

Artix[®] Connect for WCF

Getting Started Guide

Version 1.5
October 2008

Getting Started Guide

Progress Software

Version 1.5

Published 06 Nov 2008

Copyright © 2008 IONA Technologies PLC , a wholly-owned subsidiary of Progress Software Corporation.

Legal Notices

Progress Software Corporation and/or its subsidiaries may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this publication. Except as expressly provided in any written license agreement from Progress Software Corporation, the furnishing of this publication does not give you any license to these patents, trademarks, copyrights, or other intellectual property. Any rights not expressly granted herein are reserved.

Progress, IONA, IONA Technologies, the IONA logo, Orbix, High Performance Integration, Artix, FUSE, and Making Software Work Together are trademarks or registered trademarks of Progress Software Corporation and/or its subsidiaries in the US and other countries.

Java and J2EE are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. CORBA is a trademark or registered trademark of the Object Management Group, Inc. in the US and other countries. All other trademarks that appear herein are the property of their respective owners.

While the information in this publication is believed to be accurate Progress Software Corporation makes no warranty of any kind to this material including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Progress Software Corporation shall not be liable for errors contained herein, or for incidental or consequential damages in connection with the furnishing, performance or use of this material.

All products or services mentioned in this manual are covered by the trademarks, service marks, or product names as designated by the companies who market those products.

No part of this publication may be reproduced, stored in a retrieval system or transmitted, in any form or by any means, photocopying, recording or otherwise, without prior written consent of IONA Technologies PLC. No third-party intellectual property right liability is assumed with respect to the use of the information contained herein. IONA Technologies PLC assumes no responsibility for errors or omissions contained in this publication. This publication and features described herein are subject to change without notice. Portions of this document may include Apache Foundation documentation, all rights reserved.

Table of Contents

Preface	11
The Artix Connect for WCF Library	12
Document Conventions	13
Introducing the Sample Application	15
CORBA and JMS Sample Application	16
Choosing Your JMS Broker	19
Using ActiveMQ	21
Using Other JMS Brokers	22
Using TIBCO EMS	23
Using SonicMQ 7.5	26
Using WebSphere MQ 6.0	30
Using BEA WebLogic 10	34
Running the Tutorial	37
Step 1: Running the Back-end Services	38
Step 2: Opening the .NET Solution	41
Step 3: Opening the Artix Connect for WCF wizard	44
Step 4: Using the Wizard to Connect to CORBA	47
Step 5: Using the Wizard to Connect to JMS	50
Step 6: Making CORBA and JMS Operations Available to Your WCF Application	58
Step 7: Adding Code to Call to the CORBA and JMS Systems	60
Step 8: Running the Stock Purchasing Application	62
Index	65

List of Figures

1. Sample Application Architecture	16
2. Configuring a JMS Broker	19
3. CORBA Server Ready and Waiting for Requests	38
4. Fully Initialized FUSE Message Broker	39
5. Fully Initialized Java Server	40
6. .NET Stock Purchase Application	42
7. Adding an Adapter Service Reference	44
8. The Add Adapter Service Reference wizard	45
9. Artix Connect for WCF Wizard	46
10. CORBA Object Details Window	48
11. CORBA StockQuote System Added to Deployed Clients List	49
12. Adding JMS Broker Settings	51
13. Adding JMS Client Name and Payload Format Details	52
14. Defining XML Message	53
15. XML Message Defined	54
16. JMS Destinations Settings	56
17. CORBA and JMS Clients Successfully Deployed	57
18. JMS and CORBA details in the LOB Adapter Window	58
19. The Completed WCF Application	62
20. CORBA Server Logging an Operation Call	63
21. Running the Completed Stock Purchase Application	63
22. Java Server Consuming JMS Request	64

List of Tables

1. JMS Destination Settings for TIBCO EMS	24
2. JMS Destination Settings for SonicMQ	29
3. JMS Destination Settings for WebSphere MQ	33
4. JMS Destination Settings for BEA WebLogic	35
5. JMS Destination Settings for FUSE Message Broker and ActiveMQ	55

List of Examples

1. Stock Quote System—IDL	17
2. Business Interface: StockTrader.java	18
3. Sample Java Server JNDI Properties for TIBCO EMS	23
4. Sample Java Server Constructor Code for TIBCO EMS	23
5. Adding TIBCO EMS to the Classpath	24
6. Sample Java Server JNDI Properties for SonicMQ	27
7. Sample Java Server Constructor Code for SonicMQ	28
8. Adding SonicMQ to the Classpath	29
9. WebSphere MQ JMSAdmin.config	30
10. Sample Java Server JNDI properties for WebSphere MQ	31
11. Sample Java Server Constructor Code for WebSphere MQ	32
12. Adding WebSphere MQ to the Classpath	32
13. Sample Java Server JNDI Properties for WebLogic	34
14. Sample Java Server Constructor Code for BEA WebLogic	34
15. Adding Weblogic to the Classpath	35
16. Adding a ActiveMQ to the Classpath	40
17. Adding FUSE Message Broker 5.1.0.0 to the Classpath	40
18. ServiceCalls.cs after modification	60

Preface

The Artix Connect for WCF Library	12
Document Conventions	13

The Artix Connect for WCF Library

The Artix Connect for WCF documentation library consists of the following books:

- [Installation Guide](#)

Read the Installation Guide if you are about to install Artix Connect for WCF.

- [Release Notes](#)

Read the Release Notes for a list of features, known issues, and release-specific information.

- [Getting Started Guide on page 1](#)

Read this Getting Started Guide if you are new to Artix Connect for WCF and want to walk through a step-by-step tutorial that shows you how to use Artix Connect for WCF to integrate a .NET application with a CORBA and JMS back-end.

- [User's Guide](#)

Read the User's Guide if you want to use Artix Connect for WCF to integrate a .NET application with CORBA, JMS queues and topics, or EJBs.

- [BizTalk Integration Guide](#)

Read the BizTalk Integration Guide if you want to walk through a steps-by-step tutorial that shows you how to use Artix Connect for WCF to integrate BizTalk Server 2006 or BizTalk Server 2006 R2 with a JMS back-end system and a CORBA back-end system.

Document Conventions

Typographical conventions

This book uses the following typographical conventions:

<code>fixed width</code>	<p>Fixed width (Courier font) in normal text represents portions of code and literal names of items such as classes, functions, variables, and data structures. For example, text might refer to the <code>javax.xml.ws.Endpoint</code> class.</p> <p>Constant width paragraphs represent code examples or information a system displays on the screen. For example:</p> <pre>import java.util.logging.Logger;</pre>
<code>Fixed width italic</code>	<p>Fixed width italic words or characters in code and commands represent variable values you must supply, such as arguments to commands or path names for your particular system. For example:</p> <pre>% cd /users/YourUserName</pre>
<code>Italic</code>	<p>Italic words in normal text represent <i>emphasis</i> and introduce <i>new terms</i>.</p>
Bold	<p>Bold words in normal text represent graphical user interface components such as menu commands and dialog boxes. For example: the User Preferences dialog.</p>

Keying conventions






This book uses the following keying conventions:

No prompt	When a command's format is the same for multiple platforms, the command prompt is not shown.
%	A percent sign represents the UNIX command shell prompt for a command that does not require root privileges.
#	A number sign represents the UNIX command shell prompt for a command that requires root privileges.
>	The notation > represents the MS-DOS or Windows command prompt.
...	Horizontal or vertical ellipses in format and syntax descriptions indicate that material has been eliminated to simplify a discussion.
[]	Brackets enclose optional items in format and syntax descriptions.
{ }	Braces enclose a list from which you must choose an item in format and syntax descriptions.

	In format and syntax descriptions, a vertical bar separates items in a list of choices enclosed in {} (braces).
--	---

Admonition conventions

This book uses the following conventions for admonitions:

	Notes display information that may be useful, but not critical.
	Tips provide hints about completing a task or using a tool. They may also provide information about workarounds to possible problems.
	Important notes display information that is critical to the task at hand.
	Cautions display information about likely errors that can be encountered. These errors are unlikely to cause damage to your data or your systems.
	Warnings display information about errors that may cause damage to your systems. Possible damage from these errors include system failures and loss of data.

Introducing the Sample Application

This chapter introduces the Artix Connect for WCF sample application that is used in the step-by-step tutorial described in this book. In addition, it describes some prerequisite steps that you might need to complete for your JMS broker if you want to use it with the sample application.

CORBA and JMS Sample Application	16
Choosing Your JMS Broker	19
Using ActiveMQ	21
Using Other JMS Brokers	22
Using TIBCO EMS	23
Using SonicMQ 7.5	26
Using WebSphere MQ 6.0	30
Using BEA WebLogic 10	34

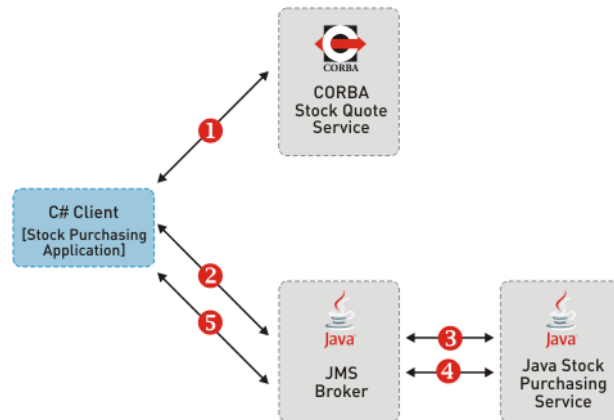
CORBA and JMS Sample Application

Overview

Artix Connect for WCF includes a ready-to-run sample application that shows how to integrate a simple C# Windows application with both a CORBA and a JMS back-end system. It demonstrates how you, as a .NET application programmer, can quickly and easily write code to connect to a CORBA and a JMS system from within the .NET environment. [Figure 1 on page 16](#)

The sample application works as shown in [Figure 1 on page 16](#).

