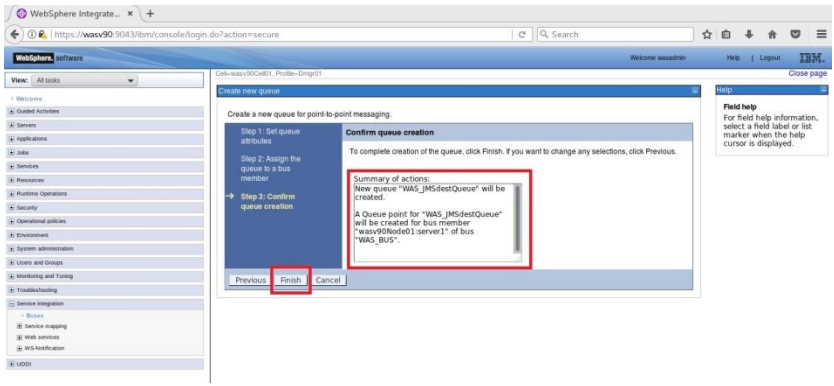


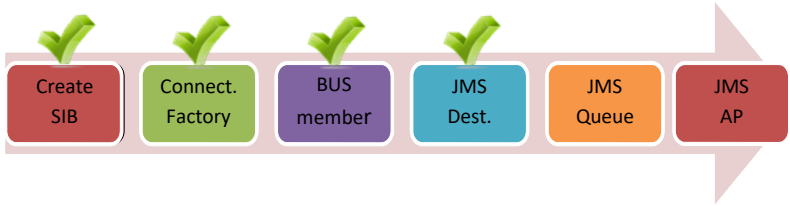
Step 6: Click on “Next”.



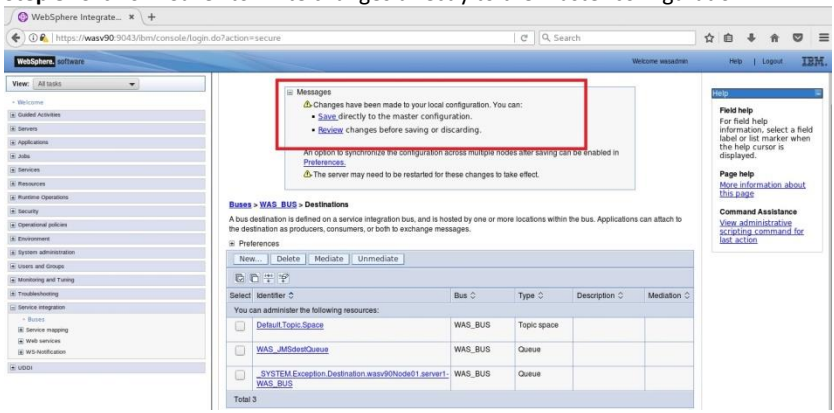


Step 7: Review the summary of actions and click “Finish”.

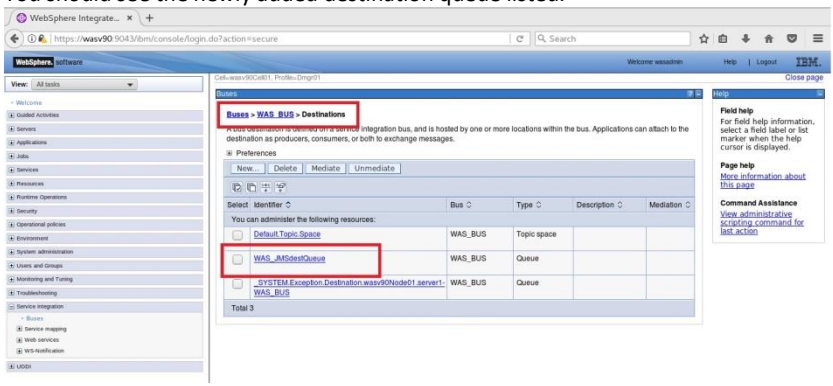




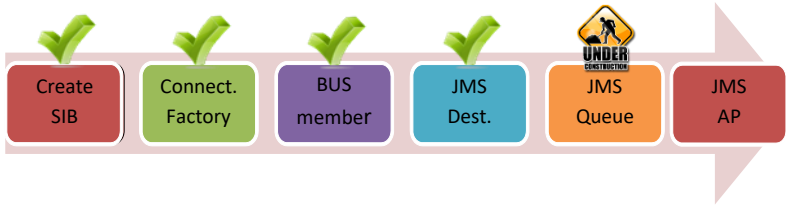
Step 8: Click on “Save” to write changes directly to the master configuration.



