Using the service integration bus link in WebSphere Application Server to route messages from a local queue to a remote queue

Manivannan Manickam

January 25, 2012

IBM® WebSphere® Application Server includes a variety of JMS providers that can be used by applications for asynchronous communication. By default, WebSphere Application Server uses a service integration bus (SI bus) for asynchronous communication. This article explains the communication between messaging engines running on different instances of WebSphere Application Server that will enable you to route a message from a local queue to a remote queue using the SI bus.

Introduction

Integration between remote messaging engines is very much required for message-based asynchronous communication between enterprise applications running on different application server instances. Message-based asynchronous communication is highly reliable, recommended, and very common in large systems, and so it is desirable to have the ability to route messages from a local queue to a remote queue for processing by an application on another system.

By default, IBM WebSphere Application Server uses a service integration bus (SI bus) for asynchronous communication. To configure SI bus communications successfully, you need to follow a series of steps and strict conventions to avoid errors and prevent disconnection. Because the information to help you do this can be scattered across many sources, it is possible (and perhaps likely) that you could miss some important points that could hamper your success. This article brings many of these details together to help you achieve message-based asynchronous communication between SI buses in separate WebSphere Application Server instances so that messages from a local queue can be routed to a remote queue.

This information applies to all WebSphere Application Server versions 6.0 through 8.0.

Fundamental elements of service integration

When enterprise applications running on different application servers need to communicate asynchronously, it is important to understand the cross-cell communication between the messaging engines running on each application server instance. WebSphere Application Server
provides built-in support to handle these messages in the form of the SI bus. To handle messages, the SI bus requires a messaging engine to manage the messaging resources for a bus member. A messaging engine gets created in the SI bus when a member for the bus is added. A member can be an application server or a cluster. Applications are connected to the messaging engine when accessing the SI bus.

**Service integration** provides asynchronous messaging services so that applications that need to produce and consume messages can exchange messages using destinations. Service integration is a complete JMS provider implementation.

A **service integration bus** is a group of one or more application servers or clusters that cooperate to provide asynchronous messaging services. Producers and consumers exchange messages using destinations, regardless of which messaging engine they are connected to. Different SI buses on remote machines can be connected.

Each bus member of an SI bus contains a **messaging engine** that can process messaging send and receive requests, and can host destinations. A messaging engine gets created in the SI bus when a member is added for the SI bus. A member can be an application server or a cluster. If the bus member is a cluster, it can have additional messaging engines to provide high availability and workload management.

To implement this basic scenario in WebSphere Application Server, there are several configuration updates required to JMS and the SI bus. To achieve cross-cell communication between messaging engines on different instances of WebSphere Application Server, a foreign bus and a service integration bus link will need to be created in each instance.

A **foreign destination** must be created in the local WebSphere Application Server instance so that messages placed in a local queue can be routed to a remote queue — via the foreign destination — using the default forward routing path configuration in the local queue. If the message placed in the remote queue needs to be routed to local queue, then a similar foreign destination configuration will be required in the remote instance.

This SI bus configuration requires specific naming conventions:

- The name of the foreign bus in the local instance must be the same as the SI bus in the remote WebSphere Application Server instance.
- The name of the SI bus link in both WebSphere Application Server instances must be the same.
- The name of the foreign destination in the local instance must match the destination name of an SI bus in the remote WebSphere Application Server instance.

The **service integration bus link** provides communication between a messaging engine in the local SI bus and a messaging engine in the remote SI bus. To configure a service integration bus link, you must have a foreign bus configured.

The sections that follow lay out the steps to complete a sample test configuration.
SI bus resources in the local application server instance

1. **Create SI bus.**
   In the WebSphere Application Server administrative console, expand **Service Integration** in the left column and create an SI bus called SenderBus (Figure 1).

   **Figure 1. Create SI bus**

   Because this sample configuration is for testing, the **Bus security** option is deselected, as security details are beyond the scope of this article. Click **Next**. In the follow up panel, click **Finish** to complete the configuration.

2. **Add a member to the SI bus.**
   Once the SI bus is created, you can add a member to it. Choose **Server** to add an application server as a member and apply the changes (Figure 2). When the application server is added as a member to the bus, a messaging engine gets created on the application server to handle all messages. Click **Next**. Leave the default options in the follow up panels and click **Finish** in the last panel to complete the configuration.

   **Figure 2. Add member to SI bus**

3. **Create a destination in the SI bus.**
   A destination must be created inside the SI bus to hold all the messages. There are four types of destinations:
   - **Queue**: Used for point-to-point messaging, which is what you will use in this scenario.
   - **Topic space**: Represents a set of publish and subscribe topics and is used for publish/subscribe messaging.
   - **Alias**: Maps an alternative name for a bus destination. Alias destination can be used for either point-to-point messaging or publish/subscribe messaging. Alias destination can be used to restrict the queue points that the producing and consuming applications use.
• **Foreign**: Represents a destination that is defined in a foreign bus. You can use a foreign destination for point-to-point messaging.