Figure 1. Sample Application Architecture



1. When you, as the user, select a stock using the C# client UI, the C# client makes a synchronous call, using IIOP on the wire, to the CORBA Stock Quote Service. The Stock Quote service returns a stock price to the C# client, which is displayed in the client UI.
2. You can then select the number of shares you want to buy, using the C# client UI, and a message is placed on a JMS queue, which is managed by a JMS broker.
3. The Stock Purchasing back-end, which is implemented in Java, consumes that message.
4. and 5. The Java server responds, synchronously, using JMS to tell the C# client that the shares have been purchased.

Artix Connect for WCF is responsible for enabling the C# client to talk to the back-end systems.

CORBA Stock Quote System

The sample CORBA system consists of a simple IDL interface that provides a stock quote system. The IDL is shown in [Example 1 on page 17](#). Clients of the service pass a stock symbol string, such as `MSFT` or `IONA`, as a parameter to the `price` operation and receive a return value simulating the market value of that stock.

Example 1. Stock Quote System—IDL

```
// OMG IDL
interface StockQuote
{
    double price (in string symbol);
};
```

JMS Stock Purchase System

The JMS system enables you to buy stock. It consists of a JMS broker and a separate Java server that consumes the messages that are sent to the JMS broker.

Unlike CORBA, JMS is not a typed middleware technology. The payloads in JMS messages are very flexible. For example, JMS architects can create encoding rules for the payloads in their system to enable them to send structured message data across the broker.

The sample JMS server consumes messages from the JMS broker's trading queue. It expects each message payload to have two parts, encoded in XML. The parts are:

- The stock symbol that is being purchased
- The quantity of stock that is being purchased

Artix Connect for WCF can follow these encoding rules and present the JMS queue as a typed business interface, rather than as a simple data transporter. It does this by examining a Java class that represents the business interface being used for the queue. The Java class can be found in the following directory of your Artix Connect for WCF installation:

```
InstallDir\Visual Studio  
Adapter\samples\corba_jms\jms\bin\com\acme\stock\trade\StockTrader.class
```

Example 2. Business Interface: StockTrader.java

```
public class StockTrader {  
    public void buyShares (String symbol, int quantity) { ... }  
    ...  
}
```

By enabling you to specify a Java class as your business interface to the JMS system, you benefit from having a typed interface to the broker's queues and you do not have to manage the payload encoding rules manually.

Artix Connect for WCF also supports WCF clients interacting with a JMS system in the basic, untyped manner. In this mode, you are responsible for encoding the payload data in the format that the consumer expects and see a very rudimentary string-based interface to the system.

Location and structure of sample

The sample application is installed in the following directory of your Artix Connect for WCF installation:

```
InstallDir\Visual Studio Adapter\samples\corba_jms
```

It contains the following subfolders:

- `bin`—contains prebuilt executables for the CORBA and JMS services.
- `corba`—contains the source code for the CORBA system.
- `dotnet`—contains the Visual Studio 2005 solution for the .NET application.
- `etc`—contains the IDL file for the CORBA system.
- `jms`—contains the source code and Java class files for the JMS system.

Choosing Your JMS Broker

Choosing a broker

You can use the Artix Administration tool to configure Artix Connect for WCF to use a particular JMS message broker by default.



Note

If you set a default JMS broker in the Artix Administration tool, you will not be asked to select a broker when creating a JMS client in the Artix Connect for WCF wizard.

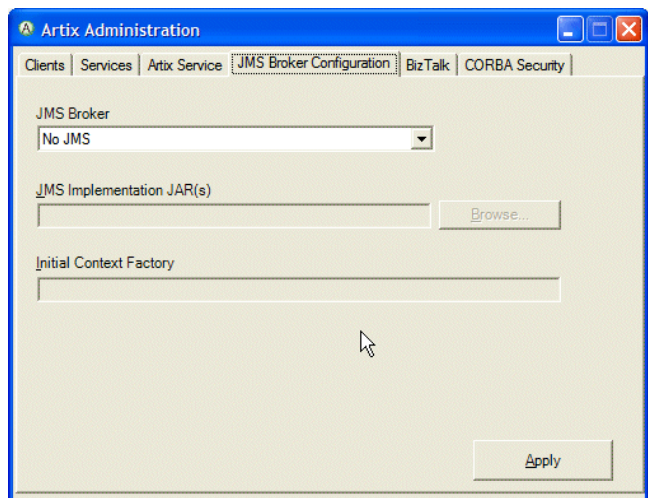
To configure a default message broker:

1. Open the Artix Administration tool from the Windows Start menu by selecting:

(All) Programs → IONA → Artix Connect For WCF Artix Administration

2. Click the **JMS Broker Configuration** tab, as shown in [Figure 2 on page 19](#):

Figure 2. Configuring a JMS Broker



3. Under JMS Broker, select the broker that you want to use from the drop-down list.

The **Initial Context Factory** field is set automatically when you select a broker.

4. In the JMS Implementation JAR(s) field, enter the location of the JMS implementation JAR or JARs for the broker that you selected in the previous step.

For a complete list of JMS implementation JARs, see [JMS Broker Implementation JARs](#) in the *Installation Guide*.

5. Click **Apply**.

Using ActiveMQ

Default broker

The sample application is set up to use Apache ActiveMQ or its enterprise equivalent, FUSE Message Broker, by default.

If running ActiveMQ, you do not need to make any changes to the sample Java server's JNDI properties or constructor code, as you do if you are using any other broker.

Disabling multicast

If your system does not allow multicast discovery, you may experience some issues when running ActiveMQ. If this happens, turn off multicast in your ActiveMQ configuration, as follows:

1. Go to your ActiveMQ or FUSE Message Broker installation directory and open the following configuration file:

```
AMQInstallDir/conf/activemq.xml
```

2. In the `activemq.xml` file locate the `NetworkConnectors` element and comment out the lines relating to multicast as follows:

```
<networkConnectors>
  <!-- by default just auto discover the other brokers
-->
  <!-- <networkConnector name="default-nc" uri="multicast://default"/> -->
  <!--
    <networkConnector name="host1 and host2" uri="static://(tcp://host1:61616,tcp://host2:61616)" fail
over="true"/>
  -->
</networkConnectors>
```

3. Save the changes to the `activemq.xml` file.

Using Other JMS Brokers

Using TIBCO EMS	23
Using SonicMQ 7.5	26
Using WebSphere MQ 6.0	30
Using BEA WebLogic 10	34

If you want to use a JMS broker other than ActiveMQ, you need to:

- Change the JNDI properties and constructor code used in the sample Java server.
- Add your JMS broker's implementation JAR to the classpath before running the sample Java server.
- Use the appropriate values when adding the JMS destination settings in the Artix Connect for WCF wizard, when running the sample application.
- Start your chosen JMS broker.

The following sections provide details on the additional changes you need to make for each supported broker.

- [Using TIBCO EMS on page 23](#)
- [Using SonicMQ 7.5 on page 26](#)
- [Using WebSphere MQ 6.0 on page 30](#)
- [Using BEA WebLogic 10 on page 34](#)

Using TIBCO EMS

Read the following section if you want to use TIBCO EMS as your JMS broker.

Updating the sample Java server JNDI properties for TIBCO

To update the sample Java server JNDI properties for TIBCO EMS:

1. In Windows Explorer, navigate to the following directory of your Artix Connect for WCF installation:

```
InstallDir\Visual Studio Adapter\samples\corba_jms\jms
```

2. Open the `jndi.properties` file and replace the contents with the following lines of code:

Example 3. Sample Java Server JNDI Properties for TIBCO EMS

```
java.naming.factory.initial = com.tibco.tibjms.naming.TibjmsInitialContextFactory
java.naming.provider.url = tcp://localhost:7222
```

3. Save the changes to the `jndi.properties` file.
-

Changing the Java server constructor code for TIBCO

To change the Java server constructor code:

1. In Windows Explorer, navigate to the following directory of your Artix Connect for WCF installation:

```
InstallDir\Visual Studio
Adapter\samples\corba_jms\jms\src\com\acme\stock\trade\jms
```

2. Open the `StockTraderJMS.java` file and change the following lines of code:

Example 4. Sample Java Server Constructor Code for TIBCO EMS

```
QueueConnectionFactory qcf = (QueueConnectionFactory)ctx.lookup("QueueConnectionFactory");

Queue queue = (Queue)ctx.lookup("queue.sample");
```

```
Queue responseQueue = (Queue) ctx.lookup("queue.sample1");
```

3. Navigate to the following directory of your Artix Connect for WCF installation:

```
InstallDir\Visual Studio Adapter\samples\corba_jms\jms
```

4. Rebuild the Java server by running the `buildjava.bat` file.

Adding TIBCO EMS to the classpath

To add the TIBCO EMS implementation JARs to your `CLASSPATH`:

1. Open a Windows command prompt
2. Run the following command:

Example 5. Adding TIBCO EMS to the Classpath

```
set CLASSPATH=TIBCOEMSInstallDir\clients\java\tibjms.jar;  
TIBCOEMSInstallDir\clients\java\jms.jar;%CLASSPATH%
```

Configuring TIBCO JMS Destination Settings

When working through the tutorial, in [Step 5: Using the Wizard to Connect to JMS on page 50](#), you are asked to provide JMS destination settings. In the JMS Destination Settings window, enter the settings shown in [Table 1 on page 24](#):

Table 1. JMS Destination Settings for TIBCO EMS

Setting	Value
<i>Destination Type</i>	Queue
<i>Request Queue Name</i>	queue.sample
<i>Reply Queue Name</i>	queue.sample1
<i>JNDI connection factory name</i>	QueueConnectionFactory

Setting	Value
JNDI naming provider URL	tcp://localhost:7222

Starting TIBCO EMS

To start the TIBCO EMS JMS broker:

1. Navigate to the following directory of your TIBCO EMS installation:

`InstallDir\bin`

2. Run the `tibemsd.exe` file.

Using SonicMQ 7.5

Read the following section if you want to use SonicMQ 7.5 as your JMS broker.

