Working with the JMS Connector in WebSphere Cast Iron, Part 1: Configuring JMS in WebSphere Application Server

Mohan Siripi (rasiripi@in.ibm.com)
Software Engineer
IBM

Aparna Srinivasan (apasrini@in.ibm.com)
Software Engineer
IBM

13 May 2015

Part 1 of this 2-part tutorial series explains how to configure Java Messaging Service in WebSphere® Application Server when connecting from WebSphere Cast Iron Studio to the JMS Connector. This tutorial also addresses configuration issues when connecting to the JMS Connector.

Introduction

Part 1 of this tutorial series explains how to configure Java™ Messaging Service (JMS) to create the Queue Connection Factory in WebSphere Application Server (hereafter called Application Server) and connect to the JMS Connector in WebSphere® Cast Iron Studio and Appliance. You will learn to configure JMS to send messages between two or more clients in Application Server.

IBM® WebSphere Cast Iron is an offering from IBM that provides clients with a platform for integrating cloud-based applications from leading SaaS (Software as a Service) providers with on-premise applications from IBM and others. IBM Cast Iron Studio (herafter called Studio) is a development tool that is used to design, test, and publish integration projects to an IBM Cast Iron Integration Appliance. Studio provides numerous entities that you can drag into a workspace and configure as part of one, or more, business-process orchestrations that comprise an integration project.

WebSphere Application Server offers options for a faster, more flexible Java application server runtime environment with enhanced reliability and resiliency for building and running applications, including cloud and mobile. Application Server implements two main messaging sub-systems. The default messaging provider is internal to WebSphere and the WebSphere MQ messaging system.
Java Message Service (JMS) provides a common interface to standard messaging protocols and also to special messaging services in support of Java programs. Messages can involve the exchange of crucial data between systems and contain information such as event notification and service requests. Messaging is often used to coordinate programs in dissimilar systems or written in different programming languages. By using the JMS interface, you can invoke the messaging services like IBM's WebSphere MQ, formerly known as MQ Series, and other popular messaging products. In addition, JMS supports messages that contain serialized Java objects and messages that contain XML-based data.

Connectors provides the ability for the Cast Iron Appliance or Cast Iron Live to interact with the enterprise server and perform certain activities, just by configuration with no coding approach. Orchestrations are the main component of a Cast Iron integration solution. All functionalities in an integration solution are controlled by the orchestration. The orchestration contains a set of activities that are performed in an order that is defined within the orchestration editor or workspace.

Prerequisites

- Prior knowledge about WebSphere Cast Iron Studio and Appliance.
- Basic knowledge about Java Messaging Service and WebSphere Application Server.

System requirements

The following products and assets are required to configure and deploy the module:

- WebSphere Application Server 8.5.5 installed in a machine and a minimum of one WebSphere Application Server profile created
- WebSphere Application Server 8.5.5 JAR files listed in the Adding WebSphere Application Server JAR files section
- WebSphere Cast Iron Studio
- WebSphere Cast Iron Appliance

Configuring JMS to create the Queue Connection Factory on WebSphere Application Server

This section describes how you can create and configure the Queue Connection Factory in WebSphere Application Server to connect to the JMS Connector of WebSphere Cast Iron. Perform the following steps:

1. Log in to the Application Server admin console.
2. On the left-hand side menu, select Resources > JMS > Queue Connection Factory.
3. Create a new Queue Connection Factory. Make sure to select the Node=Node02, Server=server1 level as shown in Figure 1.
Figure 1. Creating a new Queue Connection Factory

4. Select the default messaging provider in the new window and click OK. This opens the window to configure the Queue Connection Property.
5. Provide a name and JNDI name under Administration.
6. Under Connection, select Target Type as the bus member name as shown in Figure 2. You need to create a bus and then provide it under "Bus name". See the steps in the Creating a new bus name section.

Figure 2. Configuring the Queue Connection Factory, Part 1
7. Select the connection proximity as **Server** and provide the quality of service as shown in Figure 3. Provide the details for the **Provider endpoints** as shown in Figure 3, where Port 7277 points to **SIB_ENDPOINT_ADDRESS** and 7287 points to **SIB_ENDPOINT_SECURE_ADDRESS**.

![Figure 3. Configuring the Queue Connection Factory, Part 2](image)

**Note:** If you need to check the correct port for a particular Application Server, go to **Server > Server Types > WebSphere Application Server**. Click on the server name and under "Communications", select the + symbol on **Ports** to display all the ports as shown in Figure 4. Make sure you select the correct port number.

