Artix Connect for WCF

Getting Started Guide

Version 1.0
May 2008

Making Software Work Together™

Getting Started Guide

IONA Technologies

Version 1.0

Published 22 May 2008

Trademark and Disclaimer Notice

Copyright Notice

Table of Contents

Preface **11**
The Artix Connect for WCF Library 12
Document Conventions 13
Introduction to the Sample Application **15**
CORBA and JMS Sample Application 16
Setting up Your JMS Broker 19
Introduction 20
TIBCO EMS 23
SonicMQ 7.5 26
WebSphere MQ 6.0 30
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 43
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 59
Step 7: Adding Code to Call to the CORBA and JMS Systems 62
Step 8: Running the Stock Purchasing Application 64
Index 67

List of Figures

1. Sample Application Architecture 16
2. Configuring a JMS Broker 21
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 41
7. Adding an Adapter Service Reference 43
8. Add Adapter Service Reference Wizard 44
9. Selecting ArtixAdapterBinding 45
10. Artix Connect for WCF Wizard 46
11. CORBA Object Details Window 48
12. CORBA StockQuote System Added to Deployed Services List 49
13. Adding JMS Broker Settings 51
14. Adding JMS Service Name and Payload Format Details 53
15. Defining XML Message 54
16. XML Message Defined 55
17. JMS Destinations Settings 57
18. CORBA and JMS Services Successfully Deployed 58
19. JMS and CORBA details in the LOB Adapter Window 60
20. The Completed WCF Application 64
21. CORBA Server Logging an Operation Call 65
22. Running the Completed Stock Purchase Application 66
23. Java Server Consuming JMS Request 66

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 56

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. TIBCO EMS: Starting Java Server 24
6. Sample Java Server: jndi.properties for SonicMQ 27
7. Sample Java Server Constructor Code for SonicMQ 28
8. SonicMQ: Starting Java Server 28
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. WebSphere MQ: Starting Java Server 32
13. Sample Java Server: jndi.properties for BEA WebLogic 34
14. Sample Java Server Constructor Code for BEA WebLogic 34
15. BEA WebLogic: Starting Java Server 35
16. Starting Java Server 39
17. FUSE Message Broker: Starting Java Server 39
18. ServiceCalls.cs after modification 62

Preface

Table of Contents

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
  [http://www.iona.com/support/docs/artix/connectwcf/1.0/install_guide/index.html]
- Release Notes
  [http://www.iona.com/support/docs/artix/connectwcf/1.0/release_notes/index.html]
- Getting Started Guide
  [http://www.iona.com/support/docs/artix/connectwcf/1.0/tutorial/index.html]
- User's Guide
  [http://www.iona.com/support/docs/artix/connectwcf/1.0/users_guide/index.html]

Document Conventions

Typographical conventions

This book uses the following typographical conventions:

`fixed width`	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 `javax.xml.ws.Endpoint` class. Constant width paragraphs represent code examples or information a system displays on the screen. For example: `import java.util.logging.Logger;`
`Fixed width italic`	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: `% cd /users/`*`YourUserName`*
Italic	Italic words in normal text represent *emphasis* and introduce *new terms*.
Bold	Bold words in normal text represent graphical user interface components such as menu commands and dialog boxes. For example, the **User Preferences** dialog.

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.
`>`	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 might 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 crucial 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 might cause damage to your systems. Possible damage from these errors include system failures and loss of data.

Introduction to the Sample Application

Summary

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.

Table of Contents

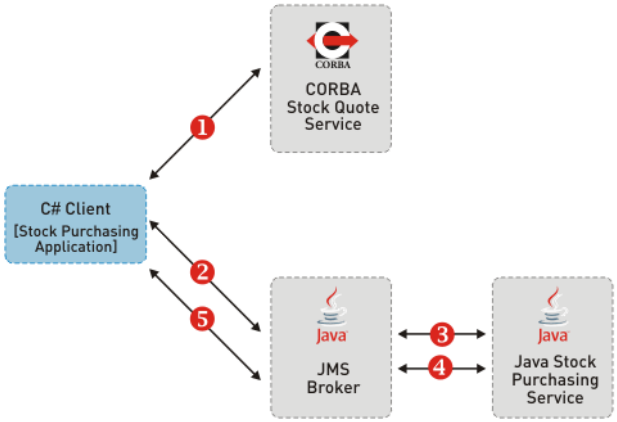
CORBA and JMS Sample Application 16
Setting up Your JMS Broker 19
- Introduction 20
- TIBCO EMS 23
- SonicMQ 7.5 26
- WebSphere MQ 6.0 30
- BEA WebLogic 10 34

CORBA and JMS Sample Application

Introduction

Artix Connect for WCF includes a ready-to-run sample application that demonstrates 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, “Sample Application Architecture” and the text that follows explain, at a high-level, how the sample application works.

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 purchase, 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, “Stock Quote System—IDL”. 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 purchase 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. JMS architects can, for example, 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 is capable of following these encoding rules and presenting 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.

Setting up Your JMS Broker

Table of Contents

Introduction 20
TIBCO EMS 23
SonicMQ 7.5 26
WebSphere MQ 6.0 30
BEA WebLogic 10 34

Introduction

Introduction

The sample application runs by default against either Apache ActiveMQ or FUSE Message Broker (which is an open source JMS broker based on Apache ActiveMQ). All you have to do is make sure that you have one of them installed on your machine.

If you do not, but want to use one of these brokers, download and install FUSE Message Broker from the following website:

http://open.iona.com/downloads

Install using all of the default settings.

Using one of the other JMS brokers

If you want to use a JMS broker other than FUSE Message Broker or Apache ActiveMQ, you must also:

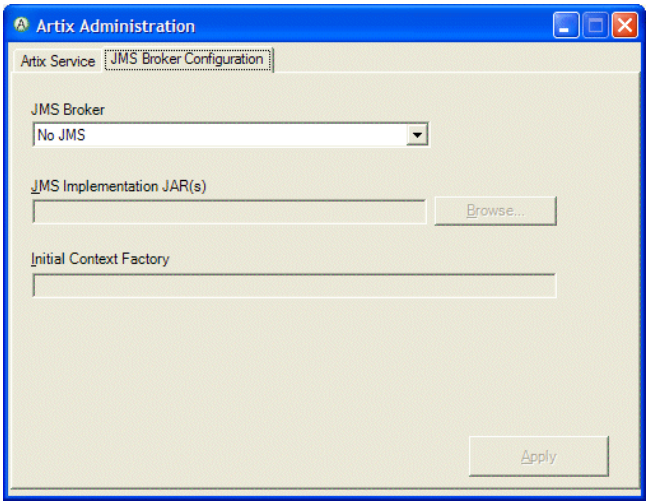
- Use the Artix Administration tool to configure your system to use your chosen JMS broker:

  1. Open the Artix Administration tool from the Windows Start menu as follows:

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

  2. Select the JMS Broker Configuration tab, as shown in Figure 2, “Configuring a JMS Broker”:

Figure 2. Configuring a JMS Broker



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

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

4. In the JMS Implementation JAR(s) field, enter the location of the JMS implementation JAR(s) for the broker that you selected in step 3.

   For a complete list of JMS implementation JARs, see JMS Broker Implementation JARs in *Installation Guide*.

