What Is the Java Message Service?

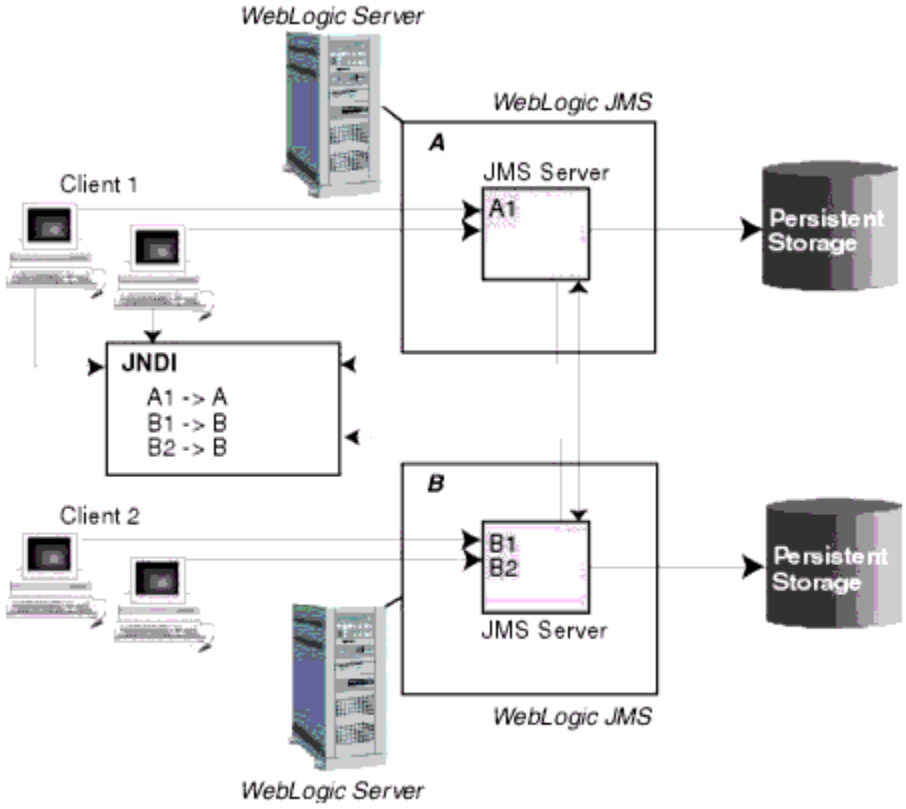
An enterprise messaging system enables applications to asynchronously communicate with one another through the exchange of messages.

A message is a request, report, and/or event that contains information needed to coordinate communication between different applications. A message provides a level of abstraction, allowing you to separate the details about the destination system from the application code.

The Java Message Service (JMS) is a standard API for accessing enterprise messaging systems that is implemented by industry messaging providers. Specifically, JMS:

- Enables Java applications that share a messaging system to exchange messages
- Simplifies application development by providing a standard interface for creating, sending, and receiving messages

WebLogic JMS Architecture and Environment



where: A1 and B1 are connection factories and B2 is a queue.

A JMS server is an environment-related configuration entity that acts as management container for JMS queue and topic resources defined within JMS modules that are targeted to specific that JMS server.

1. Introduction and Definitions

A JMS queue in Weblogic Server is associated with a number of additional resources:

JMS Server

A JMS server acts as a management container for resources within JMS modules. Some of its responsibilities include the maintenance of persistence and state of messages and subscribers. A JMS server is required in order to create a JMS module.

JMS Module

JMS modules are application-related definitions that are independent of the domain environment. You create and manage JMS resources either as *system modules* or as *application modules*.

JMS system modules are typically configured using the Administration Console or the WebLogic Scripting Tool (WLST), which adds a reference to the module in the domain's config.xml file.

WebLogic Administrators typically use the Administration Console or the WebLogic Scripting Tool (WLST) to create and deploy (target) JMS modules, and to configure the module's configuration resources, such as queues, and topics connection factories.

Subdeployment

JMS modules are targeted to one or more WLS instances or a cluster. Resources within a JMS module, such as queues and topics are also targeted to a JMS server or WLS server instances. A subdeployment is a grouping of targets. It is also known as advanced targeting.

Connection Factory

A connection factory is a resource that enables JMS clients to create connections to JMS destinations.

Connection factories are resources that enable JMS clients to create JMS connections. A connection factory supports concurrent use, enabling multiple threads to access the object

simultaneously. WebLogic JMS provides pre-configured default connection factories that can be enabled or disabled on a per-server basis, as described in

JMS Queue

A JMS queue (as opposed to a JMS topic) is a point-to-point destination type. A message is written to a specific queue or received from a specific queue.