Configuring SonicMQ for JMS

Refer to your SonicMQ documentation or speak with your SonicMQ administrator for details on how to configure SonicMQ for JMS.

The following information is given as an example for the purposes of running the Artix Connect for WCF sample application.

To configure SonicMQ for use with JMS:

1. Open a Windows command prompt and navigate to the following directory of your SonicMQ installation:

```
InstallDir\bin
```

2. Run the `startmc.bat` file.
3. Select **Tools | JMS Administered Objects**.
4. Under **Create new Connection**, select **JNDI Naming Service**.
5. Click Sonic Storage and:
 - a. Notice that the Context factory is filled in automatically.
 - b. `Domain = Domain1`
 - c. `Provider URL = localhost`
 - d. Click **Connect**.
You should see: Established store = localhost
6. On the left panel, under Objects stores, choose `localhost`.
7. On the right panel, select the Connection factories tab and create a new one as follows:
 - a. Under the General tab:

- In the Lookup name and factory type, enter:
`QueueConnectionFactory`
 - In Connection Url, enter: `tcp://localhost:2506`
- b. Click **Update**.
 - c. Make sure that, at the top of the Connection factories tab, there is a lookup name and a factory name, each with the value of `QueueConnectionFactory`.
8. Choose the Destinations tab and:
 - a. Click **New**.
 - b. Choose Lookup Name `Sample1`
 - c. Type `Queue`, with Destination Name `SampleQ1`
 9. You have now created a `QueueConnectionFactory` with a queue called `SampleQ1`. Repeat the steps to create a `SampleQ2` for your reply messages.
 10. Close the Sonic Management Console.

Updating the sample Java server JNDI properties for SonicMQ

To update the sample Java server JNDI properties for SonicMQ

1. In Windows Explorer, navigate to the following directory of your Artix Connect for WCF installation:

```
InstallDir\Visual Studio Adapter\samples\corba_jms\jms
```

2. Open the `jndi.properties` file and replace the contents with the following lines of code:

Example 6. Sample Java Server JNDI Properties for SonicMQ

```
java.naming.factory.initial = com.sonicsw.jndi.mfcontext.MFContextFactory
```

```
java.naming.provider.url = tcp://localhost:2506
```

3. Save your changes to the `jndi.properties` file.

Changing the Java server constructor code for SonicMQ

To change the Java server constructor code for SonicMQ:

1. In Windows Explorer, navigate to the following directory of your Artix Connect for WCF installation:

```
InstallDir\Visual Studio  
Adapter\samples\corba_jms\jms\src\com\acme\stock\trade\jms
```

2. Open the `StockTraderJMS.java` file and change the following lines of code:

Example 7. Sample Java Server Constructor Code for SonicMQ

```
QueueConnectionFactory qcf = (QueueConnectionFactory)ctx.lookup("QueueConnectionFactory");  
Queue queue = (Queue)ctx.lookup("SampleQ1");  
Queue responseQueue = (Queue)ctx.lookup("SampleQ2");
```

3. Navigate to the following directory of your Artix Connect for WCF installation:

```
InstallDir\Visual Studio Adapter\samples\corba_jms\jms
```

4. Rebuild the Java server by running the `buildjava.bat` file.

Adding SonicMQ to the classpath

To add the SonicMQ JMS implementation JARs to your `CLASSPATH`

1. Open a Windows command prompt
2. Run the following command:

Example 8. Adding SonicMQ to the Classpath

```
set CLASSPATH=SonicMQInstallDir\MQVersion\lib\mfcontext.jar;
SonicMQInstallDir\MQVersion\lib\sonic_XA.jar;
SonicMQInstallDir\wizard.jar;%CLASSPATH%
```

Configuring SonicMQ JMS Destination Settings

When working through the tutorial, in [Step 5: Using the Wizard to Connect to JMS on page 50](#), you are asked to provide JMS destination settings. In the JMS Destination Settings window, enter the settings shown in [Table 2 on page 29](#):

Table 2. JMS Destination Settings for SonicMQ

Setting	Value
<i>Destination Type</i>	Queue
<i>Request Queue Name</i>	SampleQ1
<i>Reply Queue Name</i>	SampleQ2
<i>JNDI connection factory name</i>	QueueConnectionFactory
<i>JNDI naming provider URL</i>	tcp://localhost:2506

Starting SonicMQ

To start the SonicMQ JMS broker:

1. Navigate to the following directory of your SonicMQ 7.5 installation:

```
InstallDir\bin
```

2. Run the `startcontainer.bat` file.

Using WebSphere MQ 6.0

Read the following section if you want to use WebSphere MQ 6.0 as your JMS broker.

Configuring WebSphere MQ

WebSphere MQ uses some local queues for specific operational purposes. You must define these queues before WebSphere MQ can use them. Refer to your WebSphere documentation or speak with your WebSphere MQ administrator for details.

The following information is given as an example of what to do for the purposes of running the Artix Connect for WCF sample application.

Creating and starting a queue manager

Assuming that your working directory is the following directory of your Artix Connect for WCF installation:

```
InstallDir\Visual Studio Adapter\samples\corba_jms\jms
```

1. Create a Queue Manager by opening a Windows command prompt and typing:

```
crtmqm -q MY_DEF_QM
```

2. Start the queue manager by, in the same Windows command prompt, typing:

```
amqmdain qmgr start
```

Setting up the WebSphere MQ administration tool

Before you can use the WebSphere MQ Administration tool you must create a JMS administration configuration file. To do so:

1. In the working directory, *ArtixConnectforWCFInstallDir\Visual Studio Adapter\samples\corba_jms\jms*, create a file called `JMSAdmin.config` with the following contents:

Example 9. WebSphere MQ JMSAdmin.config

```
INITIAL_CONTEXT_FACTORY=com.sun.jndi.fscontext.RefFSContextFactory
PROVIDER_URL=file:%IT_ARTIX_WCF_DIR%\Visual Studio Adapter\samples\wcf\corba_jms\jms
SECURITY_AUTHENTICATION=none
```

- From your working directory, run the following command:

```
java -classpath %CLASSPATH% com.ibm.mq.jms.admin.JMSAdmin
-t -v -cfg JMSAdmin.config
```

- At the `InitCtx>` prompt, type the following:

```
def qcf(QueueConnectionFactory)
def q(TradeQueue) qu(TEST.JMSTRANSPORT.TEXT)
def q(TradeResponseQueue) qu(TEST.JMSTRANSPORT.TEXT)
end
```

You will notice that a `.bindings` file is created locally.

Updating the sample Java server JNDI properties for Websphere MQ

To update the sample Java server JNDI properties for Websphere MQ:

- In Windows Explorer, navigate to the following directory of your Artix Connect for WCF installation:

```
InstallDir\Visual Studio Adapter\samples\corba_jms\jms
```

- Open the `jndi.properties` file and replace the contents with the following lines of code:

Example 10. Sample Java Server JNDI properties for WebSphere MQ

```
java.naming.factory.initial = com.sun.jndi.fscontext.RefFSContextFactory
java.naming.provider.url = file:%IT_ARTIX_WCF_DIR%\Visual Studio Ad
apter\samples\wcf\corba_jms\jms
```

- Save your changes to the `jndi.properties` file.

Changing the Java server constructor code for Websphere MQ

To change the Java server constructor code for Websphere MQ:

- In Windows Explorer, navigate to the following directory of your Artix Connect for WCF installation:

```
InstallDir\Visual Studio  
Adapter\samples\corba_jms\jms\src\com\acme\stock\trade\jms
```

2. Open the `StockTraderJMS.java` file and change the following lines of code:

Example 11. Sample Java Server Constructor Code for WebSphere MQ

```
QueueConnectionFactory qcf = (QueueConnectionFactory)ctx.lookup("QueueConnectionFactory");  
Queue queue = (Queue)ctx.lookup("TradeQueue");  
Queue responseQueue = (Queue)ctx.lookup("TradeResponseQueue");
```

3. Navigate to the following directory of your Artix Connect for WCF installation:

```
InstallDir\Visual Studio Adapter\samples\corba_jms\jms
```

4. Rebuild the Java server by running the `buildjava.bat` file.

Adding Websphere MQ to the classpath

To add the Websphere MQ implementation JARs to your `CLASSPATH`:

1. Open a Windows command prompt
2. Run the following command:

Example 12. Adding WebSphere MQ to the Classpath

```
set CLASSPATH=WebSphereMQInstallDir\java\lib\com.ibm.mqjms.jar;%CLASSPATH%
```

Configuring Websphere MQ JMS Destination Settings

When working through the tutorial, in [Step 5: Using the Wizard to Connect to JMS on page 50](#), you are asked to provide JMS destination settings. In the JMS Destination Settings window, enter the settings shown in [Table 3 on page 33](#). The settings shown are example values taken from the queue that you created in [Using WebSphere MQ 6.0 on page 30](#). You can, of course, use values for other queues that you or your administrator have created.

Table 3. JMS Destination Settings for WebSphere MQ

Setting	Value
<i>Destination Type</i>	Queue
<i>Request Queue Name</i>	TradeQueue
<i>Reply Queue Name</i>	TradeResponseQueue
<i>JNDI connection factory name</i>	QueueConnectionFactory
<i>JNDI naming provider URL</i>	file:%IT_ARTIX_WCF_DIR%\Visual Studio Adapter\samples\wcf\corba_jms\jms

Starting Websphere MQ

To start the WebSphere MQ JMS broker:

1. Navigate to the following directory of your WebSphere MQ 6.0 installation:

```
InstallDir\bin
```

2. Run the `amqsvc.exe` file.

Using BEA WebLogic 10

Read the following section if you want to use BEA WebLogic 10 as your JMS broker.

