Using the JNBridge JMS Adapter for BizTalk Server with ActiveMQ

Version 4.0

www.jnbridge.com

JNBridge, LLC
www.jnbridge.com

Contents

Quick Config for ActiveMQ

Adapter Transport Handler Properties

- Initial Context Factory: `org.apache.activemq.jndi.ActiveMQInitialContextFactory`
- JMS Scheme: `tcp`
- Queue Connection Factory: `ConnectionFactory`
- Topic Connection Factory: `ConnectionFactory`
- Class Path:

  For ActiveMQ versions 5.1-5.4, use the JAR file:

  ```
  activemq-all-x.x.x.jar
  ```

  Where x.x.x refers to the ActiveMQ version, e.g. 5.1.0. For version 5.5, use these JAR files:

  ```
  activemq-all-5.5.0.jar
  .../lib/optional/slf4j-log4j12-1.5.11.jar
  .../lib/optional/log4j-1.2.14.jar
  ```

  For all subsequent releases, use the the *activemq-all-x.x.x.jar* file.

- JVM Path (examples): `C:\Program Files\Java\jre7\bin\client\jvm.dll`

Adapter Send or Receive Port Properties

- Port Number: `61616`
- JMS Object Name

  `dynamicQueues/exampleQueue`

  `dynamicTopics/exampleTopic`

Using the JMS Adapter with ActiveMQ

This document uses the default JMS broker pre-configured in ActiveMQ. This document assumes some passing familiarity with enterprise Java such as JNDI contexts, factories and general connection requirements and parameters. This document also assumes that the BizTalk developer has access to particular information peculiar to the target JMS implementation. Finally, this document assumes that the reader is knowledgeable and experienced with BizTalk Server.

This document assumes that ActiveMQ is stand-alone rather than deployed to a JEE application server as the messaging provider using a JCA connector. If this is the case, then refer to the ActiveMQ documentation and the JEE app server documentation.

This document only discusses those property values in the adapter transport handlers and location handlers that pertain to communicating with ActiveMQ. Other properties that are not discussed here can be found in the companion *Using the JNBridge JMS Adapter for BizTalk Server* document.

Resources

- The user guide, *Using the JNBridge JMS Adapter for BizTalk Server.*
- Chances are, if the target ActiveMQ implementation is mature, the values for the configuration of BizTalk transport handlers and send/receive ports can be supplied by the ActiveMQ administrator, gleaned from existing JMS client code or property files, e.g. *jndi.properties*.
- If the ActiveMQ implementation targeted is not configured, then the default example JMS broker can be used for proof-of-concept evaluations. This document uses the default JMS broker, default JNDI implementation and the “dynamicQueues” context in ActiveMQ.
- This link is Apache documentation providing information about the ActiveMQ JNDI implementation. This document uses the JNDI implementation provided by ActiveMQ. For more information, see the section *JNDI Names*.

BizTalk Machine Prerequisites

The following prerequisites are needed for the adapter.

- A Java Run-time Environment (JRE) must be installed on the target machine. The JNBridge JMS Adapter supports the Standard Edition JRE 7 or above.
- The JNBridge JMS Adapter for BizTalk uses the stand-alone JMS environment supplied by ActiveMQ. This environment consists of one or more JAR files. This jar files are usually found under the root directory where ActiveMQ is installed. Please see the section, *Class Path*, below.

Deploy JMS Header Schema to BizTalk Application

In order to properly handle JMS header properties within BizTalk, you must deploy the assembly, `JNBridgeBTS2006JMSProperties.dll`, to your BizTalk application. This assembly contains the XSD namespaces and schemas used by the JNBridge JMS Adapter to promote JMS header properties within messages stored in the BizTalk Message Box.

! ***Deploying this assembly is mandatory.***

- To deploy the schema assembly

1 Open up the BizTalk Administrator and open your BizTalk application in the left-side tree view.

2 Right click on the application's root node and choose Add ▶ Resources. This opens the Add Resources dialog.

3 In the dialog, click on the Add button and navigate to the schema DLL in the adapter install directory, e.g. `C:\Program Files\JNBridge\JMSAdapters\BTS2006\bin\JNBridgeBTS2006JMSProperties.dll`.