5. Click Apply.

- Change some of the settings and code used in 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 subsections describe the changes that you have to make to the sample Java server, provide the JMS broker and destination settings, and give starting instructions for each of the other supported JMS brokers:

- TIBCO EMS
- SonicMQ 7.5
- WebSphere MQ 6.0
- BEA WebLogic 10

TIBCO EMS

Updating the sample Java server

If you want to use TIBCO EMS with the sample application, please make the following changes to the sample Java server:

1. Update the `jndi.properties` file as follows:

   i. 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\jms`

   ii. 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
```

   iii. Save the changes that you have made to the `jndi.properties` file.

2. Change the Java server constructor code and rebuild as follows:

   i. 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\jms\src\com\acme\stock\trade\jms`

   ii. 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");
```

iii. Build the Java server by:

a. Navigating to the following directory of your Artix Connect for WCF installation:

InstallDir\Visual Studio Adapter\samples\corba_jms\jms

b. Running the `buildjava.bat` file.

iv. Start the Java server by adding the TIBCO EMS JMS implementation JARs to your `CLASSPATH` and running the `start_java_server.bat` file as follows:

A. Open a Windows command prompt

B. Run the following command:

Example 5. TIBCO EMS: Starting Java Server

```
set CLASSPATH=TIBCOEMSInstallDir\clients\java\tibjms.jar;
TIBCOEMSInstallDir\clients\java\jms.jar;%CLASSPATH%
InstallDir\Visual Studio Adapter\samples\corba_jms\bin\start_java_server.bat
```

Configuring JMS Destination Settings

When working through the tutorial, in Step 5: Using the Wizard to Connect to JMS, you are asked to provide JMS destination settings. In the JMS Destination Settings window, enter the settings shown in Table 1, “JMS Destination Settings for TIBCO EMS”:

Table 1. JMS Destination Settings for TIBCO EMS

Setting	Value
Destination Type	`Queue`
Request Queue Name	`queue.sample`
Reply Queue Name	`queue.sample1`

Setting	Value
JNDI connection factory name	`QueueConnectionFactory`
JNDI naming provider URL	`tcp://localhost:7222`

Starting the JMS broker

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.

SonicMQ 7.5

Configuring SonicMQ for JMS

Please refer to you 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:

   i. Notice that the Context factory is filled in automatically.

   ii. `Domain = Domain1`

   iii. `Provider URL = localhost`

   iv. 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:

   i. Under the General tab:

      a. In the Lookup name and factory type, enter:
      `QueueConnectionFactory`

b. In Connection Url, enter: `tcp://localhost:2506`

ii. Click Update.

iii. 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:

i. Click New.

ii. Choose Lookup Name `Sample1`

iii. 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

If you want to use SonicMQ 7.5 with the sample application, please make the following changes to the sample Java server:

1. Update the `jndi.properties` file as follows:

i. 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\jms

ii. 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
```

iii. Save the changes that you have made to the `jndi.properties` file.

2. Change the Java server constructor code and rebuild as follows:

   i. 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\jms\src\com\acme\stock\trade\jms

   ii. 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");
```

   iii. Build the Java server by:

      a. Navigating to the following directory of your Artix Connect for WCF installation:

         InstallDir\Visual Studio Adapter\samples\corba_jms\jms

      b. Running the `buildjava.bat` file.

   iv. Start the Java server by adding the SonicMQ JMS implementation JARs to your `CLASSPATH` and running the `start_java_server.bat` file as follows:

      A. Open a Windows command prompt

      B. Run the following command:

Example 8. SonicMQ: Starting Java Server

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

```
SonicMQInstallDir\MQVersion\lib\sonic_XA.jar;
SonicMQInstallDir\wizard.jar;%CLASSPATH%
InstallDir\Visual Studio Adapter\samples\corba_jms\bin\start_java_server.bat
```

Configuring JMS Destination Settings

When working through the tutorial, in Step 5: Using the Wizard to Connect to JMS, you are asked to provide JMS destination settings. In the JMS Destination Settings window, enter the settings shown in Table 2, “JMS Destination Settings for SonicMQ”:

Table 2. JMS Destination Settings for SonicMQ

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

Starting the JMS broker

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.

WebSphere MQ 6.0

Configuring WebSphere MQ JMS Broker

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

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
```

2. From your working directory, run the following command:

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

3. 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

If you want to use WebSphere MQ 6.0 with the sample application, please make the following changes to the sample Java server:

1. Update the `jndi.properties` file as follows:

   i. 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\jms

   ii. 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
```

   iii. Save the changes that you have made to the `jndi.properties` file.

2. Change the Java server constructor code and rebuild as follows:

   i. 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\jms\src\com\acme\stock\trade\jms

ii. 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");
```

iii. Build the Java server by:

a. Navigating to the following directory of your Artix Connect for WCF installation:

InstallDir\Visual Studio Adapter\samples\corba_jms\jms

b. Running the `buildjava.bat` file.

iv. Start the Java server by adding the WebSphere MQ JMS implementation JAR to your `CLASSPATH` and running the `start_java_server.bat` file as follows:

A. Open a Windows command prompt

B. Run the following command:

Example 12. WebSphere MQ: Starting Java Server

```
set CLASSPATH=WebSphereMQInstallDir\java\lib\com.ibm.mqjms.jar;%CLASSPATH%
InstallDir\Visual Studio Adapter\samples\corba_jms\bin\start_java_server.bat
```

Configuring JMS Destination Settings

When working through the tutorial, in Step 5: Using the Wizard to Connect to JMS, you are asked to provide JMS destination settings. In the JMS

Destination Settings window, enter the settings shown in Table 3, “JMS Destination Settings for WebSphere MQ”. The settings shown are example values taken from the queue that you created in Configuring WebSphere MQ JMS Broker. 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
Destination Type	`Queue`
Request Queue Name	`TradeQueue`
Reply Queue Name	`TradeResponseQueue`
JNDI connection factory name	`QueueConnectionFactory`
JNDI naming provider URL	`file:%IT_ARTIX_WCF_DIR%\Visual Studio Adapter\samples\wcf\corba_jms\jms`

Starting the JMS broker

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.

BEA WebLogic 10

Updating the sample Java server

If you want to use BEA WebLogic 10 with the sample application, please make the following changes to the sample Java server:

1. Update the `jndi.properties` file as follows:

   i. 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\jms`

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

Example 13. Sample Java Server: jndi.properties for BEA WebLogic

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

   iii. Save the changes that you have made to the `jndi.properties` file.

2. Change the Java server constructor code and rebuild as follows:

   i. 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\jms\src\com\acme\stock\trade\jms`

   ii. 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");
```

iii. Build the Java server by:

a. Navigating to the following directory of your Artix Connect for WCF installation:

InstallDir\Visual Studio Adapter\samples\corba_jms\jms

b. Running the `buildjava.bat` file.

iv. Start the Java server by adding the BEA WebLogic JMS implementation JAR to your `CLASSPATH` and running the `start_java_server.bat` file as follows:

A. Open a Windows command prompt

B. Run the following command:

Example 15. BEA WebLogic: Starting Java Server

```
set CLASSPATH=BEAWebLogicInstallDir\server\lib\weblogic.jar;%CLASSPATH%
InstallDir\Visual Studio Adapter\samples\corba_jms\bin\start_java_server.bat
```

Configuring JMS Destination Settings

When working through the tutorial, in Step 5: Using the Wizard to Connect to JMS, you are asked to provide JMS destination settings. In the JMS Destination Settings window, enter the settings shown in Table 4, “JMS Destination Settings for BEA WebLogic”.