![Figure 4. Ports used by WebSphere Application Server](image)

8. As shown in Figure 5, provide the security settings. For this, create the **JAAS - J2C authentication data** from the right-hand corner of the page from the "Related items" link.
Figure 5. Security settings

Creating a new bus name

1. Create a new bus and provide the value for the bus name under Administration of "Queue Connection Factory and Creating JAAS-J2C authentication data". Select Buses as shown in Figure 6.

Figure 6. Select the bus link

2. Create a new bus as shown in Figure 7 by clicking on the New button.

Figure 7. Creating a new bus

3. Provide a name for the bus as shown in Figure 8.
4. In Step 1.2 shown in Figure 9, uncheck the **Require clients use SSL protected transports** option. Proceed with the other options with no changes and save the bus that you created.

**Figure 9. Step 1.2: Providing the transport level security**

5. Make sure to disable the security by changing the **Enable bus security** option by clicking on the **Enable** option and disabling it as shown in Figure 10.

**Figure 10. Disabling bus security**

6. Go to **Resources > JMS > Queue connection factory**. Click on the new Queue Connection Factory name (for example, JMSMyQCF) and select the bus name as the newly created bus name. The drop-down displays the bus name.

7. Go to **Related links > JAAS - J2C authentication data** and create a new J2C authentication. Provide a name for it and the user ID and password of the admin console to be provided, as shown in Figure 11.
8. Navigate to Resources > JMS > Queue connection factory. Click on the new Queue Connection Factory name (for example, JMSMyQCF). Select the security settings as the same as the alias name created above. The drop-down displays the alias name that you created above. Select the first and third options as shown in Figure 5.

9. Change the settings in the admin console by selecting Security > Global security > RMI/IIOP Security. Now navigate to CSIv2 Transport Layer > Transport > SSL Supported (uncheck the same) from both the CSIv2 inbound communications and CSIv2 outbound communications under RMI/IIOP Security.

Adding WebSphere Application Server JAR files

You will need to add the Application Server JAR files in the following order. You can contact WebSphere Application Server Support for further help or details.

1. com.ibm.ws.messagingClient.jar
2. com.ibm.ws.webservices.thinclient_8.5.0.jar
3. j2ee.jar
4. bootstrap.jar
5. com.ibm.ws.runtime.jar
6. com.ibm.ws.sib.client.thin.jms_8.5.0.jar

Adding JAR files to WebSphere Cast Iron Studio

1. Open WebSphere Cast Iron Studio and navigate to Tools > Install Module Providers.
2. Add the above JAR files in the following order as shown in Figure 12.

Figure 12. Install module providers

As shown in Figure 13, you can create a new JMS endpoint by providing the JNDI Initial Factory, Provider URL (iiop://hostname:port), Factory Name, such as JMSQCF (as configured), and Provider Package if needed.
Once the JMS endpoint is configured, you can start working on the activities of the JMS Connector and also integrate with other endpoints of Cast Iron.

**Conclusion**

This tutorial explained how to configure JMS on WebSphere Application Server for use with the JMS Connector of WebSphere Cast Iron. This helped you configure JMS to send and receive messages through WebSphere Cast Iron. Part 2 (future tutorial) will explain how to work with the activities of the JMS Connector.

**Acknowledgments**

The authors would like to thank Sanjay M. Kesavan and Hariharan I. Subramanian for their help in reviewing this tutorial and providing valuable input.
Resources

Learn

- WebSphere Cast Iron Cloud Integration Knowledge Center
- WebSphere Cast Iron Cloud Integration product library
- WebSphere Cast Iron Cloud Integration product page
- WebSphere Cast Iron Cloud Integration support
- developerWorks WebSphere zone

Get products and technologies

- Trial: WeSphere Cast Iron Live

Discuss

- WebSphere Cast Iron Cloud Integration Community forums
About the authors

Mohan Siripi

Mohan R. Siripi is a Software Engineer at IBM India Software Labs. He is currently working on development and customer support for JCA-based resource adapters and WebSphere Cast Iron. He has been working for the past 10 years with various Java technologies in WebSphere Application Server, WebSphere Adapters, and WebSphere Cast Iron.

Aparna Srinivasan

Aparna Srinivasan is a Software Engineer and L3 Support Engineer for the WebSphere Cast Iron and WebSphere Adapters team. She has over 3 years of experience working with technologies such as Java and Rational Functional Tester. She has a Bachelor's Degree in Engineering from Anna University, Chennai, India.

© Copyright IBM Corporation 2015
Trademark
(www.ibm.com/developerworks/ibm/trademarks/)