# JNbridge®

## SPANNING JAVA & .NET

# Using the JNBridge JMS Adapter for .NET with WebLogic
## version 4.0

**www.jnbridge.com**

# Using the JMS Adapter with WebLogic

JNBridge, LLC
www.jnbridge.com

# Contents

## Quick Config for WebLogic

**Binding Properties Tab**

- Initial Context Factory: `weblogic.jndi.WLInitialContextFactory`

- JMS Scheme: `t3`

- Queue Factory: `weblogic/examples/jms/QueueConnectionFactory`

- Topic Factory: `weblogic/examples/jms/TopicConnectionFactory`

- Class Path

  WebLogic 9.2

  `wljmsclient.jar, wlclient.jar`

  WebLogic 10.x

  `wljmsclient.jar, wlclient.jar, wlsafclient.jar`

  WebLogic 11g and 12c

  `wlthint3client.jar`

! *WebLogic version 9.x jar files may fail on a send with a* java.lang. NoClassDefFoundError *as the anonymous inner class* weblogic.jms.common. MessageImpl$1 *cannot be found. Please use the jar files from WebLogic version 10.x: they're backward compatible with WebLogic version 9.x.*

- JVM Path (example)

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

**URI Properties Tab**

- Port Number: `7001`

# Using the JMS Adapter with WebLogic

This document uses the example JMS service that comes pre-configured in WebLogic. 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 only discusses those property values that pertain to communicating with WebLogic. 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 WebLogic JMS implementation is mature, the values for the configuration of BizTalk transport handlers and send/receive ports can be supplied by the WebLogic administrator, gleaned from existing JMS client code or property files, e.g. *jndi.properties*.

- If the WebLogic JMS implementation targeted is not configured, then the default example JMS service installed with WebLogic can be used for proof-of-concept evaluations. This document uses the example JMS service in WebLogic 9.2.

- 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 BizTalk uses the stand-alone or thin-client JMS environment supplied by Oracle.

## 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 WebLogic. The versions of WebLogic discussed in this document are 9.2, 10.x and 11.x.

### 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 *WebLogic* 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. that can not be constructed from the Host Name and Port Number properties. 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 of the choice, the JNBridge JMS Adapter will correctly implement the protocol. For a default WebLogic 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 WebLogic is:

**weblogic.jndi.WLInitialContextFactory**

**!** *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 WebLogic, the scheme is:

**t3** or **t3s**.

When using the WebLogic thin-client, the t3 protocol is actually implemented as RMI-IIOP.

■ 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 WebLogic is:

**weblogic/examples/jms/QueueConnectionFactory**

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

■ Topic Connection Factory

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

**weblogic/examples/jms/TopicConnectionFactory**

**!** *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 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.

**WebLogic 9.2**

```
[WebLogic_Directory]\server\lib\wljmsclient.jar
[WebLogic_Directory]\server\lib\wlclient.jar
```

**WebLogic 10.x**

```
[WebLogic_Directory]\server\lib\wljmsclient.jar
[WebLogic_Directory]\server\lib\wlclient.jar
[WebLogic_Directory]\server\lib\wlsafclient.jar
```

WebLogic 11g and 12c

```
[WebLogic_Directory]\server\lib\wlthint3client.jar
```

**!** *WebLogic version 9.x jar files may fail on a send with a* **java.lang. NoClassDefFoundError** *as the anonymous inner class* **weblogic.jms.common. MessageImpl$1** *cannot be found. Please use the jar files from WebLogic version 10.x: they're backward compatible with WebLogic version 9.x.*