You should see the newly added destination queue listed.

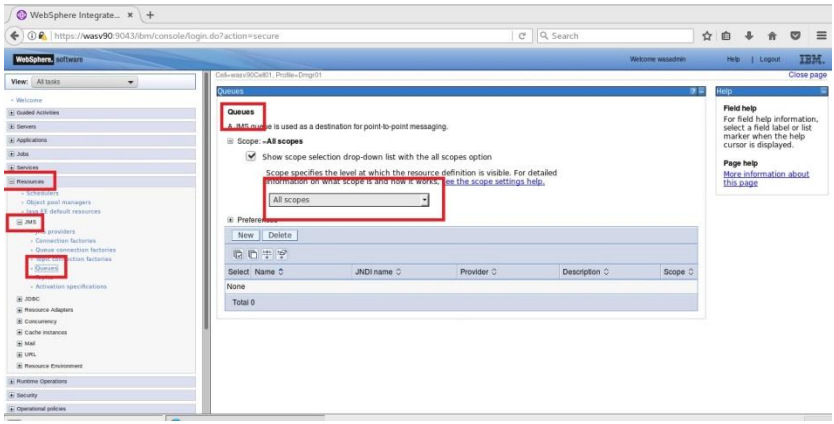


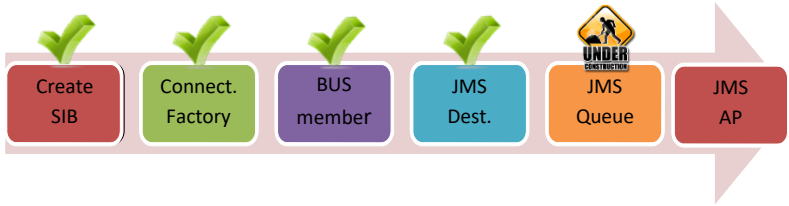
Task 4 is complete!



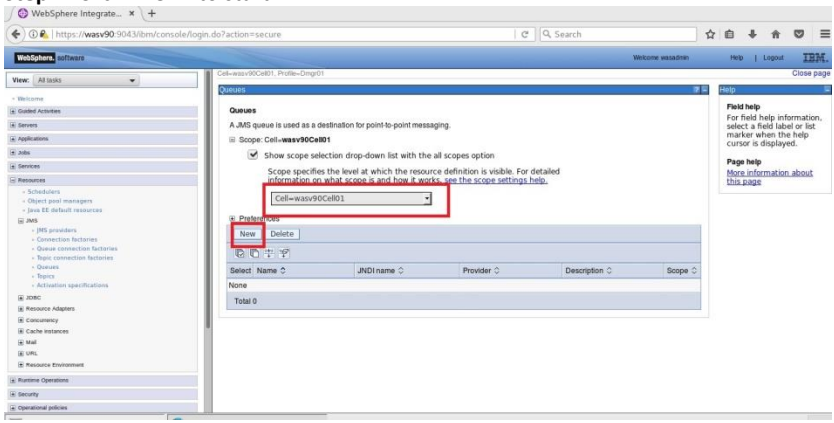
Task 5: Configure JMS queue

Step 1: Navigate to “Resources>JMS>Queues” and select the scope of the definition.