Updating the sample Java server JNDI properties for WebLogic

To update the sample Java server JNDI properties for WebLogic:

1. In Windows Explorer, navigate to the following directory of your Artix Connect for WCF installation:

```
InstallDir\Visual Studio Adapter\samples\corba_jms\jms
```

2. Open the `jndi.properties` file and replace the contents with the following lines of code:

Example 13. Sample Java Server JNDI Properties for WebLogic

```
java.naming.factory.initial = weblogic.jndi.WLInitialContextFactory
java.naming.provider.url = t3://localhost:7001
```

3. Save your changes to the `jndi.properties` file.
-

Changing the Java server constructor code for WebLogic

To change the Java server constructor code for WebLogic:

1. In Windows Explorer, navigate to the following directory of your Artix Connect for WCF installation:

```
InstallDir\Visual Studio Adapter\samples\corba_jms\jms\src\com\acme\stock\trade\jms
```

2. Open the `StockTraderJMS.java` file and change the following lines of code:

Example 14. Sample Java Server Constructor Code for BEA WebLogic

```
QueueConnectionFactory qcf = (QueueConnectionFactory)ctx.lookup("weblogic.examples.jms.QueueConnectionFactory");
```

```
Queue queue = (Queue)ctx.lookup("weblogic.examples.jms.exampleQueue");
Queue responseQueue = (Queue)ctx.lookup("weblogic.examples.jms.exampleQueue");
```

3. Navigate to the following directory of your Artix Connect for WCF installation:

```
InstallDir\Visual Studio Adapter\samples\corba_jms\jms
```

4. Rebuild the Java server by running the `buildjava.bat` file.

Adding WebLogic to the classpath

To add the WebLogic implementation JARs to your `CLASSPATH`:

1. Open a Windows command prompt
2. Run the following command:

Example 15. Adding Weblogic to the Classpath

```
set CLASSPATH=BEAWebLogicInstallDir\server\lib\weblogic.jar;%CLASSPATH%
```

Configuring WebLogic JMS Destination Settings

When working through the tutorial, in [Step 5: Using the Wizard to Connect to JMS on page 50](#), you are asked to provide JMS destination settings. In the JMS Destination Settings window, enter the settings shown in [Table 4 on page 35](#).

Table 4. JMS Destination Settings for BEA WebLogic

Setting	Value
<i>Destination Type</i>	Queue
<i>Request Queue Name</i>	weblogic.examples.jms.exampleQueue
<i>Reply Queue Name</i>	weblogic.examples.jms.exampleQueue2
<i>JNDI connection factory name</i>	weblogic.examples.jms.QueueConnectionFactory

Setting	Value
<i>JNDI naming provider URL</i>	t3://localhost:7001

Starting WebLogic

For the purposes of running the Artix Connect for WCF tutorial, start the WebLogic example server. You can do this from the Windows start menu as follows:

Start | BEA Products | Examples | WebLogic Server | Start Example Server

Running the Tutorial

This chapter walks you, step-by-step, through the Artix Connect for WCF sample application.

Step 1: Running the Back-end Services	38
Step 2: Opening the .NET Solution	41
Step 3: Opening the Artix Connect for WCF wizard	44
Step 4: Using the Wizard to Connect to CORBA	47
Step 5: Using the Wizard to Connect to JMS	50
Step 6: Making CORBA and JMS Operations Available to Your WCF Application	58
Step 7: Adding Code to Call to the CORBA and JMS Systems	60
Step 8: Running the Stock Purchasing Application	62

Step 1: Running the Back-end Services

Overview

The back-end services consist of a CORBA server, a JMS broker and a Java server.

Running the CORBA server

To run the CORBA server:

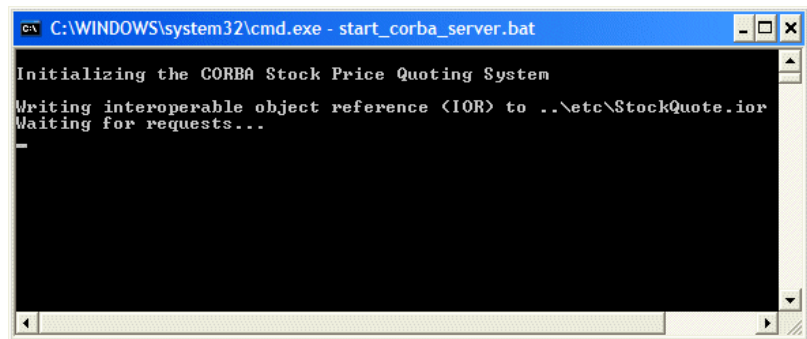
1. Open a Windows Explorer window and navigate to the following directory of your Artix Connect for WCF installation:

```
InstallDir\Visual Studio Adapter\samples\corba_jms\bin
```

2. Double-click on the `start_corba_server.bat` file to start the CORBA server.
3. If the Windows Firewall asks if you want to unblock the application, select **Unblock**.

The CORBA server takes a few seconds to start. When it is ready and listening for incoming requests, it should appear as shown in [Figure 3 on page 38](#):

Figure 3. CORBA Server Ready and Waiting for Requests



Running the JMS broker

This section assumes that you are using FUSE Message Broker or Apache ActiveMQ. If you are not, make whatever configuration changes your JMS broker requires and start the broker as described in [Using Other JMS Brokers on page 22](#).

To run ActiveMQ:

1. Open Windows Explorer and navigate to the following directory of your Apache ActiveMQ or FUSE Message Broker installation:

```
InstallDir\bin
```

2. Double-click on the `activemq.bat` file to start the JMS broker.
3. Wait for the broker to fully initialize; that is, until you see the message "Started SelectChannelConnector@0.0.0.0:8161" (see [Figure 4 on page 39](#):

Figure 4. Fully Initialized FUSE Message Broker

```
C:\WINDOWS\system32\cmd.exe
INFO BrokerService - ActiveMQ JMS Message Broker (localhost, I
D:DavidD800-3484-1224244084810-0:0) started
INFO log - Logging to org.slf4j.impl.JCLLoggerAdapte
r(org.morthay.log) via org.morthay.log.Slf4jLog
INFO log - jetty-6.1.0.0-fuse
INFO WebConsoleStarter - ActiveMQ WebConsole initialized.
INFO /admin - Initializing Spring FrameworkServlet 'dis
patcher'
INFO log - ActiveMQ Console at http://0.0.0.0:8161/a
dmin
INFO log - ActiveMQ Web Demos at http://0.0.0.0:8161
/demo
INFO log - RESTful file access application at http://
/0.0.0.0:8161/fileserver
INFO log - WebApp@13043068 at http://0.0.0.0:8161/co
nsole
INFO log - Started SelectChannelConnector@0.0.0.0:81
61
INFO FailoverTransport - Successfully connected to tcp://localhost
:6161
```

Running the Java Server

This section presumes that you are running ActiveMQ or FUSE Message broker, which does not require you to make any changes to the code or the JNDI properties of the sample Java server.

If you are running a different JMS broker, make whatever changes to the Java server that your broker requires, as described in the relevant section of [Using Other JMS Brokers on page 22](#).

To run the Java server:

1. Open a Windows command prompt and add the ActiveMQ JAR to your classpath, as follows:

Example 16. Adding a ActiveMQ to the Classpath

```
set CLASSPATH=PathToActiveMQJar;%CLASSPATH%
```

For example, for FUSE Message Broker 5.1.0.0, run:

Example 17. Adding FUSE Message Broker 5.1.0.0 to the Classpath

```
set CLASSPATH=ActiveMQInstallDir\activemq-all-5.1.0.0-fuse.jar;%CLASSPATH%
```

2. Change directory to the following directory:

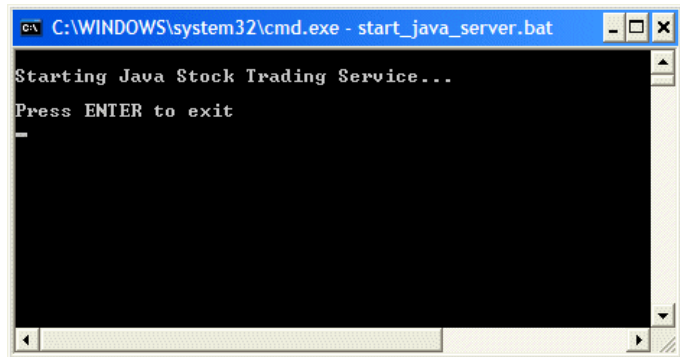
```
InstallDir\Visual Studio Adapter\samples\corba_jms\bin\
```

3. Start the Java server using the following command:

```
start_java_server.bat
```

4. Wait for the Java service to fully initialize and connect to the JMS broker; that is, until it displays the message "Press ENTER to exit" (see [Figure 5 on page 40](#)).

Figure 5. Fully Initialized Java Server



Step 2: Opening the .NET Solution

Overview

Now that you have the CORBA and JMS systems running, the next step is to get the .NET WCF stock purchase client application to talk to them.