Select **Queue** and click **Next** (Figure 3). Name the destination **SenderQueue** (Figure 4).

**Figure 3. Create a destination**

![Create a destination](image)

**Figure 4. Name the new destination**

![Name the new destination](image)

Click **Next**. Leave the default option in the follow up panel and click **Finish** in the last panel to complete the configuration.

4. **Create a foreign bus connection and service integration bus link.**

The name of the foreign bus must be the same name as the SI bus in the remote WebSphere Application Server instance. Continuing through the Foreign Bus panels, enter or select values for these properties (Figures 5 through 7):

- Bus connection type: **Direct connection**
- Foreign bus type: **Service integration bus**
- Foreign bus name: **ReceiverBus**

**For WebSphere Application Server V7.0 and later**

The properties listed next are specified during creation of the foreign bus connection, which in turn creates the necessary service integration bus link. In earlier versions of WebSphere Application Server, these properties are specified during service integration bus link creation. The name of the service integration bus link in both instances must be the same. Values for the remaining properties are obtained from the SI bus of the remote WebSphere Application Server instance:

- Gateway messaging engine in the foreign bus: **Node01.server1-ReceiverBus**
- Service integration bus link name: **Buslink**
• Target inbound transport chain: **InboundBasicMessaging**
• Bootstrap service integration bus provider endpoints: host2:7277:BootstrapBasicMessaging

Use the format `hostname:portNumber:chainName` for specifying bootstrap endpoint. The port number used is the `SIB_ENDPOINT_ADDRESS` of the remote messaging engine. If SI bus security is enabled, use `SIB_ENDPOINT_SECURE_ADDRESS` port number and use `chainName` as `BootstrapSecureMessaging`. This property is set in the same way as the provider endpoint property in the JMS connection factory settings.

**Figure 5. Define bus connection type**

**Figure 6. Define foreign bus type**

**Figure 7. Name foreign bus**
Click **Next** to proceed to summary page and click **Finish** to complete the configuration. You can verify whether the service integration bus link is created using the admin console by navigating to the messaging engine, as shown in Figure 8.

**Figure 8. Verify new SI bus link**

![Figure 8. Verify new SI bus link](image)

When you find the service integration bus link, **Buslink**, select the link to display the values of the properties specified during the foreign bus connection creation.

**For WebSphere Application Server V6.0 and V6.1**

Select the messaging engine and create a service integration bus link from the **Additional properties** panel (Figure 9). The name of the service integration bus link in both instances must be the same. When creating the service integration bus link, enter or select values for these properties:

- **Name**: Buslink
- **Foreign bus name**: ReceiverBus
- **Remote messaging engine name**: Node01.server1-ReceiverBus
- **Target inbound transport chain**: InboundBasicMessaging
- **Bootstrap endpoints**: host2:7277:BootstrapBasicMessaging

**Figure 9. Additional properties panel for messaging engine**

![Figure 9. Additional properties panel for messaging engine](image)

5. Create a foreign destination in the service integration bus
Create a destination of type **Foreign**. The name of the foreign destination must be the same as the queue destination in the SI bus of remote instance. Enter values for these properties (Figure 10):

- **Identifier**: ReceiverQueue
- **Bus**: ReceiverBus

**Figure 10. Create a foreign destination**

![Create a foreign destination](image)

**SI bus resources in the remote application server instance**

Several of these steps are similar to those listed above for setting up the local SI bus. Refer to the earlier steps for further detail.

1. **Create SI bus.**
   In the WebSphere Application Server admin console under **Service integration**, create a SI bus called **ReceiverBus**. As before, make sure **Bus security** is deselected. Continue with the next steps.

2. **Add a member to SI bus.**
   Add an application server as a member of the bus. A messaging engine is created on the application server to handle all messages.

3. **Create a destination in the SI bus.**
   Again, a destination is needed inside an SI Bus to hold all the messages. Create a destination of type **Queue** and name the destination as **ReceiverQueue**.

4. **Create a foreign bus connection and service integration bus link**
   The foreign bus name must be the same as the SI bus in the remote WebSphere Application Server instance. Enter or select the following properties:
   - **Bus connection type**: Direct connection
   - **Foreign bus type**: Service integration bus
   - **Foreign bus name**: SenderBus

   **For WebSphere Application Server V7.0 and later**
   Enter or select the values for these properties, which would be obtained from the SI bus of the remote WebSphere Application Server:
   - **Gateway messaging engine in the foreign bus**: Node01.server1-SenderBus
• Service integration bus link name: Buslink
• Target inbound transport chain: InboundBasicMessaging
• Bootstrap service integration bus provider endpoints: host1:7276:BootstrapBasicMessaging

For WebSphere Application Server V6.0 and V6.1
Select the messaging engine and create a service integration bus link available under the Additional properties section. The name of the service integration bus link in both instances must be the same. Enter or select values for these properties:
• Name: Buslink
• Foreign bus name: SenderBus
• Remote messaging engine name: Node01.server1-SenderBus
• Target inbound transport chain: InboundBasicMessaging
• Bootstrap endpoints: host1:7276:BootstrapBasicMessaging

JMS administered objects in the local application server instance

All of the information discussed to this point has been related to configuring the SI bus in both local and remote instances of WebSphere Application Server. Next, you need to describe the JMS-related configuration that is required for WebSphere Application Server to send a JMS message programmatically.