Table 4. JMS Destination Settings for BEA WebLogic

Setting	Value
Destination Type	`Queue`
Request Queue Name	`weblogic.examples.jms.exampleQueue`
Reply Queue Name	`weblogic.examples.jms.exampleQueue2`
JNDI connection factory name	`weblogic.examples.jms.QueueConnectionFactory`

Setting	Value
JNDI naming provider URL	`t3://localhost:7001`

Starting the JMS broker

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

Summary

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

Table of Contents

Step 1: Running the Back-end Services 38
Step 2: Opening the .NET Solution 41
Step 3: Opening the Artix Connect for WCF wizard 43
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 59
Step 7: Adding Code to Call to the CORBA and JMS Systems 62
Step 8: Running the Stock Purchasing Application 64

Step 1: Running the Back-end Services

Overview

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

Running the CORBA service

To run the CORBA service:

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

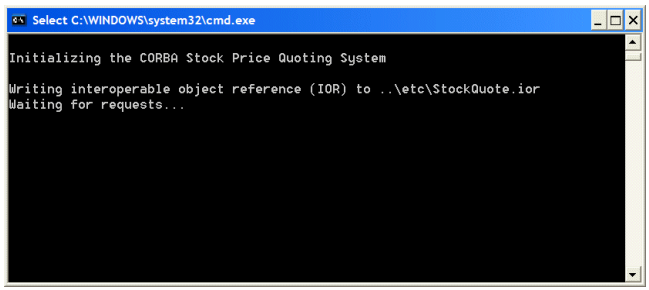
   ```
   InstallDir\Visual Studio Adapter\samples\corba_jms\bin
   ```

2. Double-click on the `start_corba_server.exe` 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, “CORBA Server Ready and Waiting for Requests”:

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, please refer to your JMS broker vendor documentation for instructions on how to run your JMS broker.

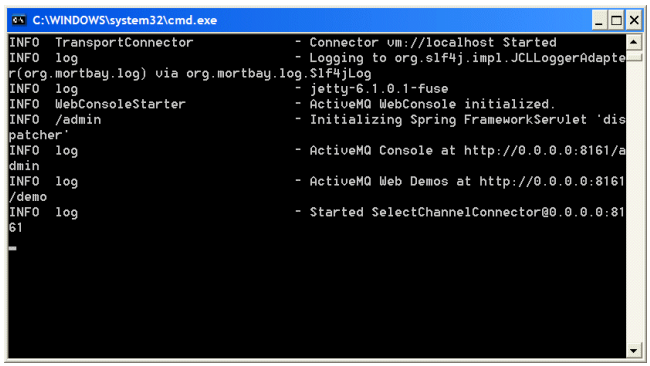
1. Open a Windows Explorer window and navigate to the following directory of your FUSE Message Broker or Apache ActiveMQ installation:

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, “Fully Initialized FUSE Message Broker”:

Figure 4. Fully Initialized FUSE Message Broker



Running the Java Server

To run the Java server, add the JMS implementation JAR to your `CLASSPATH` and run the `start_java_server.bat` file as described below:

1. Open a Windows command prompt and run the following command:

Example 16. Starting Java Server

```
set CLASSPATH=MyBroker.jar;%CLASSPATH%
InstallDir\Visual Studio Adapter\samples\corba_jms\bin\start_java_server.bat
```

For FUSE Message Broker 5.0.0.9, for example, run:

Example 17. FUSE Message Broker: Starting Java Server

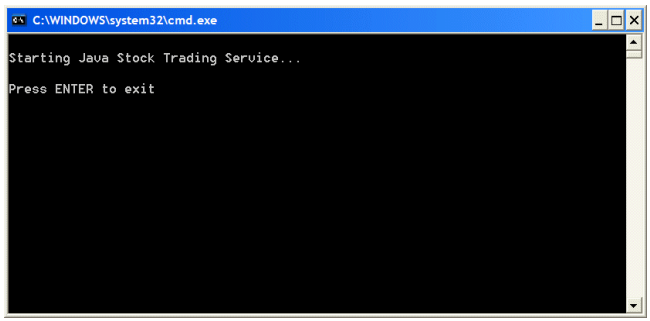
```
set CLASSPATH=FUSEInstallDir\activemq-all-5.0.0.9-fuse.jar;%CLASSPATH%
```

```
InstallDir\Visual Studio Adapter\samples\corba_jms\bin\start_java_server.bat
```

For a list of JMS implementation JARs, see JMS Broker Implementation JARs in *Installation Guide*:

2. 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, “Fully Initialized Java Server”).

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

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

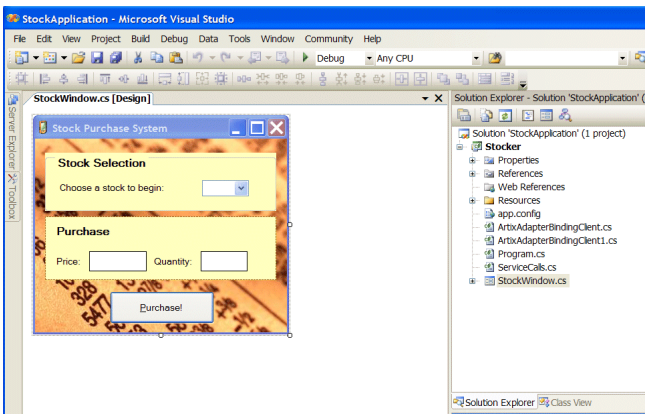
2. Double-click on the `StockApplication.sln` solution file.

   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, ".NET Stock Purchase Application":

Figure 6. .NET Stock Purchase Application



The application works very simply. When you 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. Then, when you fill in the quantity of stock that you want to buy, the Purchase button is enabled. Clicking Purchase sends 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.

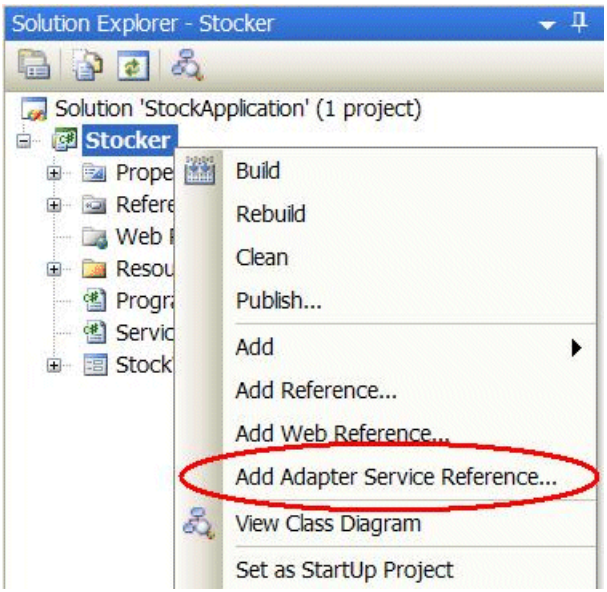
Step 3: Opening the Artix Connect for WCF wizard

Steps

To open the Artix Connect for WCF wizard:

1. Within the Solution Explorer window, right-click on the Stocker project and choose Add Adapter Service Reference… from the context menu, as shown in Figure 7, “Adding an Adapter Service Reference”.