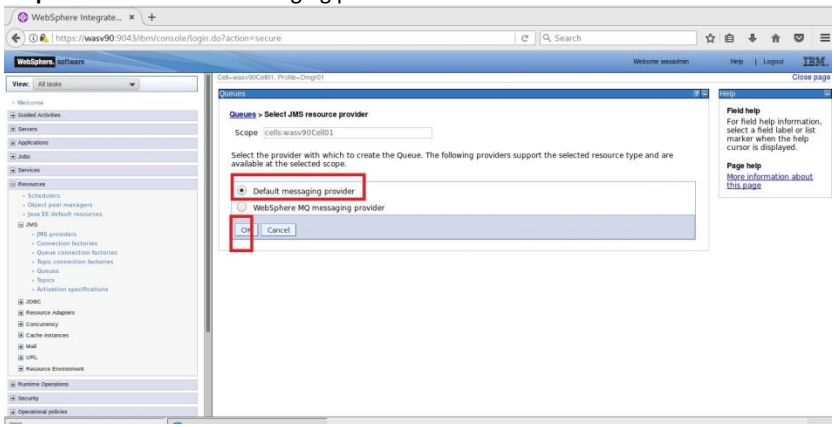


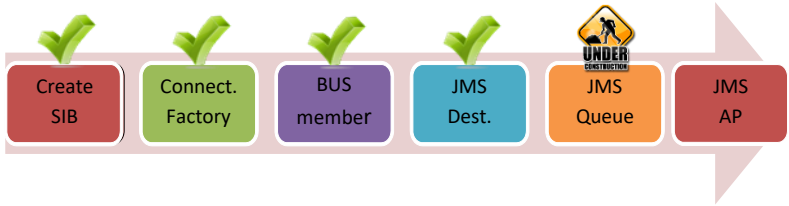


Step 2: Click “New” to start.



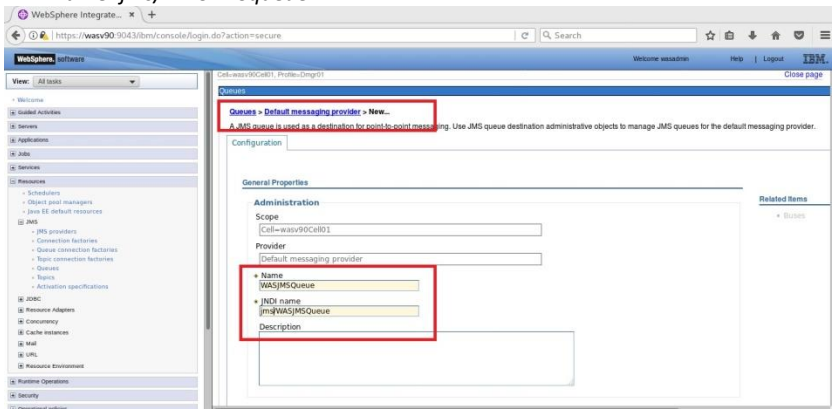
Step 3: Select “Default messaging provider” and then click “OK”.