4 Click on OK to close the Add Resources dialog.

5 Open the Schemas folder in your application. You should see the three schemas: `JMSSendProperty.SendPropertySchema`, `JMSRecvProperty.RecvPropertySchema` and `JMSConfProperty.ConfPropertySchema`.

6 Restart the host instance and application.

Configuring the Adapter Send and Receive Transport Handlers

The transport handler property grids for the Send and Receive sides contain properties global to all send or receive ports configured to use the JNBridge JMS Adapter and that reside in the same BTS host instance. In other words, all JMS Adapter send or receive ports in the BTS host instance will inherit these transport properties. You must configure send handler transport properties in order to produce messages to queues and topics. Likewise, you must configure receive handler transport properties in order to consume messages from queues and topics. In most cases, the values of the properties will be identical between the send and receive handlers; however, depending on the JMS server implementation, they may be different.

JMS Properties Category

The JMS Properties category are properties used to properly connect to a JMS server.

- JMS Acknowledge Mode

  The Acknowledge Mode is a drop-down list containing the JMS specification that determines how a JMS client and server institute a reliable messaging protocol. The choices are AUTO_

ACKNOWLEDGE, CLIENT_ACKNOWLEDGE and DUPS_OK_ACKNOWLEDGE. Regardless of the choice, the JNBridge JMS Adapter will correctly implement the protocol.

For ActiveMQ, **`AUTO_ACKNOWLEDGE`** is the default configuration.

- Initial Context Factory

  This is a text-editable field containing the name of the JNDI initial context factory. The initial context factory is a JNDI class used to locate and instance factories and JMS destinations. The default initial context factory for the ActiveMQ JNDI implementation:

  ```
  org.apache.activemq.jndi.ActiveMQInitialContextFactory
  ```

  If ActiveMQ is used as a messaging provider with another J2EE app server, then the JNDI initial context class may be different.

! ***Factory names are case sensitive—be sure the name (including the complete namespace, if necessary) is typed correctly.***

- JMS Scheme

  This is a text-editable field. The JMS Scheme or *Protocol* is particular to each vendor's implementation. The protocol is part of the URI used to connect to the JMS service.

  For ActiveMQ, the default scheme is:

  ```
  tcp
  ```

  When using ActiveMQ, the tcp protocol is referred to as a *transport*.

- JMS Version

  This property tells the adapter which JMS implementation to expect when it loads the vendor's client stack—the JAR files in the Class Path property. The drop-down list contains two choices, 'JMS 1.1' and 'JMS 2.0'.

- JMS Security Mode

  The JMS Security Mode is a drop-down list that specifies the type of security required by the JMS server implementation. The choices are none, simple and strong. If the choice is simple, then the server expects a user name and password.

! ***If the JMS server implements simple security, it is not necessary to configure this property. Enter a user name and password—the JNBridge JMS Adapter will automatically switch to the simple security mode.***

- Queue Connection Factory

  This is a text-editable field. The default queue connection factory in ActiveMQ is:

  ```
  ConnectionFactory
  ```

- Topic Connection Factory

  This is a text-editable field. The default topic connection factory in ActiveMQ is:

  ```
  ConnectionFactory
  ```

  ActiveMQ uses a single connection factory for both queues and topics.

JNBridge Properties Category

The JNBridge Properties Category correctly configure the .NET-to-Java interoperability core components.

- Class Path

  The Java class path is a set of semicolon-separated paths to the JAR or class files required for a JMS client installation. The class path is used by the JNBridge Java and .NET interoperability components to locate the JMS and JNDI client Java classes so they can be instanced in the Java Virtual Machine.

  To edit the class path, click in the field to enable the browse button. Click on the button to launch the Edit Class Path dialog. Note that only checked elements will be added to the class path when the dialog is dismissed.