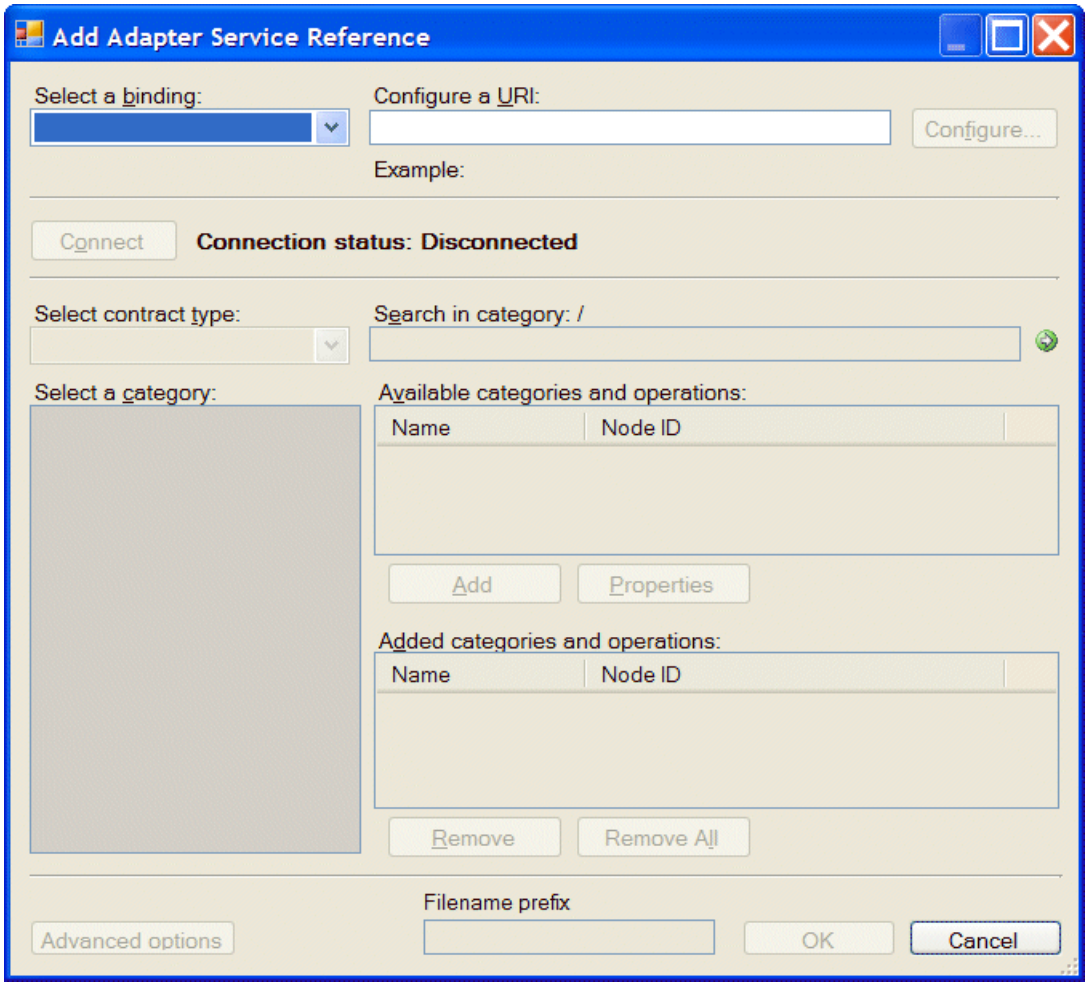
Figure 7. Adding an Adapter Service Reference



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

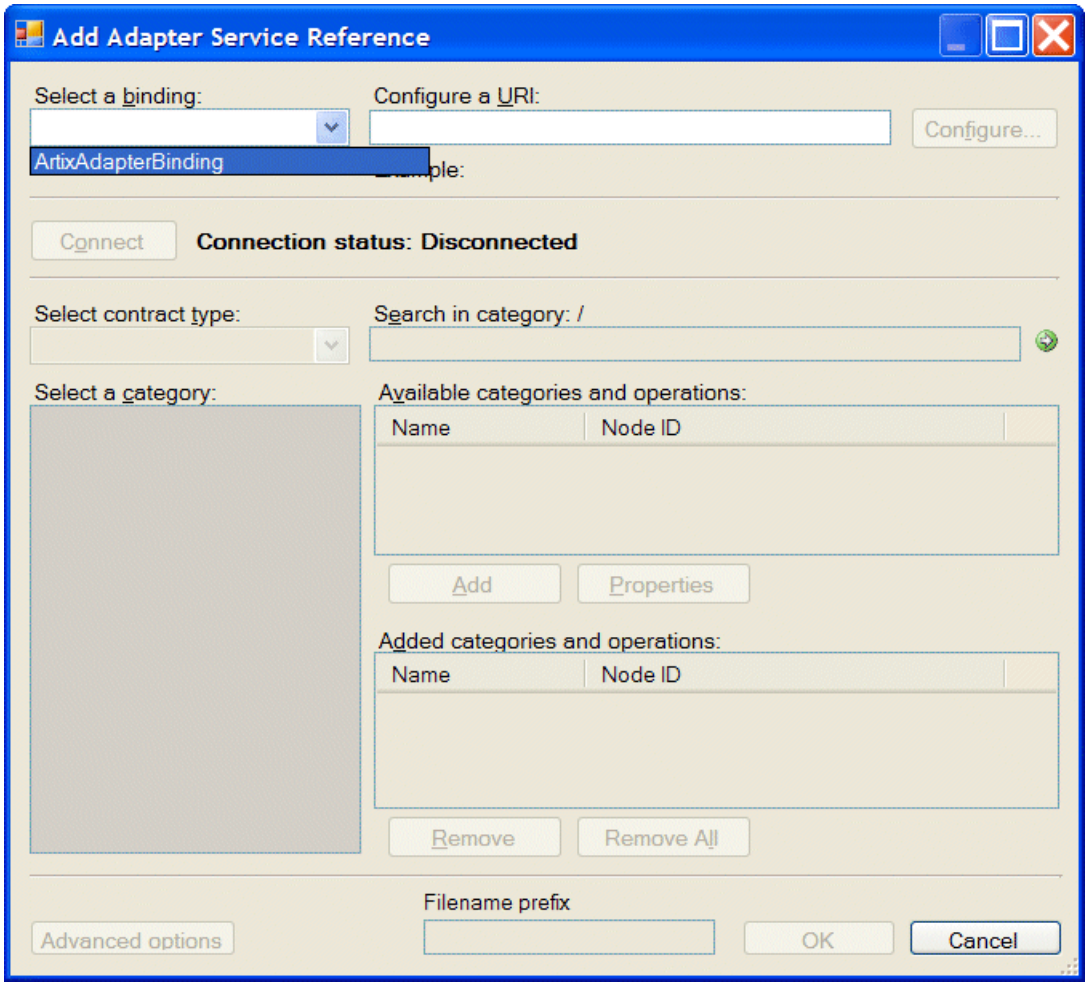
2. In the Add Adapter Service Reference wizard, shown in Figure 8, “Add Adapter Service Reference Wizard”:

Figure 8. Add Adapter Service Reference Wizard



i. In the Select a binding field, choose ArtixAdapterBinding from the drop-down list of bindings, as shown in Figure 9, “Selecting ArtixAdapterBinding”.

Figure 9. Selecting ArtixAdapterBinding



ii. Click the Configure... button.

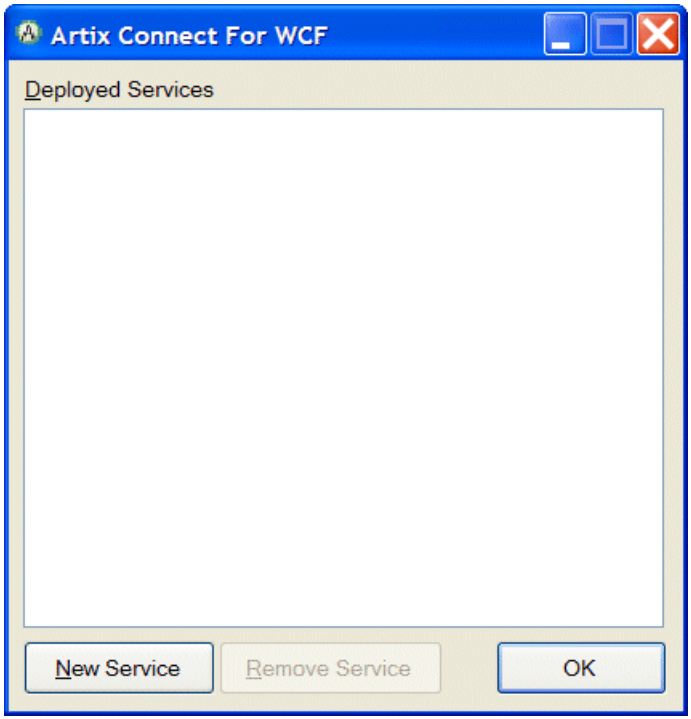
iii. In the Configure Adapter wizard that launches, click OK.