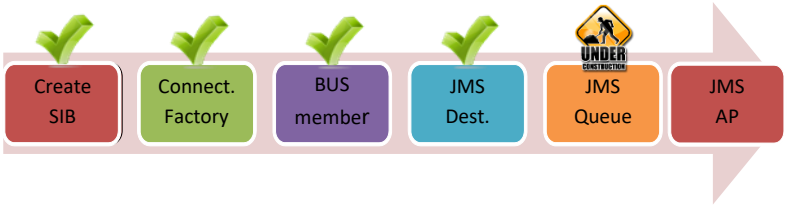




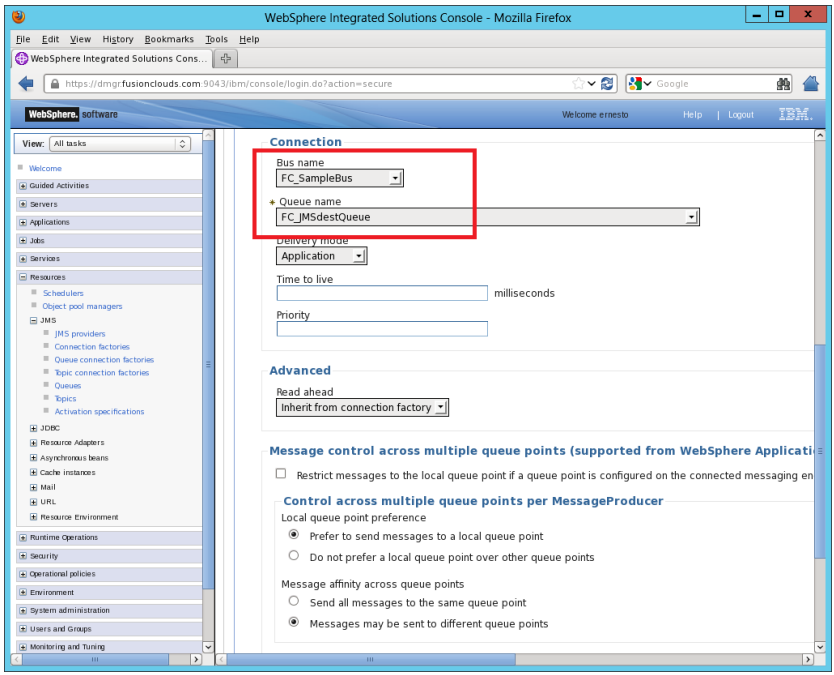
Step 4: Enter “Name” and “JNDI name”.

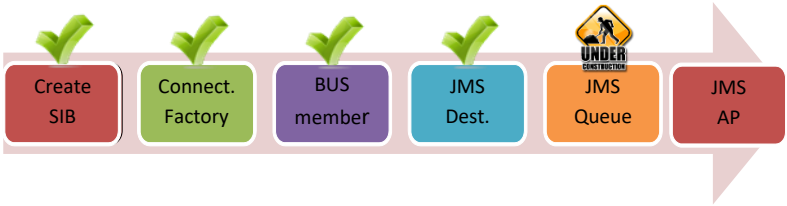
Name: WASJMSqueue
JNDI name: jms/WASJMSqueue



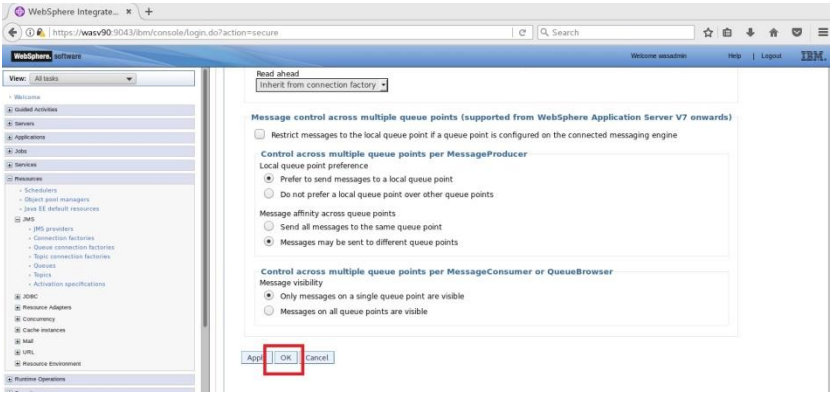


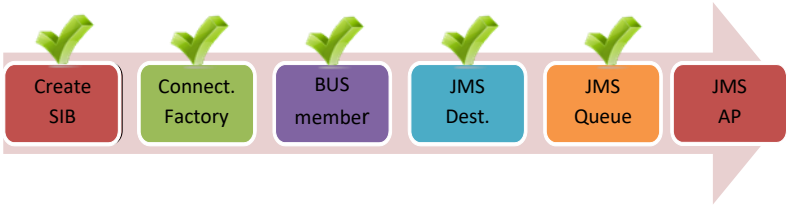
Select the “Bus name” and “Queue name” values we added in previous tasks.