Opening the .NET solution

To open the WCF solution:

1. Open Windows Explorer and navigate to the following directory of your Artix Connect for WCF installation:

```
InstallDir\Visual Studio Adapter\samples\corba_jms\dotnet
```

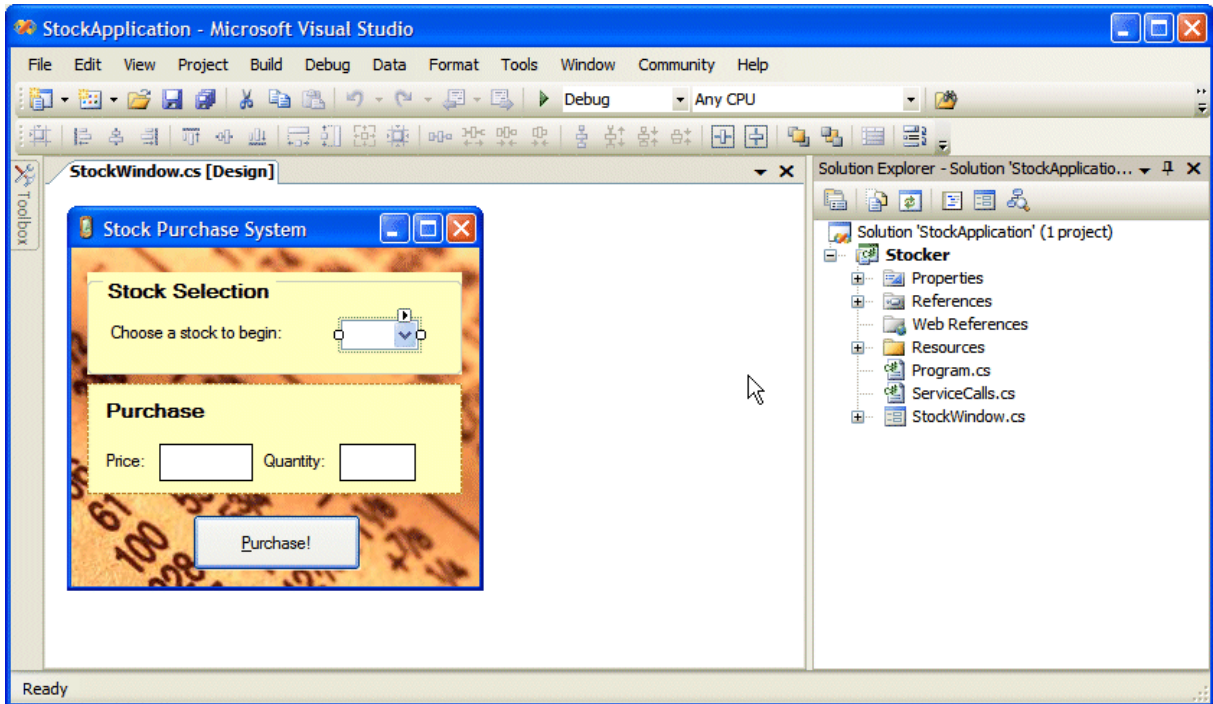
2. Double-click on the `StockApplication.sln` solution file.

This launches Visual Studio 2005 and opens the solution. The application is ready to build and run, but is not yet modified to talk to either the CORBA or JMS back-end system.

3. Open the `StockWindow.cs` file.

The .NET stock purchase application appears as shown in [Figure 6 on page 42](#):

Figure 6. .NET Stock Purchase Application



The application works as follows:

1. Choose a stock from the **Stock Selection** drop-down menu. The application makes a call to the CORBA back-end system to retrieve the price for that stock and fills in the **Price** field with the returned value.
2. Fill in the quantity of stock that you want to buy. The **Purchase** button is enabled.
3. Click **Purchase** to send the order message to a queue within the JMS broker. The order message is then consumed by the Java server.



Note

If you try using the application now, you will notice that neither the CORBA nor the JMS system are reachable. To enable the application

to communicate with the CORBA and JMS systems, you need to add the required code using Artix Connect for WCF.

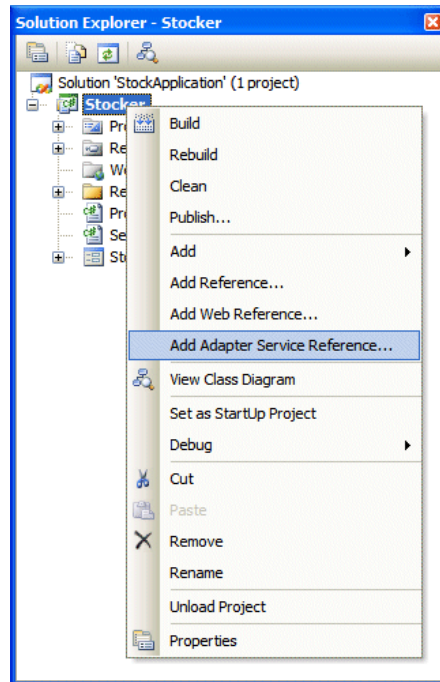
Step 3: Opening the Artix Connect for WCF wizard

Steps

To open the Artix Connect for WCF wizard:

1. In the Solution Explorer window, right-click the **Stocker** project and choose **Add Adapter Service Reference**, as shown in [Figure 7 on page 44](#).

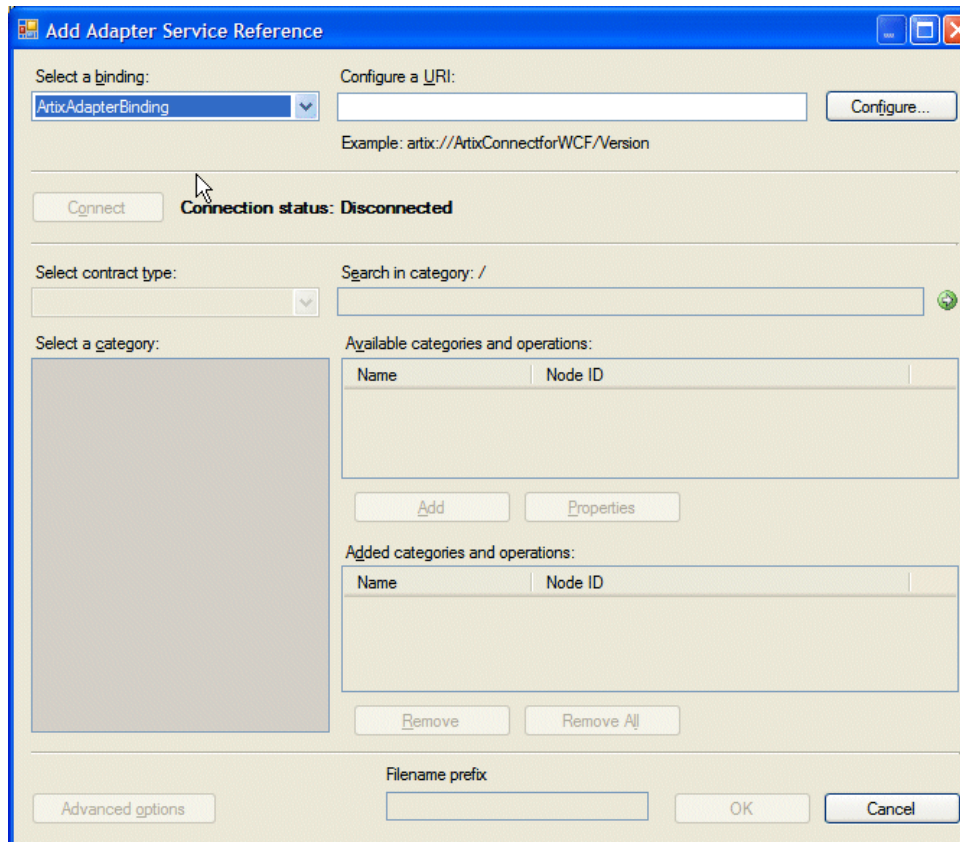
Figure 7. Adding an Adapter Service Reference



This launches the Microsoft LOB Adapter framework. Artix Connect for WCF is a plug-in to the LOB Adapter Framework.

2. In the Add Adapter Service Reference wizard, choose **ArtixAdapterBinding** from the **Select a binding** drop-down list.

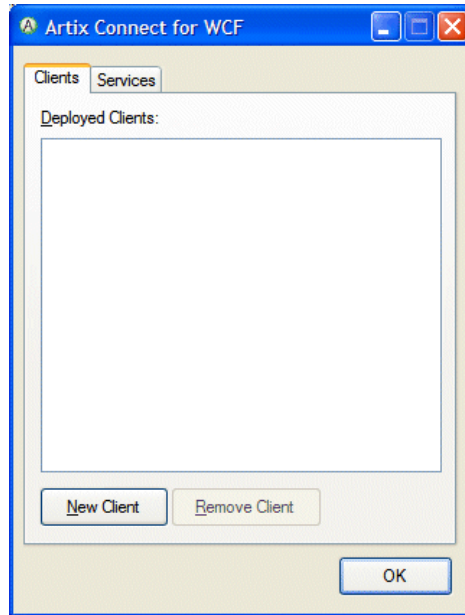
Figure 8. The Add Adapter Service Reference wizard



3. Click the **Configure** button.
4. In the Configure Adapter wizard that launches, click **OK**.
Notice that the **Connect** button is now enabled.
5. Click **Connect**.

The Artix Connect for WCF wizard opens as shown in [Figure 9 on page 46](#). The Deployed Clients list is empty because you have not yet connected to either the CORBA or JMS back-end system.