Notice that the Connect button is now enabled.

iv. Click Connect.

The Artix Connect for WCF wizard opens as shown in Figure 10, “Artix Connect for WCF Wizard”. The Deployed Services list is empty because you have not yet connected to either the CORBA or JMS back-end system.

Figure 10. Artix Connect for WCF Wizard



Step 4: Using the Wizard to Connect to CORBA

Steps

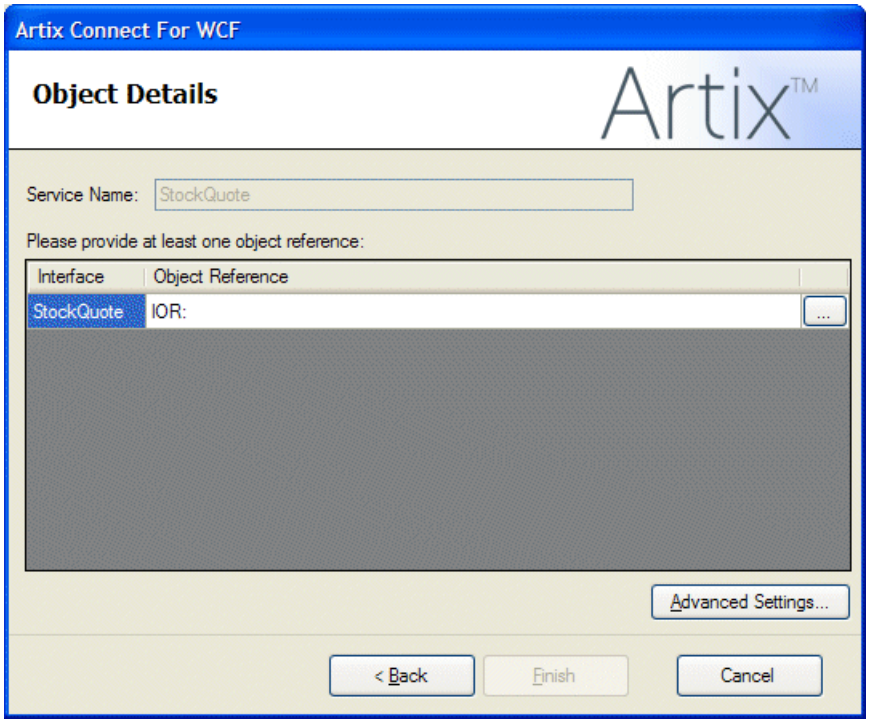
To use the Artix Connect for WCF wizard to connect to the CORBA system:

1. In the Artix Connect for WCF wizard, click New Service.
2. In the New Service window, select the CORBA radio button and click OK.
3. In the IDL File Selection window, click Browse.
4. 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\wcf\corba_jms\etc
   ```

5. Select the `StockQuote.idl` file and click Open.
6. In the Select IDL File Selection window, click Next. The wizard checks that the IDL file is valid.
7. In the Object Details window, the interface defined in the IDL file is displayed, as shown in Figure 11, “CORBA Object Details Window”.

Figure 11. CORBA Object Details Window



8. 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`

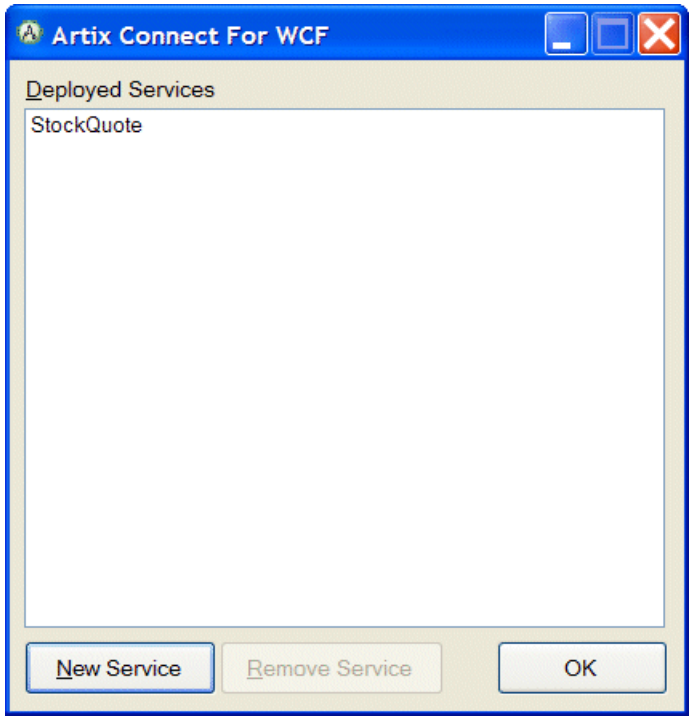
9. Select the `StockQuote.ior` file and click Open.

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

10. Click Finish.

In the Artix Connect for WCF wizard, the CORBA stock quote system is added to the list of deployed services (see Figure 12, “CORBA StockQuote System Added to Deployed Services List”).

Figure 12. CORBA StockQuote System Added to Deployed Services List



Step 5: Using the Wizard to Connect to JMS

Introduction

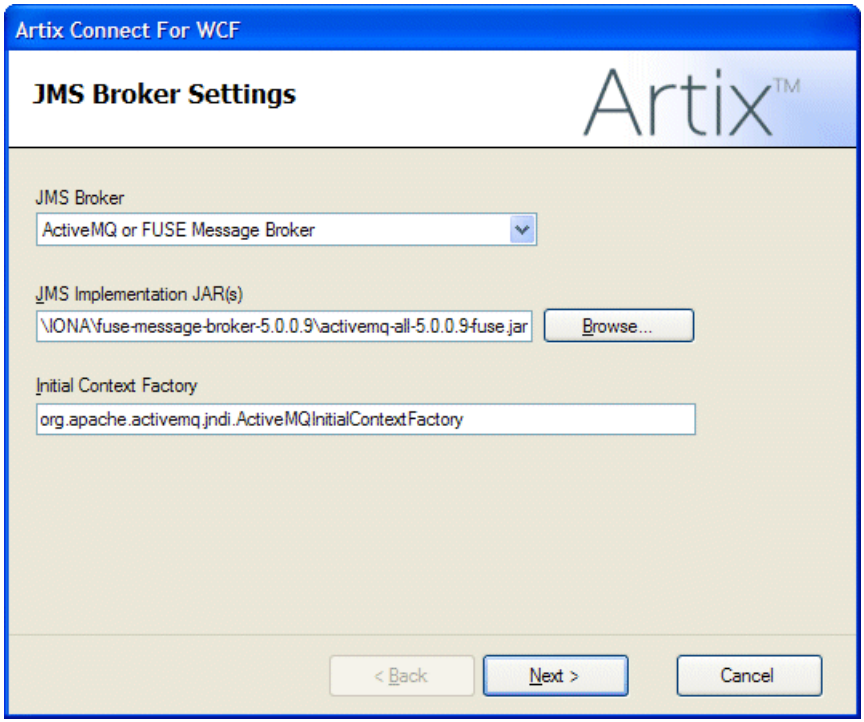
The default JMS broker used in this tutorial is FUSE Message Broker. If you want to use any of the other supported JMS brokers, please refer to Using one of the other JMS brokers. It provides you with prerequisite steps that you need to complete before using another JMS broker with this sample application. In addition, for each JMS broker, it includes the JMS broker and destination settings that you need when working through the steps in this section.

Selecting a JMS Broker

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

1. In the Artix Connect for WCF wizard, click New Service.
2. In the New Service window, select the JMS radio button.
3. Click Next.
4. In the JMS Broker Settings window, shown in Figure 13, “Adding JMS Broker Settings”:

Figure 13. Adding JMS Broker Settings



i. Under JMS Broker, select ActiveMQ or FUSE Message Broker.

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

ii. 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 5.0.0.9, browse to the top level of the product installation directory and select the `activemq-all-5.0.0.9-fuse.jar` file.

   For a complete list of JMS implementation JARs, see JMS Broker Implementation JARs in *Installation Guide*.

iii. Click `Next`.

Note

You are only asked to set JMS broker settings once. The JMS Broker Settings window does not appear when you run the Artix Connect for WCF wizard again. If you want to subsequently change the JMS broker that you are using, please use the Artix Administration tool to enter details of the new broker. For details, see Using one of the other JMS brokers.

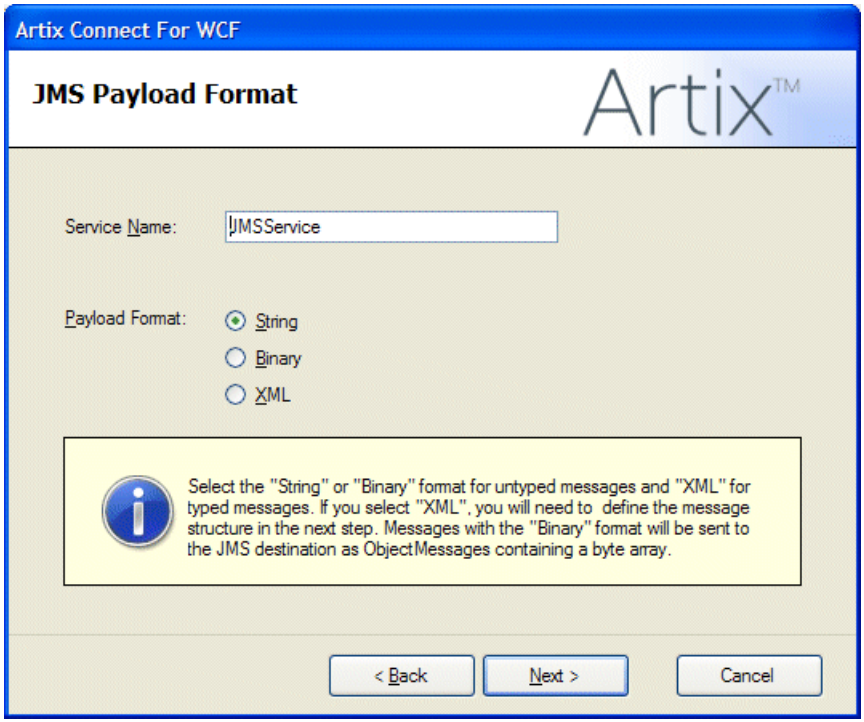
Selecting a payload format

The JMS Payload Format window enables you to give the service 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. Leave the Service Name as `JMSService`
2. Under Payload Format, select the XML radio button, as shown in Figure 14, “Adding JMS Service Name and Payload Format Details”.