Queue and Topic Destination Configuration

A JMS destination identifies a queue (point-to-point) or topic (publish/subscribe) resource within a JMS module.

Each queue and topic resource is targeted to a specific JMS server.

A JMS server's primary responsibility for its targeted destinations is to maintain information on what persistent store is used for any persistent messages that arrive on the destinations, and to maintain the states of durable subscribers created on the destinations.

A JMS queue defines a *point-to-point* destination type for a JMS server. A message delivered to a queue is distributed to a single consumer.

A JMS topic identifies a *publish/subscribe* destination type for a JMS server. Topics are used for asynchronous peer communications. A message delivered to a topic is distributed to all consumers that are subscribed to that topic.

Queue and Topic Targeting

Stand-alone queues and topics can only be deployed to a specific JMS server in a domain because they depend on the JMS servers they are targeted to for the management of persistent messages, durable subscribers, and message paging.

If you want to associate a group of queues and/or topics with a connection factory on a specific JMS server, you can target the destinations and connection factory to the same subdeployment, which links these resources to the JMS server targeted by the subdeployment. However, when standalone destinations are members of a subdeployment, a connection factory can only be targeted to the same JMS server.

Foreign Server Configuration

A foreign server resource enables you to reference third-party JMS providers within a local WebLogic Server JNDI tree.

With a foreign server resource, you can quickly map a foreign JMS provider so that its associated connection factories and destinations appear in the WebLogic JNDI tree as local JMS objects.

A foreign server resource can also be used to reference remote instances of WebLogic Server in another cluster or domain in the local WebLogic JNDI tree.

The objects used in this example are:

Object Name	Type	JNDI Name
TestJMSServer	JMS Server	
TestJMSModule	JMS Module	
TestSubDeployment	Subdeployment	
TestConnectionFactory	Connection Factory	jms/TestConnectionFactory
TestJMSQueue	JMS Queue	jms/TestJMSQueue
TestJMSTopic	JMS Topic	Jms/TestJMSTopic

2. Configuration Steps

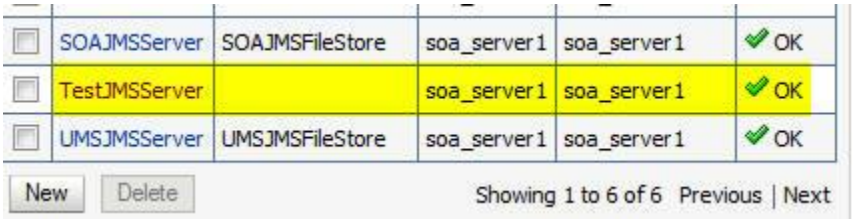
The following steps are done in the WebLogic Server Console, beginning with the left-hand navigation menu.

2.1 Create a JMS Server

1. Services > Messaging > JMS Servers.

1. Select New
2. Name: TestJMSServer
   Persistent Store: (none)
3. Target: soa_server1 (or choose an available server)
4. Finish

The JMS server should now be visible in the list with Health OK.



2.2 Create a JMS Module

1. Services > Messaging > JMS Modules

2. Select New
3. Name: TestJMSModule
   Leave the other options empty
4. Targets: soa_server1 (or choose the same one as the JMS server)
   Press Next
5. Leave “Would you like to add resources to this JMS system module” unchecked and press Finish.

2.3 Create a SubDeployment

A subdeployment is not necessary for the JMS queue to work, but it allows you to easily target subcomponents of the JMS module to a single target or group of targets. We will use the subdeployment in this example to target the following connection factory and JMS queue to the JMS server we created earlier.

1. Services > Messaging > JMS Modules
2. Select TestJMSModule
3. Select the Subdeployments tab and New
4. Subdeployment Name: TestSubdeployment
5. Press Next
6. Here you can select the target(s) for the subdeployment. You can choose either Servers (i.e. WebLogic managed servers, such as the soa_server1) or JMS Servers such as the JMS Server created earlier. As the purpose of our subdeployment in this example is to target a specific JMS server, we will choose the JMS Server option.
   Select the TestJMSServer created earlier
7. Press Finish

2.4 Create a Connection Factory

1. Services > Messaging > JMS Modules
2. Select TestJMSModule and press New
3. Select Connection Factory and Next
4. Name: TestConnectionFactory
   JNDI Name: jms/TestConnectionFactory
   Leave the other values at default
5. On the Targets page, select the Advanced Targeting button and select TestSubdeployment
6. Press Finish

The connection factory should be listed on the following page with TestSubdeployment and TestJMSServer as the target.