Figure 9. Artix Connect for WCF Wizard



Step 4: Using the Wizard to Connect to CORBA

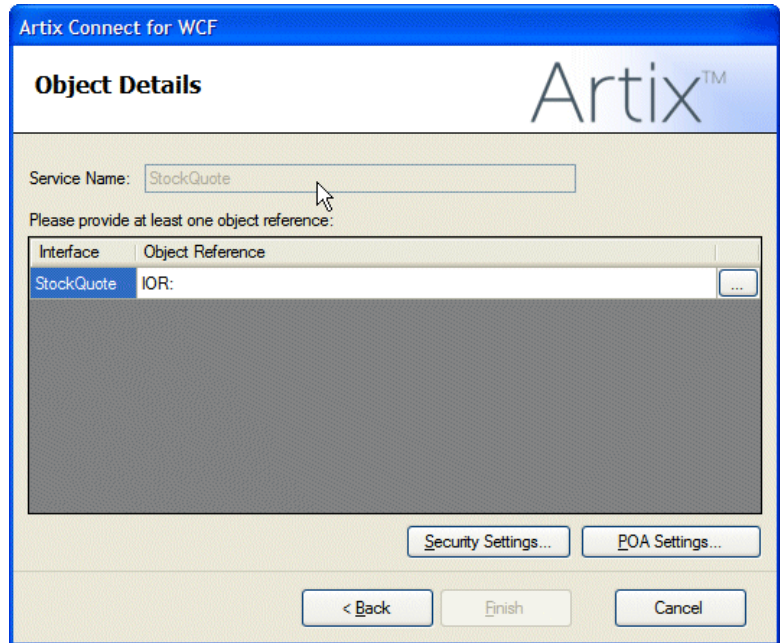
Steps

To use the Artix Connect for WCF wizard to connect to the CORBA service:

1. In the Artix Connect for WCF wizard, click **New Client**.
2. In the New Client panel, select the **CORBA** radio button and click **OK**.
3.
 - If you are creating a CORBA client for the first time, the CORBA Security Settings panel appears. As you are not creating a secure CORBA client in this sample, leave **Security Mode** set to **None** and click **Next**.
 - If you previously created a secure CORBA client, for example by running the Security sample, you need to reset the security mode in the Artix Administration tool. See [Using the Artix Administration Tool](#) in the *User's Guide* for details.
4. In the IDL File Selection panel, click **Browse**.
5. In the Select IDL File dialog box, browse to the location of the sample CORBA system IDL file, which is located in the following directory of your Artix Connect for WCF installation:

```
InstallDir\Visual Studio Adapter\samples\corba_jms\etc
```
6. Select the `StockQuote.idl` file and click **Open**.
7. In the IDL File Selection panel, click **Next**. The wizard checks that the IDL file is valid.
8. In the Object Details panel, the interface defined in the IDL file is displayed, as shown in [Figure 10 on page 48](#).

Figure 10. CORBA Object Details Window



9. To provide the CORBA service object reference, click ... and browse to the location of the sample CORBA system IOR file. It is located in the same directory as the IDL file; that is:

`InstallDir\Visual Studio Adapter\samples\corba_jms\etc`

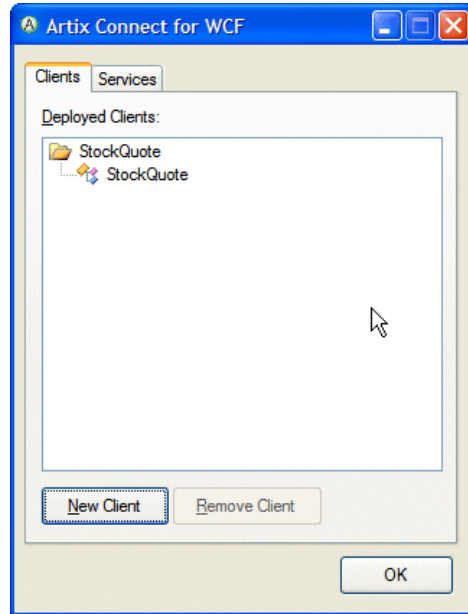
10. Select the `StockQuote.ior` file and click **Open**.

The wizard adds the IOR file to the Object Reference field of the Object Details window.

11. Click **Finish**.

In the Artix Connect for WCF wizard, the CORBA stock quote system is added to the list of deployed clients (see [Figure 11 on page 49](#)).

Figure 11. CORBA StockQuote System Added to Deployed Clients List



Step 5: Using the Wizard to Connect to JMS

Creating a JMS client

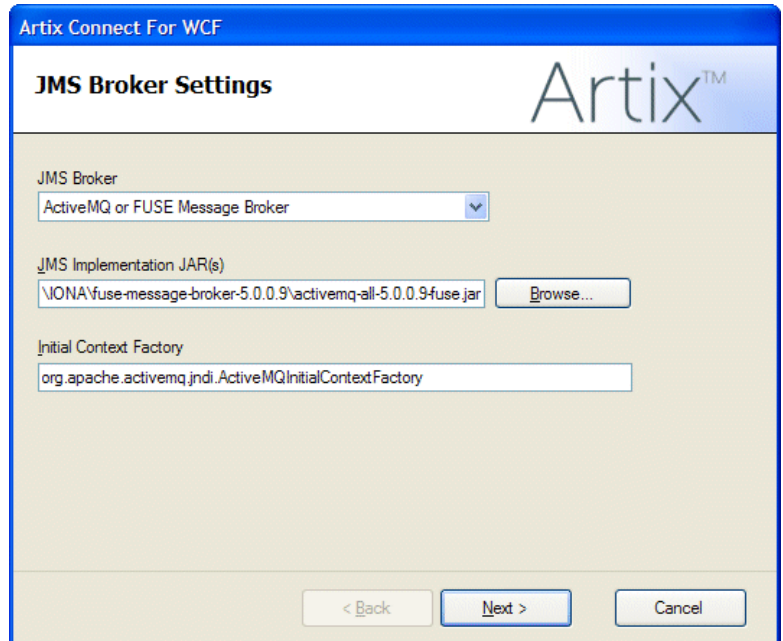
To use Artix Connect for WCF to connect the JMS system:

1. In the Artix Connect for WCF wizard, click **New Client**.
2. In the New Client panel, select the JMS radio button.
3. Click **Next**.
4.
 - If you have already chosen a JMS broker as described in [Choosing Your JMS Broker on page 19](#), go to [Selecting a payload format on page 51](#).
 - If you have not previously chosen a JMS broker, the JMS Broker Settings panel displays. Select **ActiveMQ** or **FUSE Message Broker** from the **JMS Broker** drop-down list.

Note that the Initial Context Factory is set automatically when you select a JMS broker.

5. Under **JMS Implementation JAR(s)**, click **Browse** and select the implementation JAR for the FUSE Message Broker or Apache ActiveMQ version that you are using. For example, for FUSE Message Broker 5.1.0.0, browse to the top level of the FUSE Message Broker installation directory and select the `activemq-all-5.1.0.0-fuse.jar` file.

For a complete list of JMS implementation JARs, see [JMS Broker Implementation JARs](#) in the *Installation Guide*.

Figure 12. Adding JMS Broker Settings

6. Click **Next**.

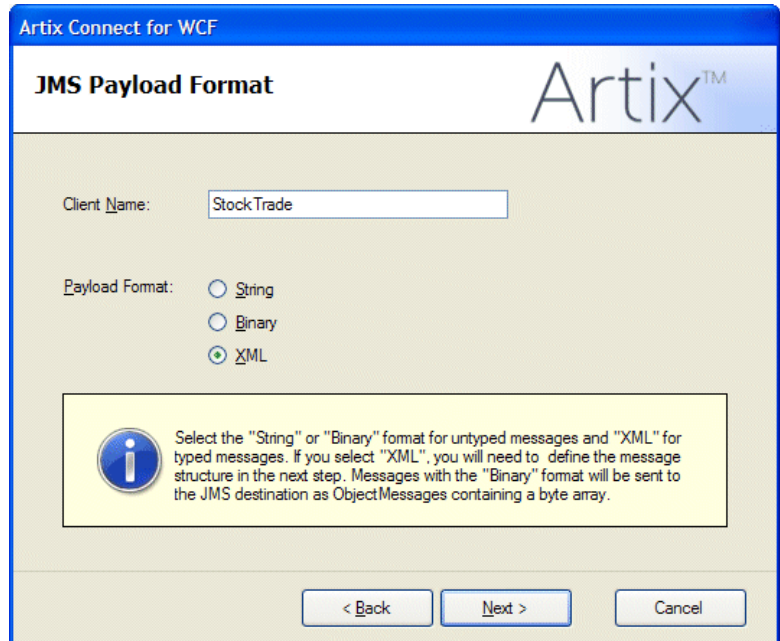
Selecting a payload format

The JMS Payload Format window enables you to give the client a name and to select the type of message that you are sending. In the sample application, the message type is XML.

To set the JMS payload format for the sample application:

1. Under Client Name, type `StockTrade`.
2. Under Payload Format, select the **XML** radio button, as shown in [Figure 13 on page 52](#).

Figure 13. Adding JMS Client Name and Payload Format Details



3. Click **Next**.

Defining the message structure

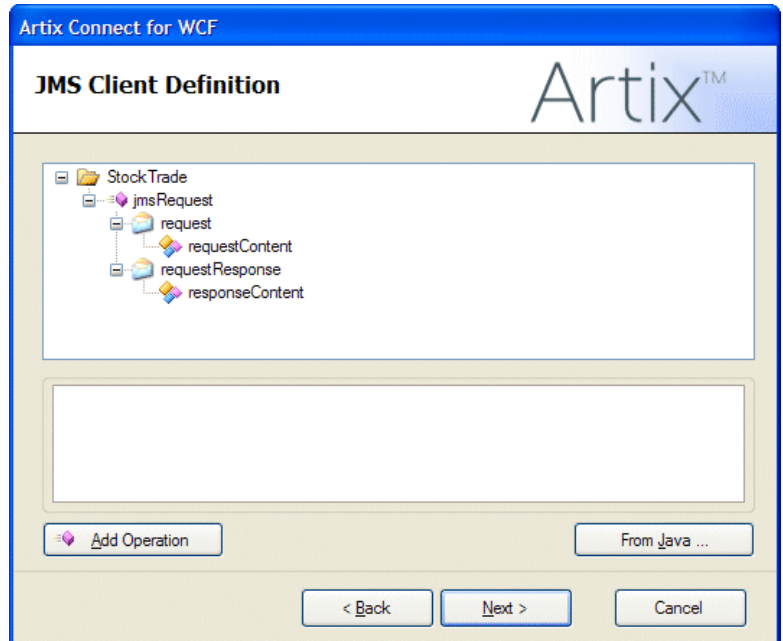
Because you are sending an XML message, you need to define the message structure. For the purposes of this tutorial, you need to use the business interface and operations defined by the sample Java server.