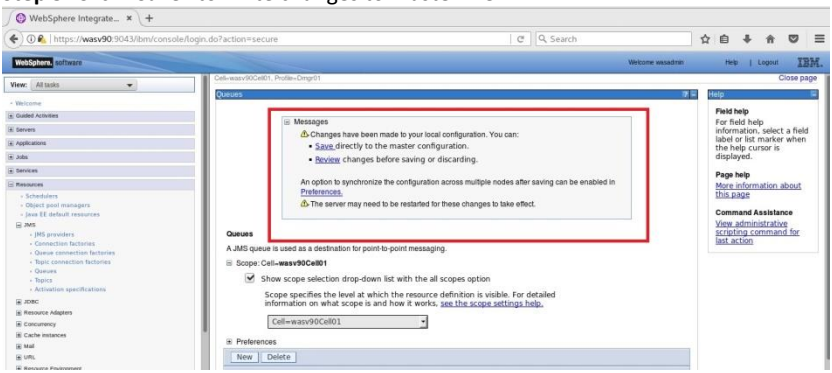


Click on “OK” to continue.

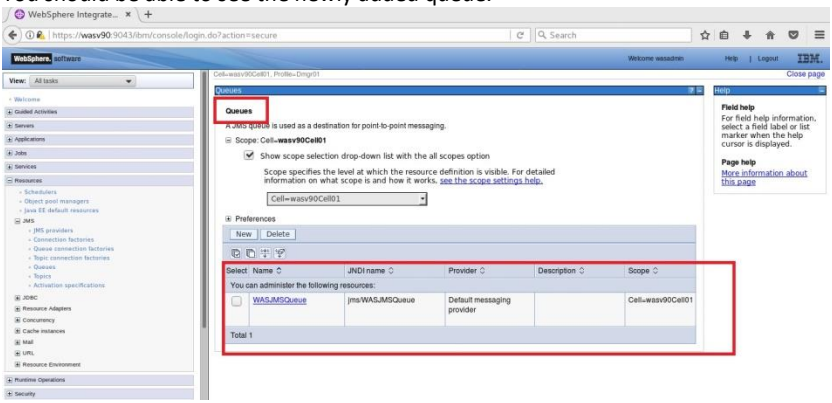




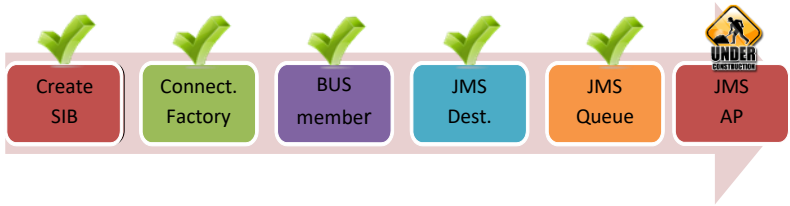
Step 5: Click “Save” to write changes to master file.



You should be able to see the newly added queue.

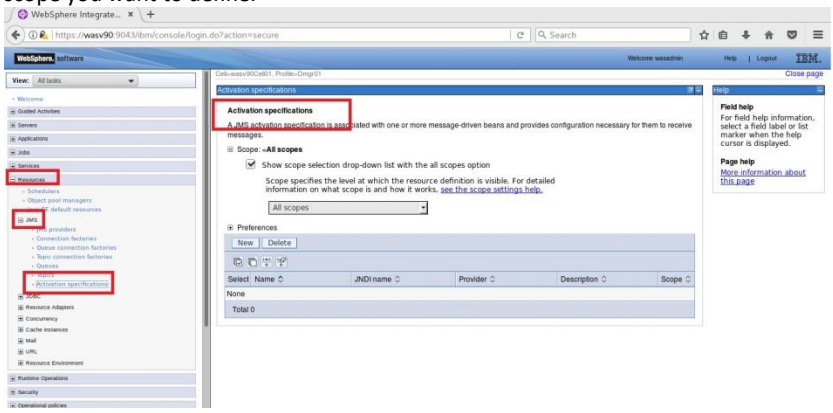


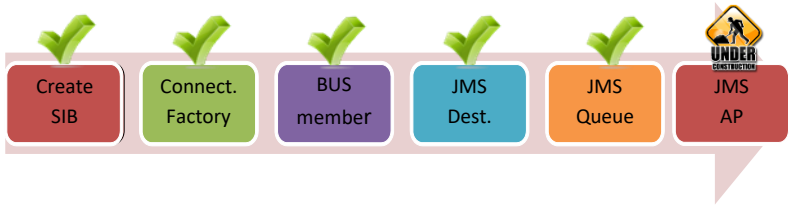
Task 5 is complete!