■ 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
```

WebLogic supports Java 5 and Java 6. Java 4 is not supported.

**URI Properties Tab**

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

■ Host

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

■ Port

This is a text editable field. Enter the port where WebLogic is listening for client connections.

By default this is port `7001`.

**Security Tab**

This tab need only be used if security is implemented in the WebLogic 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 WebLogic

## JNDI Names

Figure1 shows the WebLogic 9.2 Admin Console. The console is displaying the JNDI directory structure for a server. It is important to use complete JNDI paths, either forward slash ('/') or dot ('.') delineated, for connection factories and JMS destinations.
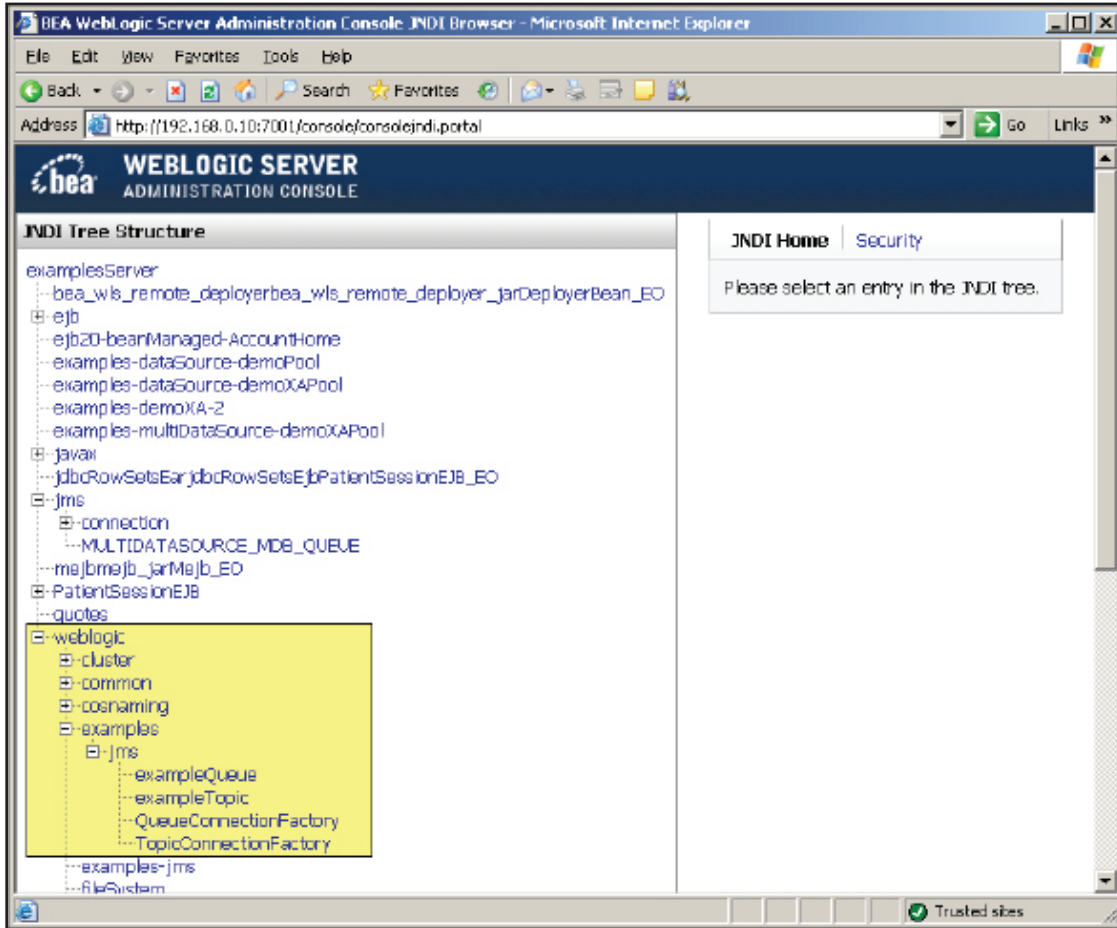


**Figure 1. The JNDI tree**

## Touble Shooting

### Could not initialize LookupInitializer class

On occasion, the WebLogic JMS client may throw an exception complaining about the inability to initialize an i18n class. This exception seems to be related to closing, then reopening the connection to the server. This is the exception:

```
An unexpected failure occurred while processing a message. The
text associated with the exception is "Could not initialize class
weblogic.i18ntools.L10nLookup$L10nLookupInitializer".
```

The associated Java-side stack trace looks like this:

```
java.lang.ExceptionInInitializerError
at weblogic.i18ntools.L10nLookup.getLocalizer(L10nLookup.java:362)
at weblogic.i18n.logging.Loggable.getMessage(Loggable.java:181)
at weblogic.i18n.logging.Loggable.getMessage(Loggable.java:207)
at weblogic.jms.common.JMSException.<init>(JMSException.java:66)
        .
        .
        .
Caused by: java.lang.NullPointerException
at weblogic.i18ntools.L10nLookup.loadProps(L10nLookup.java:125)
at weblogic.i18ntools.L10nLookup.<init>(L10nLookup.java:187)
at weblogic.i18ntools.L10nLookup.<init>(L10nLookup.java:27)
at weblogic.i18ntools.L10nLookup$L10nLookupInitializer.<clinit>(L10nLookup.
java:79)
```

This exception is caused by the Java class loader not being able to locate an inner class. This can be solved by appending the WebLogic thin-client JAR files to the JVM boot class path using the JVM Arguments property. Enter this into the Binding Properties tab or as the value for the JVMArgs attribute in the app.config file:

```
-Xbootclasspath/a:[CLASSPATH]
```

where [CLASSPATH] is the same class path as in the Class Path property.