To implement this, administered objects must be created in the local WebSphere Application Server instance. There are two JMS administered objects available, both of which you will create in the steps below:

• Connection factory
• Destination

1. Create JMS queue connection factory.
A connection factory is used to create connections to the JMS provider of the JMS destination. It is used for point-to-point and publish/subscribe messaging. The example used here describes how to configure point-to-point messaging (publish/subscribe messaging configuration is similar).

Create a JMS queue connection factory and enter or select values for these properties:
• JMS resource provider: Default Messaging Provider
• Name: testcf
• JNDI Name: jms/testcf
• Bus Name: SenderBus
• Provider endpoints: host1:7276:BootstrapBasicMessaging

Default Messaging Provider is selected because it uses the SI bus for asynchronous communication (Figure 11).
Using the service integration bus link in WebSphere Application Server to route messages from a local queue to a remote queue

Figure 11. Create JMS queue connection factory

Click OK. In the next panel, provide values for the remaining properties mentioned above and click Apply to complete the configuration.

2. Create JMS queue.

Create a JMS Queue destination and enter or select values for these properties:

- JMS resource provider: Default Messaging Provider
- Name: testqueue
- JNDI Name: jms/testqueue
- Bus name: SenderBus
- Queue name: SenderQueue

Figure 12. Create JMS queue

Click OK. In the next panel, provide values for the remaining properties mentioned above and click Apply to complete the configuration.

After configuring the objects above, running the Java™ client program sends a message to the SenderQueue destination inside SenderBus by performing the JNDI lookup to the JMS connection factory and JMS queue destination. Instead of using a standalone Java client, you can also deploy an enterprise application (servlet or JSP) to send the message to the SenderQueue.

As per the example scenario presented in this article, you need to route a message to the remote queue as soon as a message reaches the SenderQueue. To route a message from the SenderQueue to a remote queue, additional configuration is still necessary.
3. Edit the SenderQueue by navigating to **Buses > SenderBus > Destinations > SenderQueue** in the admin console (Figure 13).

   The Default forward routing path, which appears at the bottom of the panel, is used to forward messages to one or more destinations. The value must be in the format “Foreign-Bus-Name: Foreign-Destination-Name,” and so enter **ReceiverBus:ReceiverQueue** as the value, where “ReceiverBus” is the name of the foreign bus and “ReceiverQueue” is the name of the foreign destination in the local WebSphere Application Server instance.

   **Figure 13. Edit routing path**

   ![Edit routing path](image)

   You must restart the servers in both local and remote instances for all configuration changes to go effect. After restarting the servers, ensure that service integration bus link in both instances is running. This can be verified by checking the status of service integration bus link in the admin console for each instance and make sure the status is "Started." The “Started” status means the service integration bus link is started on the local messaging engine and has an active connection to the foreign bus.

   A sample Java client program for sending a simple JMS text message is included with this article for download. This program sends a message to the SenderQueue in the local WebSphere Application Server instance, which in turn routes the message to the ReceiverQueue in the remote instance.

   You can verify the successful transition of the message, as well as successful cross-cell messaging configurations, by checking the availability of the message in the ReceiverQueue in remote WebSphere Application Server instance. To do this:

   1. Navigate to **Buses > ReceiverBus > Messaging engines > Node01.server1-ReceiverBus > Queue points > ReceiverQueue@Node01.server1-ReceiverBus**.
   2. Select the **Runtime** tab for the option to view the message in the console.

   In an actual production scenario, you would more likely have a message driven bean (MDB) enterprise application deployed in the remote WebSphere Application Server admin console, which would be mapped to the ReceiverQueue using the activation specification configuration. As soon as a message reaches the ReceiverQueue, the application would consume and process the message using the onMessage method. Once the message is consumed, it is removed from the ReceiverQueue.
Conclusion

This article described the concept of message-based asynchronous communication using the service integration bus link available in WebSphere Application Server, compiling details regarding the configuration of service integration bus resources, JMS resources, and critical naming conventions to support the use of this feature.
# Downloadable resources

<table>
<thead>
<tr>
<th>Description</th>
<th>Name</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample client program</td>
<td>JMSClient.zip</td>
<td>1 KB</td>
</tr>
</tbody>
</table>
Related topics

- Service integration
- Java Message Service (JMS) SDK and specification
- Choosing a messaging system: WebSphere MQ vs. the WebSphere Application Server Service Integration Bus
- WebSphere Application Server Information Center

© Copyright IBM Corporation 2012
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)