  For ActiveMQ versions 5.1.x-5.4.x, use the JAR file:

  ```
  activemq-all-x.x.x.jar
  ```

  Where x.x.x refers to the ActiveMQ version, e.g. 5.1.0.

  For version 5.5.x, use these JAR files:

  ```
  activemq-all-5.5.0.jar
  .../lib/optional/slf4j-log4j12-1.5.11.jar
  .../lib/optional/log4j-1.2.14.jar
  ```

  For all subsequent releases, use the the *activemq-all-x.x.x.jar* file.

- JVM Path

  The JVM Path property is the absolute path to the Java Virtual Machine implementation,

jvm.dll. To edit the JVM Path property, click in the field to enable the browse button. Click on the button to launch the standard File Open dialog. Navigate to `jvm.dll` and click OK. In the example shown, the JVM used is:

```
C:\Program Files\Java\jre7\bin\client\jvm.dll
```

Security Properties Category

This category provides security credentials necessary to connect to a JMS server, if the JMS implementation supports security mode simple.

- Password

  Click in this field to drop-down the password edit field. Type in the password.

- User Name

  This is a text editable field. Enter the user name necessary to connect to the JMS server.

Configuring Send Ports and Receive Locations

Configuring Send Ports

Connection Properties Category

These properties determine where the JMS server resides and the port number where it is listening for connections.

- Host Name

  This a text-editable field. Enter the name or IP address of the machine hosting the JMS server.

- Port Number

  This is a text-editable field. Enter the port number on which the JMS server is accepting connections. For ActiveMQ configured for the tcp transport, this port is usually, by default, `61616`.

- Proprietary Connection String

  This is a text-editable field. This property is only used if the JMS implementation uses complex URLs containing query expressions, or some proprietary connection string. For example, ActiveMQ, supports a simple URL connection string, tcp://medtner:61616, that can be constructed from the Host Name and Port Number properties. However, if connection and protocol properties must be set, ActiveMQ supports URLs with query expressions:

  ```
  failover:( tcp://scriabin:61616?wireFormat.maxInactivityDuration=30000,
  tcp://elgar:61616?wireFormat.maxInactivityDuration=30000,
  tcp://cage:61616?wireFormat.maxInactivityDuration=30000
  )?randomize=false
  ```

  If this property contains a value, then the Host Name and Port Number properties will be ignored.

JMS Operation Properties Category

These properties determine what operation the send port will enable.

- JMS Object Name

  This is a text-editable field. Enter the JNDI name bound to the JMS queue or topic. For ActiveMQ, the dynamicQueues context is used allowing lazy dynamic creation of queues.

  ```
  dynamicQueues/myQueue
  ```

  ```
  dynamicTopics/myTopic
  ```

- JMS Object Type

  This is a drop-down list containing two values: Queue or Topic.

- Message Type

  This is a drop-down list containing the values: Text, Text UTF, Text ISO-8859-15 or Bytes. If Text is chosen, then the JNBridge JMS Adapter will send a JMS Text Message. Because a JMS Text Message is by definition UTF-16 BE, the types Text UTF and Text ISO-8859-15 ensure that the string which is the payload of the JMS message is correctly encoded from the binary representation in the BizTalk Message Box. If the type Text is chosen, then the binary representation is considered UTF-8. If Bytes is chosen, then the JNBridge JMS Adapter will send a JMS Bytes Message.

Configuring Receive Ports and Locations

Connection Properties Category

These properties determine where the JMS server resides and the port number where it is listening for connections.

- Host Name

  This a text-editable field. Enter the name or IP address of the machine hosting the JMS server.

- Port Number

  This is a text-editable field. Enter the port number on which the JMS server is accepting connections. For ActiveMQ configured for the tcp transport, this port is usually, by default, **61616**.

