



**Using the JNBridge JMS Adapter for .NET  
with ActiveMQ  
version 4.0**

[www.jnbridge.com](http://www.jnbridge.com)

# Using the JMS Adapter with ActiveMQ

---

JNBridge, LLC  
www.jnbridge.com

**COPYRIGHT © 2008-2016 JNBridge, LLC. All rights reserved.**

JNBridge is a registered trademark and JNBridgePro and the JNBridge logo are trademarks of JNBridge, LLC.

Oracle and Java are registered trademarks of Oracle and/or its affiliates.

Microsoft, Windows, Windows Server, BizTalk and the Windows logo are trademarks, or registered trademarks of Microsoft Corporation in the United States and/or other countries.

All other marks are the property of their respective owners.

This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>).

## Contents

Quick Config for ActiveMQ .....	4
Binding Properties Tab .....	4
URI Properties Tab .....	4
Using the JMS Adapter with ActiveMQ .....	5
Resources .....	5
Machine Prerequisites .....	5
Binding Properties Tab .....	6
JNBridge Properties Category .....	8
URI Properties Tab .....	9
Security Tab .....	9
JNDI Names .....	10
Trouble shooting .....	11

# Using the JMS Adapter with ActiveMQ

---

## Quick Config for ActiveMQ

### Binding Properties Tab

- Initial Context Factory: `org.apache.activemq.jndi.ActiveMQInitialContextFactory`
- JMS Scheme: `tcp`
- Queue Factory: `ConnectionFactory`
- Topic 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 (example)

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

### URI Properties Tab

- Port Number: `61616`

## Using the JMS Adapter with ActiveMQ

This document uses the example JMS service that comes 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 .NET developer has access to particular information peculiar to the target JMS implementation. Finally, this document assumes that the reader is knowledgeable and experienced with .NET.

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 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 .NET* document.

### Resources

- The user guide, *JNBridge JMS Adapter for .NET Users' Guide*.
- Chances are, if the target JMS implementation is mature, the values for the configuration of can be supplied by the ActiveMQ administrator, developers or gleaned from existing JMS client code.
- If the ActiveMQ implementation targeted is not configured, then the default example JMS service installed with ActiveMQ can be used for proof-of-concept evaluations.
- It is strongly suggested that the developer read the section *Tips and Tricks* in the *JNBridge JMS Adapter for .NET Users' Guide*.

### Machine Prerequisites

The following prerequisites are needed for the adapter.

- A public 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 .NET uses a stand-alone client 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.

## Configuring the Adapter Connection Properties

The **Add Adapter Service Reference** development tool in Visual Studio is used to generate the `app.config` file and the WCF client file. The WCF client contains the methods chosen to send and receive JMS messages. The `app.config` file contains the binding element whose attribute values are the parameters used to initialize and connect to the JMS server. Each of the binding attribute values can be entered in the **Binding Properties** tab of the **Configure Adapter** dialog box. While it is possible to enter these values and then connect to the JMS server from the **Add Adapter Service Reference** dialog box, it is more efficient to work off-line and generate the `app.config` and the WCF client using only the generic operations. Please see the section *Tips and Tricks* in the *JNBridge JMS Adapter for .NET Users' Guide* for more information about working off-line and using the generic operations.

What follows are the property values required to connect to the default native JMS implementation in ActiveMQ. The versions of ActiveMQ discussed in this document are 5.1 and 5.2.

### Binding Properties Tab

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

#### ■ Choose JMS Vendor

This is a drop-down control that comes pre-charged with default vendor connection properties. Click and select *ActiveMQ* and the default configuration values will automatically appear.

#### ■ Custom 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 and Port properties in the URI tab will be ignored..

#### ■ 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

# Using the JMS Adapter with ActiveMQ

---

of the choice, the JNBridge JMS Adapter will correctly implement the protocol. For a default ActiveMQ connection factory, **AUTO\_ACKNOWLEDGE** is the default configuration.

## ■ Initial Context Factory

This is a text-editable field containing the name of the initial context factory. The initial context factory is a class used to create a JNDI initial context used to look-up connection factories and destinations. The default initial context factory for ActiveMQ is:

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

**!** *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 RMI implementation. The protocol is part of the URI used to connect to the JMS service.

For ActiveMQ, the scheme is:

```
tcp
```

## ■ 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
```

**!** *This value includes the JNDI directory path to the connection factory. The complete JNDI path must be provided.*

# Using the JMS Adapter with ActiveMQ

---

## ■ Topic Connection Factory

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

`ConnectionFactory`

**!** *This value includes the JNDI directory path to the connection factory. The complete JNDI path must be provided.*

## 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 to .NET interoperability components to locate the JMS and JNDI client Java classes so they can be instantiated 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`

## ■ 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 JRE used is:

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

ActiveMQ supports Java 7 or above.



## URI Properties Tab

This category provides the location of the host where ActiveMQ is running and the port that ActiveMQ listens to for connections.

- **Host**

The host name or IP address of the machine running ActiveMQ.

- **Port**

This is a text editable field. Enter the port where ActiveMQ is listening for client connections. By default this is port 61616.

## Security Tab

This tab need only be used if security is implemented in the ActiveMQ server and is of type *simple*. Note that the password will appear in clear text in the app.config file. Please see the section *Deploying Solutions* in the *Users' Guide*.

- **Client credential type**

This is a drop-down control. Choose the credential type *Username*.

- **User name**

Enter the user name credential.

- **Password**

Enter the password credential

# Using the JMS Adapter with ActiveMQ

---

## 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 fail over 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\DotNet\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 fail over 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*, is the primary URL configured in the app.config. The fail over 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 JAR files. Add the JAR file to the transport handler Class Path property.
2. Modify the property Initial Context Factory using this value:

```
org.apache.activemq.jndi.ActiveMQJNBridgeInitialContextFactory
```

3. In the 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
```

# Using the JMS Adapter with ActiveMQ

---

```
-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
```

## Using the JMS Adapter with ActiveMQ

---

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

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