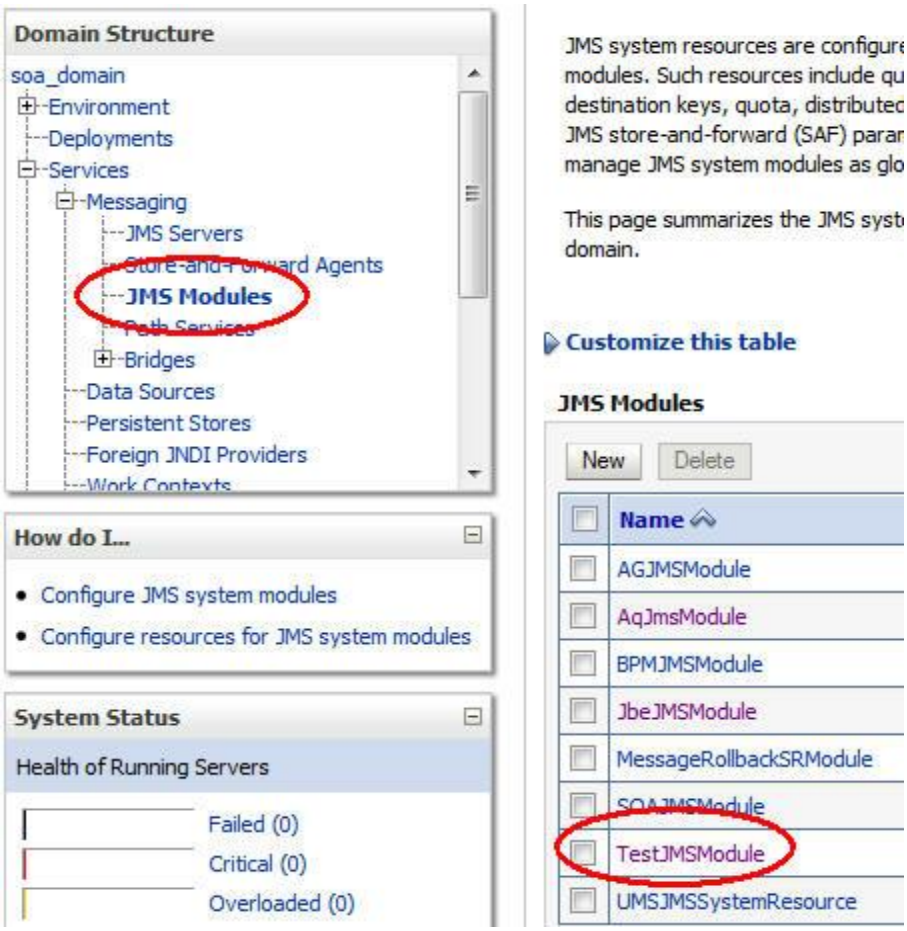
2.5 Create a JMS Queue

1. Services > Messaging > JMS Modules
2. Select TestJMSModule and press New

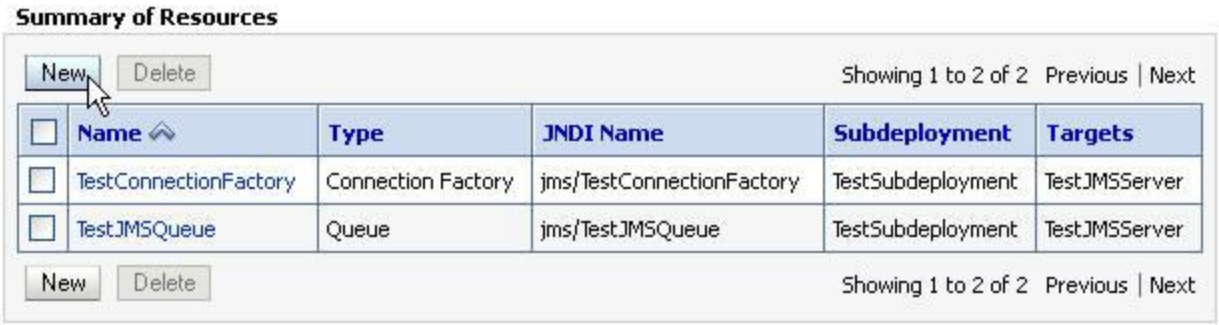
3. Select Queue and Next
4. Name: TestJMSQueue
   JNDI Name: jms/TestJMSQueue
   Template: None
   Press Next
5. Subdeployments: TestSubdeployment
6. Finish

The TestJMSQueue should be listed on the following page with TestSubdeployment and TestJMSServer.

Confirm the resources for the TestJMSModule. Using the Domain Structure tree, navigate to soa_domain > Services > Messaging > JMS Modules then select TestJMSModule



You should see the following resources



The JMS queue is now complete and can be accessed using the JNDI names

jms/TestConnectionFactory and
jms/TestJMSQueue.

In the following blog post in this series, I will show you how to write a message to this queue, using the WebLogic sample Java program QueueSend.java.