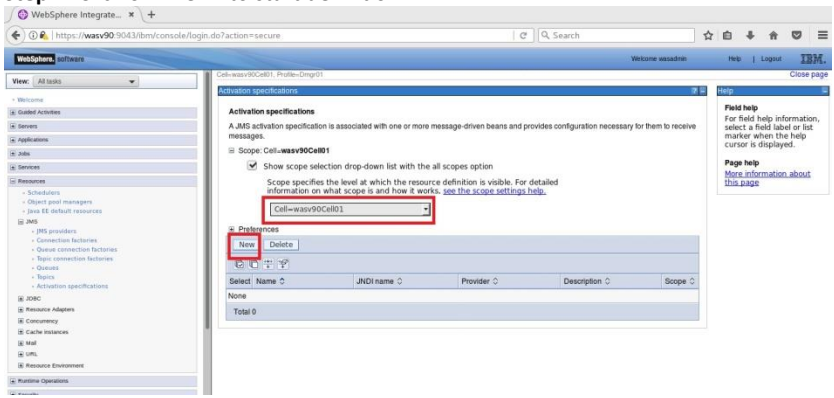
Task 6: Configure JMS Activation Specifications

Step 1: Navigate to “Resources>JMS>Activation specifications” and select your scope you want to define.

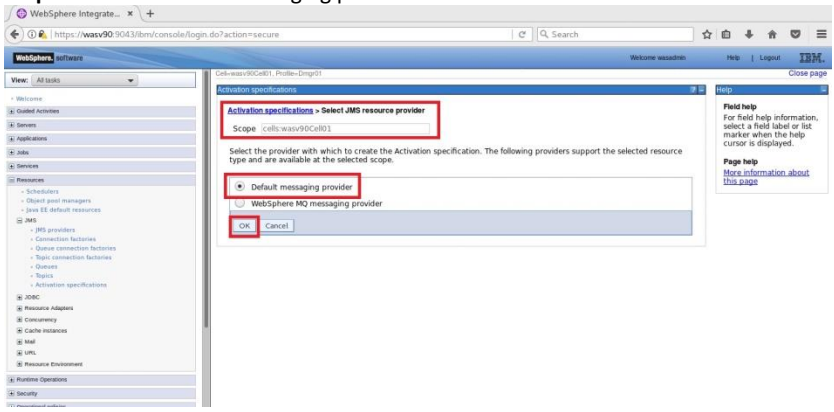


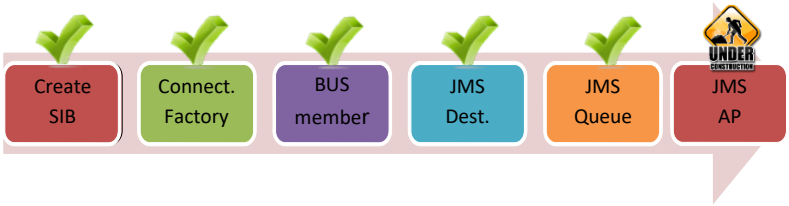


Step 2: Click on “New” to start definition.



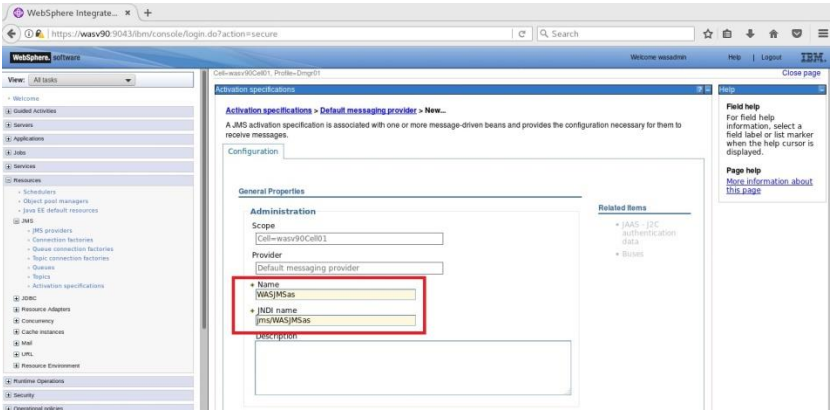
Step 3: Select “Default messaging provider” and click “OK”.