Figure 14. Adding JMS Service Name and Payload Format Details



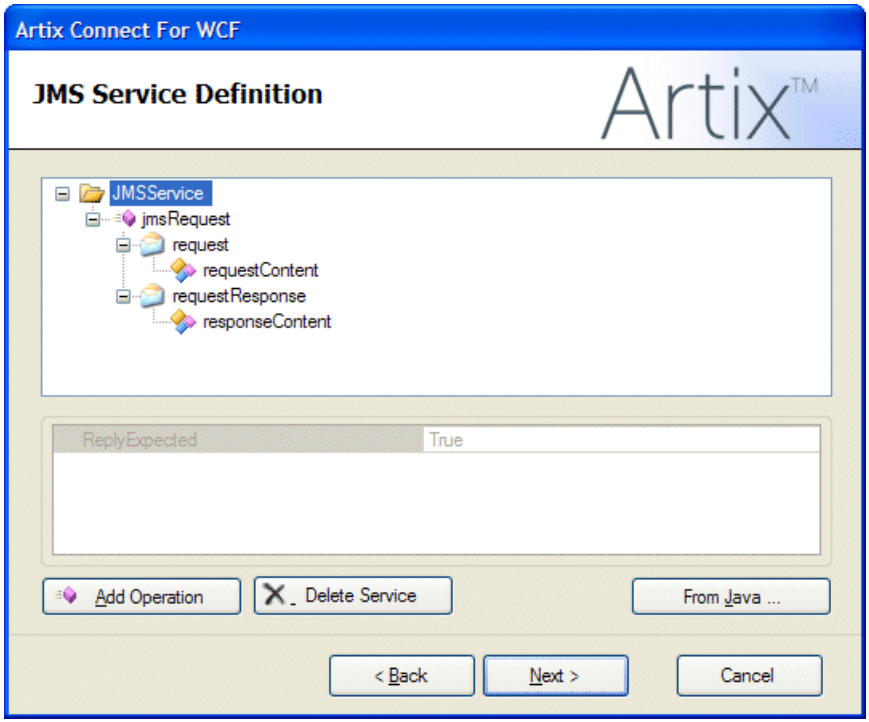
3. Click Next.

4. 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 service panel, the sample application includes a Java class file that represents the interface that you are trying to access.

   In the JMS Service Definition window (shown in Figure 15, “Defining XML Message”):

Figure 15. Defining XML Message



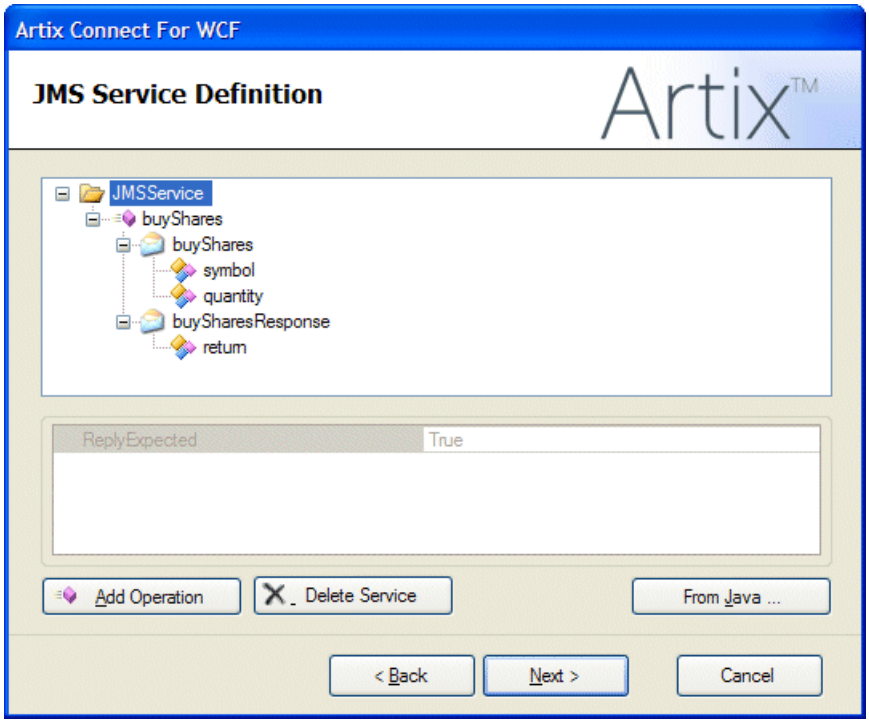
i. Click From Java.

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

iii. 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 Service Definition window, as shown in Figure 16, “XML Message Defined”:

Figure 16. 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.

5. 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, please refer to Using one of the other JMS brokers. It provides JMS destination settings for each of the supported JMS brokers.

To set JMS destination settings for the sample application:

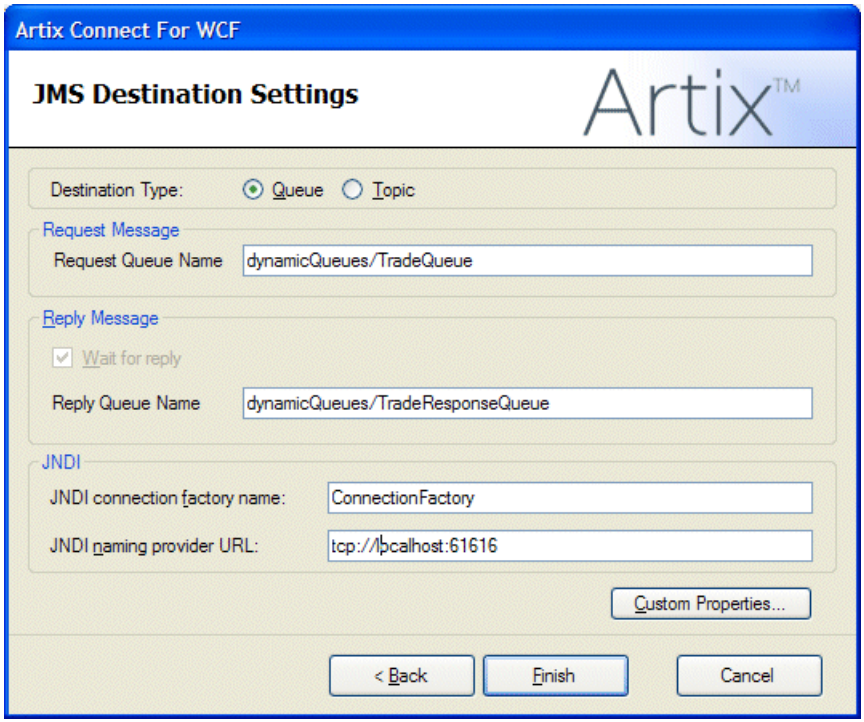
1. Fill in the JMS Destination Settings window with the values shown in Table 5, “JMS Destination Settings for FUSE Message Broker and ActiveMQ”:

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

Setting	Value	Description
Destination Type	`Queue`	Specifies whether you are connecting to a JMS queue or topic.
Request Queue Name	`dynamicQueues/TradeQueue`	Specifies the name of the JMS queue or topic to which you are trying to connect.
Reply Queue Name	`dynamicQueues/TradeResponseQueue`	Specifies the name of the response JMS queue or topic to which you are trying to connect.
JNDI connection factory name	`ConnectionFactory`	Specifies the name of the JMS broker connection factory.
JNDI naming provider URL	`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 17, “JMS Destinations Settings”.

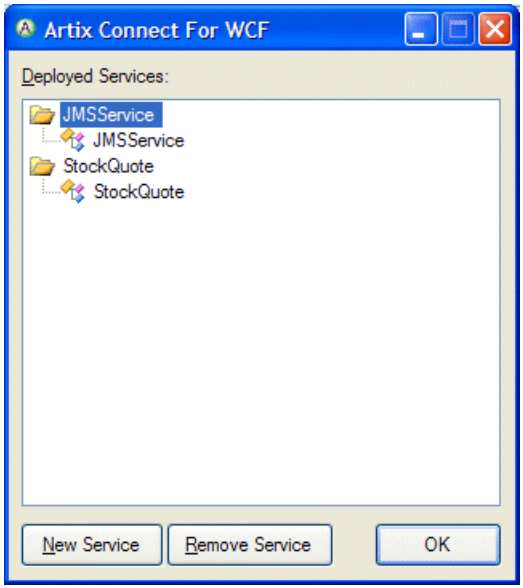
Figure 17. JMS Destinations Settings



2. Click Finish.

   The wizard completes its tasks and returns to the original starting window. Notice, however, that both the CORBA and JMS services are listed under Deployed Services, as shown in Figure 18, “CORBA and JMS Services Successfully Deployed”.

Figure 18. CORBA and JMS Services Successfully Deployed



3. Click OK.

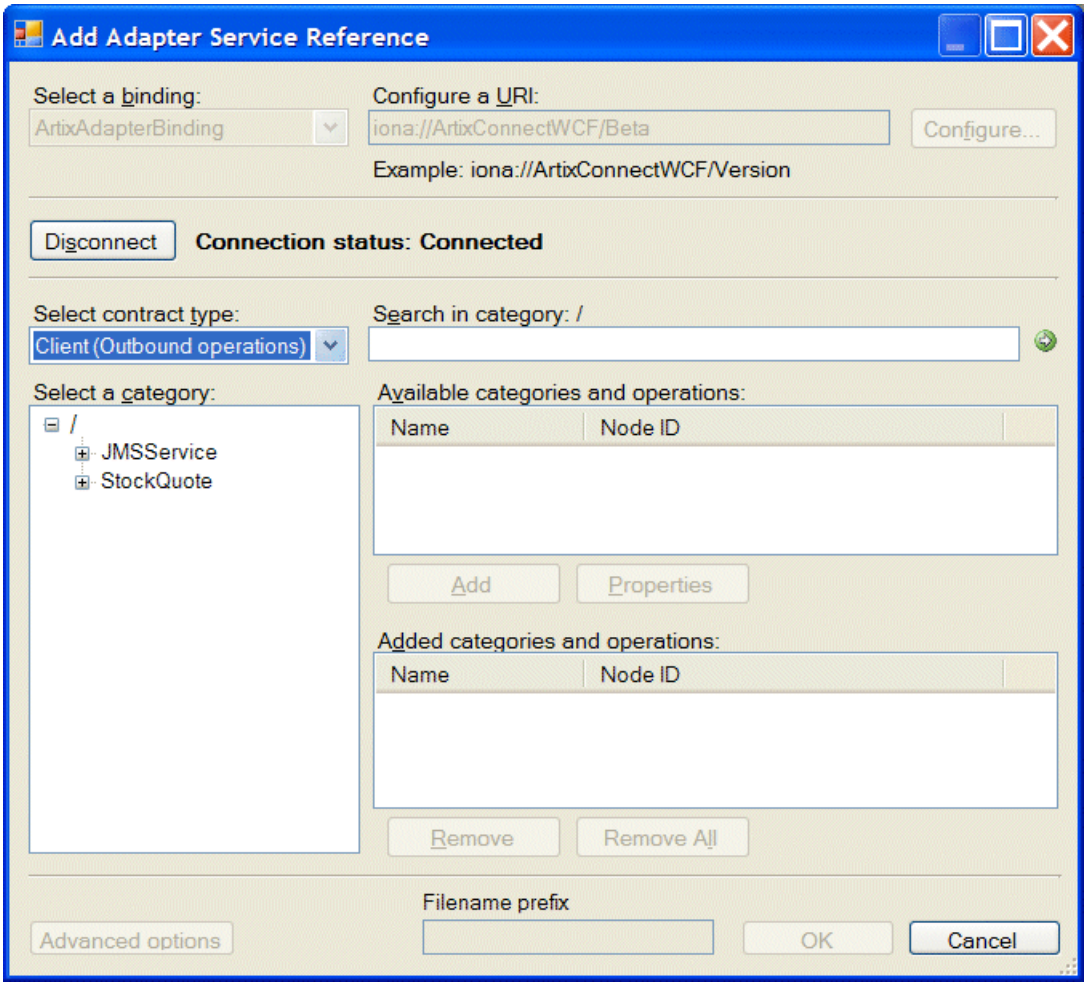
   The wizard completes and returns to the LOB adapter window. It may take a few moments for the LOB adapter window to become responsive again while the JMS and CORBA system details are processed.

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 19, “JMS and CORBA details in the LOB Adapter Window”, with the `JMSService` and `StockQuote` entries listed in the Select a category: panel.

Figure 19. 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 `JMSService` 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, “ServiceCalls.cs after modification”.

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)
            {
                    JMSServiceClient purchaser = new JMSServiceClient();
                    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