- Proprietary Connection String

  This is a text-editable field. This property is only used if the JMS implementation uses complex URLs containing query expressions, or some proprietary connection string. For example, ActiveMQ, supports a simple URL connection string, tcp://medtner:61616, that can be constructed from the Host Name and Port Number properties. However, if connection and protocol properties must be set, ActiveMQ supports URLs with query expressions:

  ```
  failover:( tcp://scriabin:61616?wireFormat.maxInactivityDuration=30000,
  tcp://elgar:61616?wireFormat.maxInactivityDuration=30000,
  tcp://cage:61616?wireFormat.maxInactivityDuration=30000
  )?randomize=false
  ```

  If this property contains a value, then the Host Name and Port Number properties will be ignored.

JMS Operation Properties Category

These properties determine what operation type of operation the send port will enable.

- JMS Object Name

  This is a text-editable field. Enter the JNDI name bound to the JMS queue or topic. For ActiveMQ, the dynamicTopics context is used allowing lazy dynamic creation of topics.

  ```
  dynamicQueues/myQueue
  dynamicTopics/myTopic
  ```

- JMS Object Type

  This is a drop-down list containing two values: Queue, Topic or SharedTopic.

- Message Type

  This is a drop-down list containing the values: Text, Text UTF-8, Text UTF-16, Text ISO-8859-15, Bytes or Map. If Text is chosen, then the JNBridge JMS Adapter expects to receive a JMS Text Message. If Bytes is chosen, then the JNBridge JMS Adapter expects to receive a JMS Bytes Message. If Map is chosen, then the JNBridge JMS Adapter expects to receive a JMS Map Message.

  Because a JMS Text Message by definition contains UTF-16 BE text, the types Text UTF-8, Text UTF-16 and Text ISO-8859-15 ensure that the text in the body of the message is encoded correctly to binary for submittal to the BizTalk Message Box. The type Text without a qualifier means UTF-8.

  - Client ID

  This is a unique string that identifies the receive port connection to ActiveMQ. It is only used if durable subscriptions are enabled.

- Durable Subscription Name

  Durable subscriptions are particular to topics only. A durable subscription for a topic allows consumers to register a name with the JMS server such that whenever a receive port is active, all messages in the topic will be received. In this way, a receive port does not have to be continually connected to receive messages from a topic. A receive port that does not use durable subscriptions must be active and connected in order to subscribe to a topic—any messages published by the topic while a nondurable receive port is not active will not be available to that receive port when it becomes active. This is a text-editable field. Enter the durable subscription name.

- Message Selector Filter

  Message selectors are used by receive ports to filter or select messages from topics and queues based on JMS and custom message header properties.

  This is a text-editable field. Enter in a selector expression. The expression is derived from a subset of the SQL92 standard.

JNDI Names

The ActiveMQ JNDI implementation is a simple initial context factory configured using a *jndi.properties* file, figure 1. If queues and topics are dynamic, then the jndi.properties file does not need to be used. However, the JNDI name must be of the form:

```
dynamicQueues/[a queue name]
dynamicTopics/[a topic name]
```

If the jndi.properties file is used, it must be included in a jar file and added to the Class Path property. The JNDI name used in the JMS Object Name property does not need to be preceded by "queue." or "topic.". Please go to ActiveMQ JNDI, for more information regarding the simple JNDI implementation in ActiveMQ.

To create a jar file called *jndiprops.jar* that contains a jndi.properties file , use the following command:

```
jar.exe cf jndiprops.jar jndi.properties
```

where jar.exe resides in a J2SE Java Development Kit, e.g `C:\Program Files\Java\jdk142_05\bin`

If another JNDI provider is used or if ActiveMQ is run inside of a J2EE container, then the Initial Context, Class Path, Connection Factories and Destination will be different or require different JNDI paths.

```
# register some queues in JNDI using the form
# queue.[jndiName] = [physicalName]
queue.MyQueue = example.MyQueue


# register some topics in JNDI using the form
# topic.[jndiName] = [physicalName]
topic.MyTopic = example.MyTopic
```

Figure 1. The jndi.properties file

Trouble Shooting

! ***Please see the Send Port and Receive Location property, Proprietary Connection String, as it is a replacement for using the special initial context factory described below.***

ActiveMQ uses the connection URL to configure failover brokers and wire configuration properties. Because the syntax of the URL is non-standard, a special initial context factory class must be used. This context factory can be found in the support directory in the adapter installation directory, e.g. `C:\Program Files\JNBridge\JMSAdapters\BTS2006\support`. The zip archive *ActiveMQ_ICF.zip* contains a JAR file, *ActiveMQJNBridgeIFC.jar*. The zip archive also includes the source and an Eclipse project.