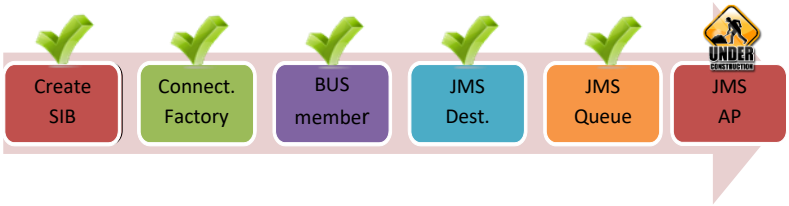




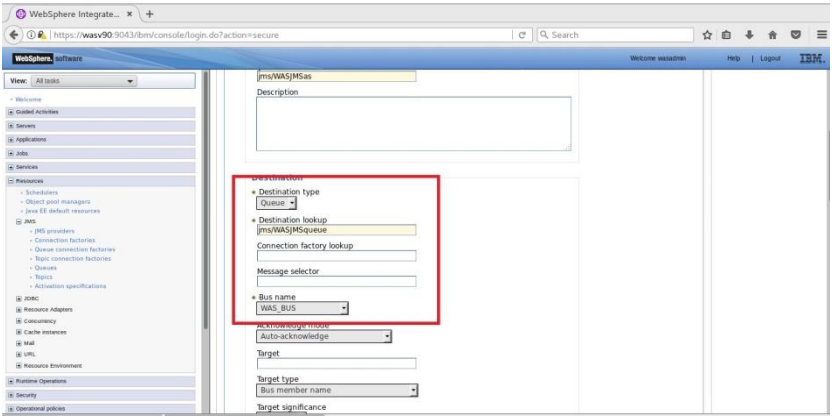
Step 4: Define “Name” and “JNDI name”.