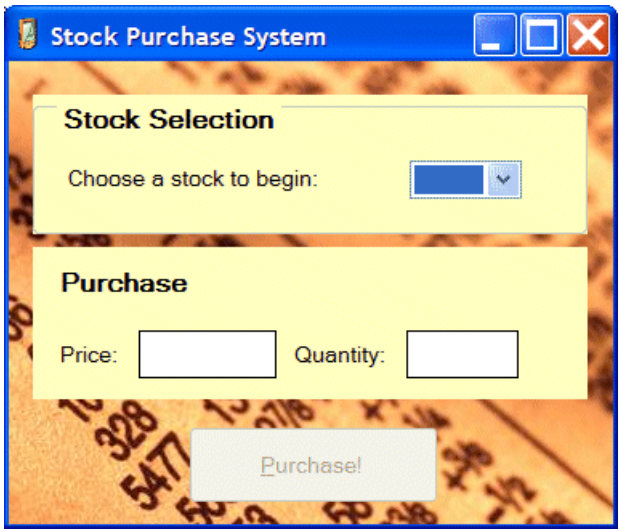
Introduction

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 20, “The Completed WCF Application ”.

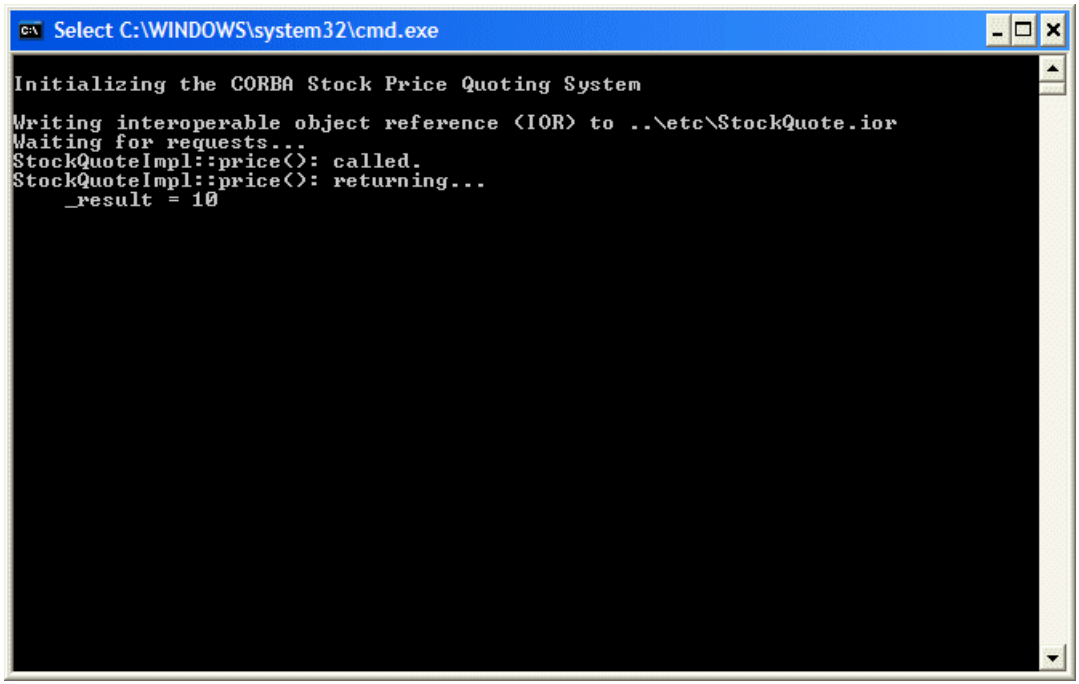
Figure 20. The Completed WCF Application



Try the following, for example:

1. Select IONA from the drop-down list in the Stock Selection panel.
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 21, “CORBA Server Logging an Operation Call”).

Figure 21. CORBA Server Logging an Operation Call



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 22, “Running the Completed Stock Purchase Application”.

Figure 22. Running the Completed Stock Purchase Application

You will see the order being consumed by the Java server, as shown in Figure 23, “Java Server Consuming JMS Request”.

Figure 23. Java Server Consuming JMS Request

Index

Symbols

.NET solution
  opening, 41

A

Apache ActiveMQ
  JNDI connection factory name, 56
  JNDI naming provider URL, 56
Artix Connect for WCF wizard
  connecting to CORBA, 47
  connecting to JMS, 50
  opening, 43

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
  set-up for sample application, 34
  starting JMS broker, 36
  updating sample Java server, 34

C

CORBA
  adding code to call, 62
CORBA operations
  making available to WCF, 59
CORBA sample, 16
CORBA service
  running, 38
CORBA stock quote system, 17

D

Destination Type
  BEA WebLogic, 35
  sample application, 56
  SonicMQ, 29
  TIBCO EMS, 24
  WebSphere MQ, 32

F

FUSE Message Broker
  downloading and installing, 20
  JNDI connection factory name, 56
  JNDI naming provider URL, 56

J

Java server
  running, 39
JMS
  adding code to call, 62
JMS broker
  running FUSE Message Broker, 38
  set-up, 20
JMS destination settings
  BEA WebLogic, 35
  SonicMQ, 29
  TIBCO EMS, 24
  WebSphere MQ, 32
JMS operations
  making available to WCF, 59
JMS sample, 16
JMS stock purchase system
  introduction to, 17
JNDI connection factory name
  Apache ActiveMQ, 56
  BEA WebLogic, 35
  FUSE Message Broker, 56
  SonicMQ, 29
  TIBCO EMS, 24
  WebSphere MQ, 32
JNDI naming provider URL
  Apache ActiveMQ, 56
  BEA WebLogic, 35
  FUSE Message Broker, 56
  SonicMQ, 29
  TIBCO EMS, 24
  WebSphere MQ, 32

R

Reply Queue Name
- BEA WebLogic, 35
- sample application, 56
- SonicMQ, 29
- TIBCO EMS, 24
- WebSphere MQ, 32

Request Queue Name
- BEA WebLogic, 35
- sample application, 56
- SonicMQ, 29
- TIBCO EMS, 24
- WebSphere MQ, 32

S

Sample application
- Destination Type, 56
- how it works, 42
- Reply Queue Name, 56
- Request Queue Name, 56

ServiceCalls.cs, 62

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
- set-up for sample application, 26
- starting JMS broker, 29
- updating sample Java server, 27

stock purchasing application
- running, 64

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
- set-up for sample application, 23
- starting JMS broker, 25
- updating sample Java server, 23

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
- set-up for sample application, 30
- starting JMS broker, 33
- updating sample Java server, 31