An ActiveMQ URL that sets wire format properties looks like this:

```
tcp://scriabin:61616?wireFormat.maxInactivityDuration=0&keepAlive=true
```

An ActiveMQ failover URL looks like this:

```
failover:(tcp://scriabin:61616,tcp://stravinsky:61616)?randomize=false
```

A combination of wire formats and failover brokers would look like this:

```
failover:( tcp://scriabin:61616?wireFormat.maxInactivityDuration=30000,
           tcp://elgar:61616?wireFormat.maxInactivityDuration=30000,
           tcp://cage:61616?wireFormat.maxInactivityDuration=30000
         )?randomize=false
```

The first broker, running on the machine *scriabin* and listening on port *61616*, is the primary URL configured in the BizTalk Receive or Send port's Host Name and Port properties. The failover brokers are on machines *elgar* and *cage*.

To use the initial context factory, please follow these instructions:

1. Unpack the zip archive, *ActiveMQ_ICF.zip*, and place the JAR file, *ActiveMQJNBridgeIFC.jar*, in the directory with the other ActiveMQ client JAR files. Add the JAR file to the transport handler Class Path property.
2. Modify the BizTalk transport handler property Initial Context Factory using this value:

   ```
   org.apache.activemq.jndi.ActiveMQJNBridgeInitialContextFactory
   ```
3. In the BizTalk transport handler property, JVM Arguments, use the following syntax:

   ```
   -Dcom.jnbridge.jmsadapter.activemq.failover1=tcp://elgar:61616
   -Dcom.jnbridge.jmsadapter.activemq.failover2=tcp://cage:61616
   -DwireFormat.maxInactivityDuration=30000
   -Drandomize=true
   ```

Up to three failover URLs can be defined.

If the ActiveMQ connection URL requires a custom path, use the *com.jnbridge.jmsadapter.activemq.custompath* property. Custom paths can specify proxy servers or local ports.

The following properties...

```
-DwireFormat.maxInactivityDuration=30000
-Dcom.jnbridge.jmsadapter.activemq.custompath=aProxyServer
```

...would result in a URL like this:

```
tcp://scriabin:61616/aProxyServer?wireFormat.maxInactivityDuration=30000
```

This example shows using the ActiveMQ feature to set the local port used by the JMS client to connect to the broker, avoiding default ephemeral ports and aiding in configuring firewalls. The following properties...

```
-Dcom.jnbridge.jmsadapter.activemq.custompath=0.0.0.0:11467
-DwireFormat.maxInactivityDuration=0
```

...would result in a URL like this:

```
tcp://scriabin:61616/0.0.0.0:11467?wireFormat.maxInactivityDuration=0
```

In addition, the following failover properties can be defined:

```
initialReconnectDelay
maxReconnectDelay
useExponentialBackOff
backOffMultiplier
maxReconnectAttempts
startupMaxReconnectAttempts
randomize
backup
timeout
trackMessages
maxCacheSize
updateURIsSupported
```

The following wire format properties can be defined for the primary URL and each of the failover URLs:

```
wireFormat.stackTraceEnabled
wireFormat.tcpNoDelayEnabled
wireFormat.cacheEnabled
wireFormat.tightEncodingEnabled
wireFormat.prefixPacketSize
wireFormat.maxInactivityDuration
wireFormat.maxInactivityDurationInitalDelay
wireFormat.cacheSize
wireFormat.maxFrameSize
```