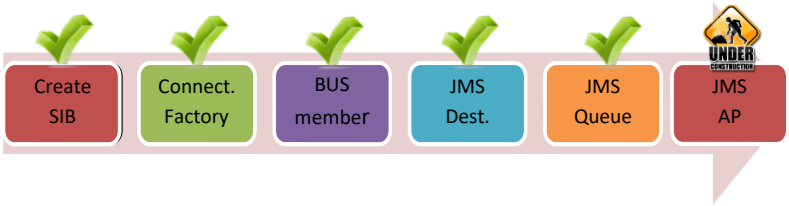
Name: WASJMSas
JDNI name: jms/WASJMSas



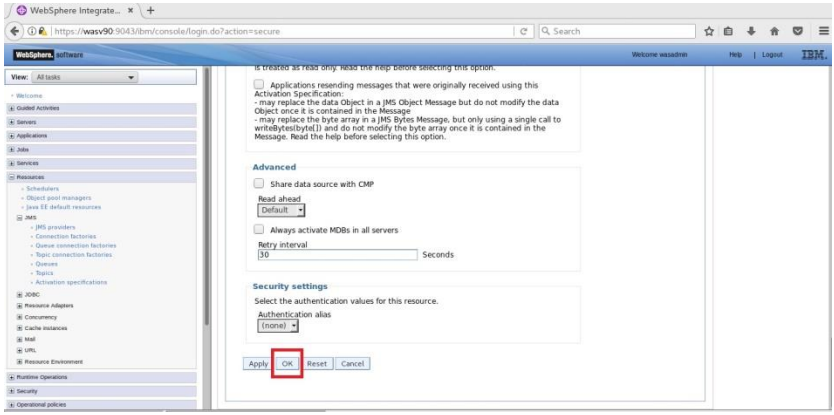


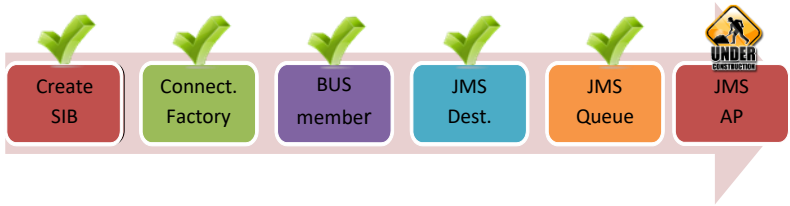
Select “Destination type” as “Queue”, for “Destination JNDI name” enter “jms/WASJMSqueue” and select “Bus name” as “WAS_Bus”.



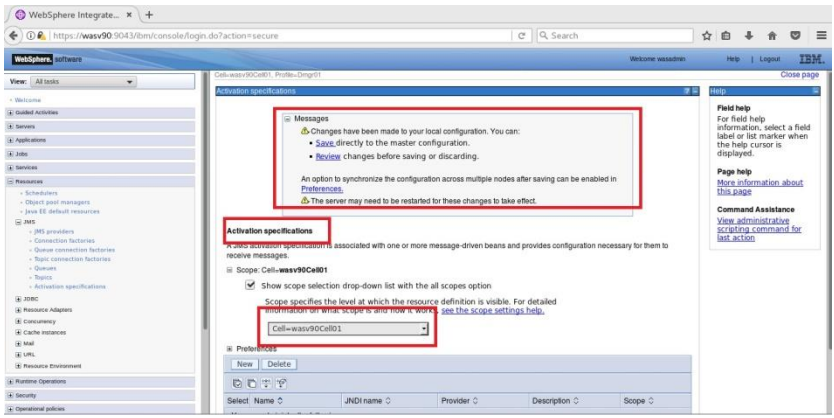


Scroll down and click “OK”.

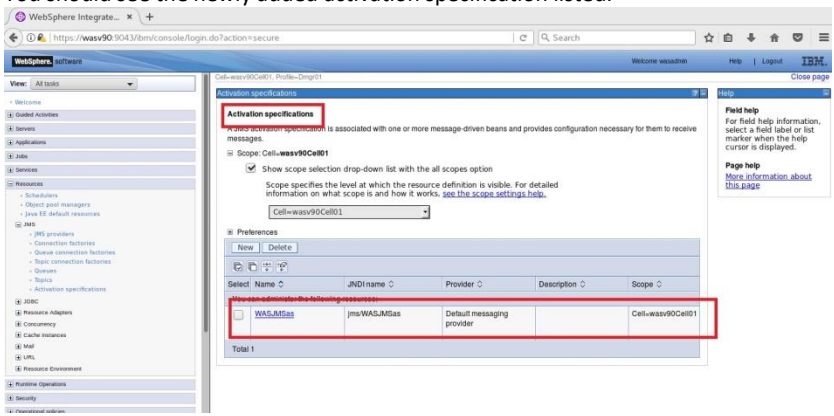




Step 5: Click on “Save” to write changes to the master configuration.



You should see the newly added activation specification listed.



Task 6 is complete!

SUMMARY

WebSphere Application Server supports asynchronous messaging that allows to create, send, receive and read requests between applications. For this communication, you need to configure different components of JMS depending on your application. WebSphere can work with default messaging provider, or with WebSphere MQ or with many other messaging software.

REFERENCES

- http://pic.dhe.ibm.com/infocenter/tivihelp/v50r1/index.jsp?topic=%2Fcom.ibm.mbs.doc%2Ffm_sag%2Fconfigsys%2Fc_jms_config_websphere.html
- http://pic.dhe.ibm.com/infocenter/prodconn/v1r0m0/index.jsp?topic=%2Fcom.ibm.scenarios.wmqwas101.doc%2Ftopics%2Fins_wascfg.htm

INDEX