Although it is possible to manually add this information, using the tree and the buttons below the client panel, the sample application includes a Java class file that represents the interface that you are trying to access.

To define the message structure:

1. In the JMS Client Definition window, click **From Java**.

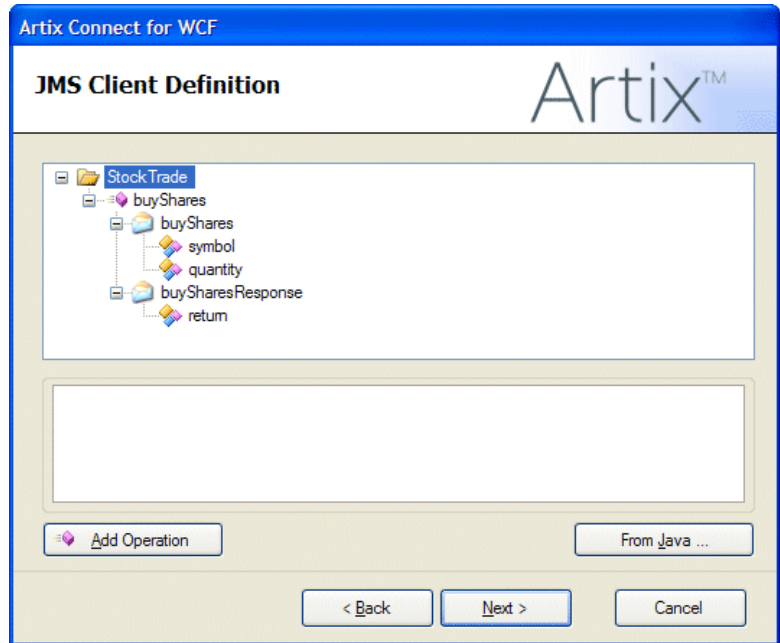
Figure 14. Defining XML Message



2. Navigate to `InstallDir\Visual Studio Adapter\samples\corba_jms\jms\bin\com\acme\stock\trade` and select the `StockTrader.class`.
3. Click **Open**.

The wizard examines the Java class and extracts the relevant interface information from it. This information is displayed in the top panel the JMS Client Definition window, as shown in [Figure 15 on page 54](#):

Figure 15. XML Message Defined



You can see the `buyShares` and `buySharesResponse` operations listed in the tree, along with the `string` and `integer` parameters that represent the stock name and quantity required, respectively.

4. Click **Next**.

Specifying JMS destination settings

In the JMS Destination Settings window you need to set JMS destination information. This information is specific to the JMS service to which you want to connect and the JMS broker that you are using. For the purposes of this tutorial, the following section provides information for use with either FUSE Message Broker or Apache ActiveMQ. If you want to use any of the other supported JMS brokers to run the sample application, refer to [Using Other JMS Brokers on page 22](#). It provides JMS destination settings for each of the supported JMS brokers.

To set JMS destination settings for the sample application:

To specify JMS destination settings:

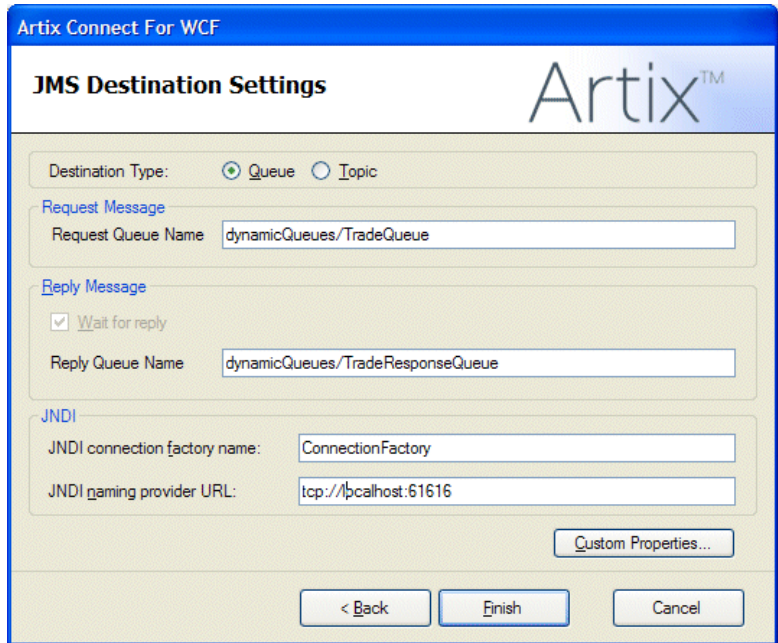
1. Fill in the JMS Destination Settings window with the values shown in [Table 5 on page 55](#):

Table 5. JMS Destination Settings for FUSE Message Broker and ActiveMQ

Setting	Value	Description
<i>Destination Type</i>	Queue	Specifies whether you are connecting to a JMS queue or topic.
<i>Request Queue Name</i>	dynamicQueues/TradeQueue	Specifies the name of the JMS queue or topic to which you are trying to connect.
<i>Reply Queue Name</i>	dynamicQueues/TradeResponseQueue	Specifies the name of the response JMS queue or topic to which you are trying to connect.
<i>JNDI connection factory name</i>	ConnectionFactory	Specifies the name of the JMS broker connection factory.
<i>JNDI naming provider URL</i>	tcp://localhost:61616	Specifies the URL used to locate and connect to the JMS broker.

When you have finished, the JMS Destinations Settings screen should appear as shown in [Figure 16 on page 56](#).

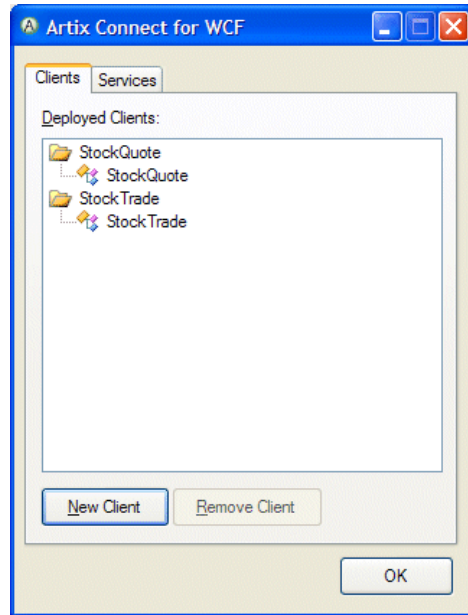
Figure 16. JMS Destinations Settings



2. Click **Finish**.

The wizard completes its tasks and returns to the original starting window. Notice, however, that both the CORBA and JMS clients are listed under Deployed Clients, as shown in [Figure 17 on page 57](#).

Figure 17. CORBA and JMS Clients Successfully Deployed



3. Click **OK**.

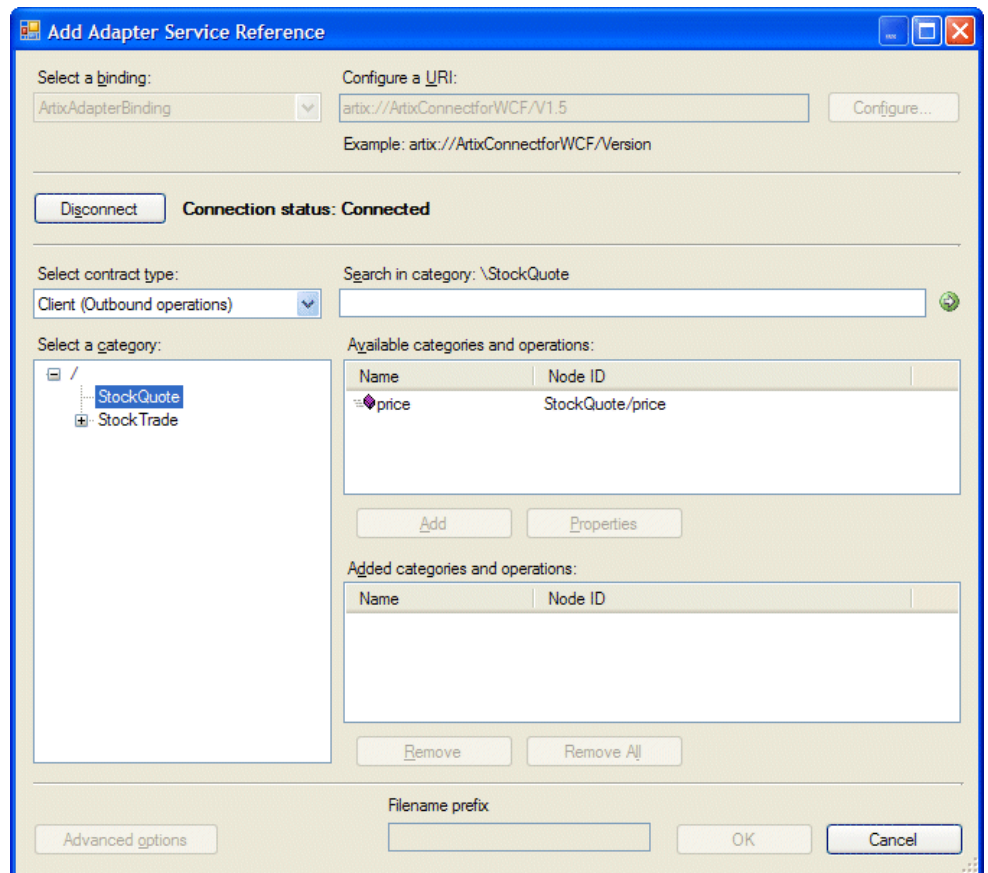
The wizard completes and returns to the LOB adapter window. It may take a few moments for the LOB adapter window to become responsive again while the JMS and CORBA system details are processed.

Step 6: Making CORBA and JMS Operations Available to Your WCF Application

Introduction

After a few moments, the LOB Adapter window will look similar to [Figure 18 on page 58](#), with the `StockTrade` and `StockQuote` entries listed in the **Select a category:** panel.

Figure 18. JMS and CORBA details in the LOB Adapter Window



The **OK** button is disabled. It will remain so until you specify which operations you want to use within your WCF application code.

Choosing CORBA operations

To choose a CORBA operation, complete the following steps:

1. In the Add Adapter Service Reference wizard, under the **Select a category** panel, select the `StockQuote` category.
 2. In the **Available categories and operations** panel, select the `price` operation.
 3. Click **Add** to add the `price` operation to the **Added categories and operations** panel.
-

Choosing JMS operations

To choose a JMS operation, complete the following steps:

1. In the Add Adapter Service Reference wizard, under the **Select a category** panel, select the `StockTrade` category.
2. In the **Available categories and operations** panel, select the `buyShares` operation.
3. Click **Add** to add the `buyShares` operation to the **Added categories and operations** panel.
4. Click **OK**.

The wizard starts to generate code and configuration to enable your WCF application to use these operations.

Step 7: Adding Code to Call to the CORBA and JMS Systems

Introduction

You will notice after clicking the **OK** button that your project has some new files in it, and also that your Visual Studio IntelliSense offers new symbols relating to the CORBA and JMS operations that you just added. Your project has been modified to include new code that presents the CORBA and JMS systems as native WCF endpoints. The code to interact with these systems is both simple and identical.

Steps

To add the code to call the CORBA and JMS systems to your WCF application, complete the following steps:

1. Open the `ServiceCalls.cs` file. This file encapsulates all of the required remote calls that the WCF application needs to make in order to become functional.
2. Uncomment the two commented sections of code in the `GetPrice()` and `PlaceOrder()` operations, as shown in [Example 18 on page 60](#).

Example 18. `ServiceCalls.cs` after modification

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Windows.Forms;

namespace StockApplication
{
    class ServiceCalls
    {
        public static double GetPrice(String symbol)
        {
            double price = 0.0;

            StockQuoteClient quoter = new StockQuoteClient();
            price = quoter.price(symbol);

            return price;
        }

        public static void PlaceOrder(String symbol, int quantity)
```

```
{  
    StockTradeClient purchaser = new StockTradeClient();  
    string confirmation = purchaser.buyShares(symbol, quantity);  
    MessageBox.Show(confirmation,  
        Application.ProductName,  
        MessageBoxButtons.OK,  
        MessageBoxIcon.Information);  
}  
}
```

As you can see, very little work is required to call the CORBA and JMS systems. They are presented just like native WCF endpoints and are just as easy to use.

Step 8: Running the Stock Purchasing Application

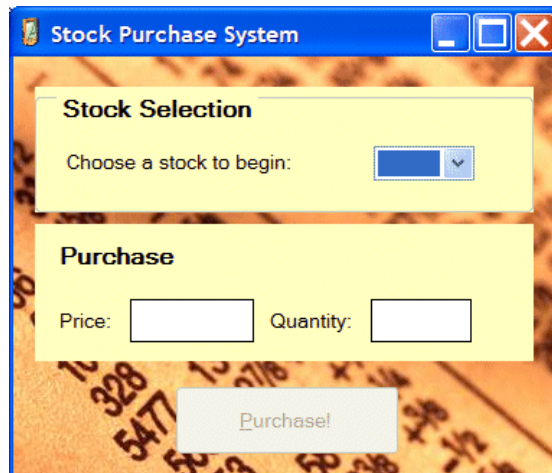
Introduction

You have now done everything needed to allow the WCF application to communicate with both the CORBA and the JMS back-end systems. All that is left for you to do is to build, run and play with the WCF application.

Playing with the application

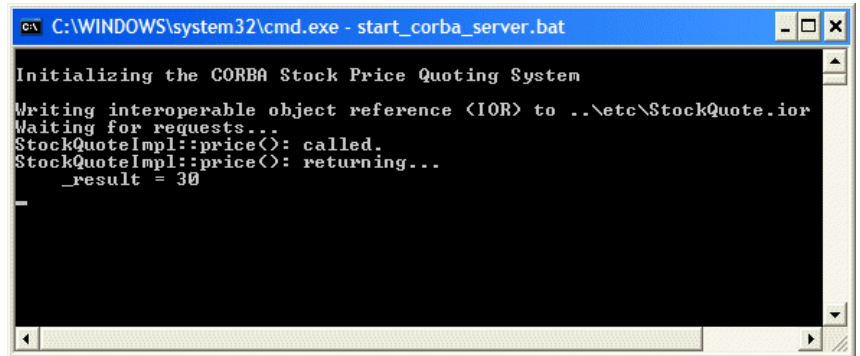
When you build and run the application, it should appear as shown in [Figure 19 on page 62](#).

Figure 19. The Completed WCF Application



Try the following, for example:

1. Under Stock Selection, select **MSFT** from the drop-down list.
2. You will see a price appear in the **Price** field. This is the price that was returned from the CORBA server. If you examine the CORBA server window, you will see that the request has been logged (see [Figure 20 on page 63](#)).

Figure 20. CORBA Server Logging an Operation Call

```
C:\WINDOWS\system32\cmd.exe - start_corba_server.bat

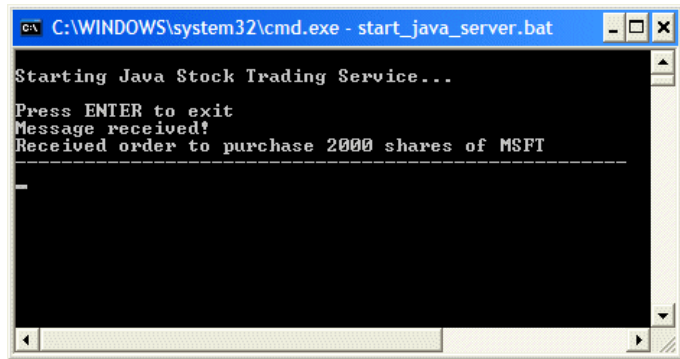
Initializing the CORBA Stock Price Quoting System
Writing interoperable object reference (IOR) to ..\etc\StockQuote.ior
Waiting for requests...
StockQuoteImpl::price(): called.
StockQuoteImpl::price(): returning...
_result = 30
```

3. Now that you have the price of the stock, you can purchase the stock. Type in a quantity, for example, 2000, and click **Purchase!**, as shown in [Figure 21 on page 63](#).

Figure 21. Running the Completed Stock Purchase Application

You will see the order being consumed by the Java server, as shown in [Figure 22 on page 64](#).

Figure 22. Java Server Consuming JMS Request



```
C:\WINDOWS\system32\cmd.exe - start_java_server.bat
Starting Java Stock Trading Service...
Press ENTER to exit
Message received!
Received order to purchase 2000 shares of MSFT
-----
-

```


Index

Symbols

.NET solution
opening, 41

A

ActiveMQ
setup, 21
Apache ActiveMQ
JNDI connection factory name, 54
JNDI naming provider URL, 54
Artix Connect for WCF wizard
connecting to CORBA, 47
connecting to JMS, 50
opening, 44

B

BEA WebLogic
Destination Type, 35
JMS destination settings, 35
JNDI connection factory name, 35
JNDI naming provider URL, 35
Reply Queue Name, 35
Request Queue Name, 35
setup, 34
starting, 36

C

CORBA
adding code to call, 60
CORBA operations
making available to WCF, 58
CORBA sample, 16
CORBA server
running, 38
CORBA stock quote system, 17

D

Destination Type

BEA WebLogic, 35
sample application, 54
SonicMQ, 29
TIBCO EMS, 24
WebSphere MQ, 32

F

FUSE Message Broker
JNDI connection factory name, 54
JNDI naming provider URL, 54

J

Java server
running, 39
JMS
adding code to call, 60
JMS broker
choosing, 19
running, 38
JMS destination settings
BEA WebLogic, 35
SonicMQ, 29
TIBCO EMS, 24
WebSphere MQ, 32
JMS operations
making available to WCF, 58
JMS sample, 16
JMS stock purchase system
introduction to, 17
JNDI connection factory name
Apache ActiveMQ, 54
BEA WebLogic, 35
FUSE Message Broker, 54
SonicMQ, 29
TIBCO EMS, 24
WebSphere MQ, 32
JNDI naming provider URL
Apache ActiveMQ, 54
BEA WebLogic, 35
FUSE Message Broker, 54
SonicMQ, 29
TIBCO EMS, 24
WebSphere MQ, 32

R

- Reply Queue Name
 - BEA WebLogic, 35
 - sample application, 54
 - SonicMQ, 29
 - TIBCO EMS, 24
 - WebSphere MQ, 32
- Request Queue Name
 - BEA WebLogic, 35
 - sample application, 54
 - SonicMQ, 29
 - TIBCO EMS, 24
 - WebSphere MQ, 32

S

- Sample application
 - Destination Type, 54
 - how it works, 42
 - Reply Queue Name, 54
 - Request Queue Name, 54
- ServiceCalls.cs, 60
- SonicMQ
 - configuring for JMS, 26
 - Destination Type, 29
 - JMS destination settings, 29
 - JNDI connection factory name, 29
 - JNDI naming provider URL, 29
 - Reply Queue Name, 29
 - Request Queue Name, 29
 - setup, 26
 - starting, 29
- stock purchasing application
 - running, 62

T

- TIBCO EMS
 - Destination Type, 24
 - JMS destination settings, 24
 - JNDI connection factory name, 24
 - JNDI naming provider URL, 24
 - Reply Queue Name, 24
 - Request Queue Name, 24

- setup, 23
- starting, 25

W

- WebSphere MQ
 - configuring for JMS, 30
 - Destination Type, 32
 - JMS destination settings, 32
 - JNDI connection factory name, 32
 - JNDI naming provider URL, 32
 - Reply Queue Name, 32
 - Request Queue Name, 32
 - setup, 30
 - starting, 33