PROGRESS SOFTWARE

Artix® Connect for WCF

User's Guide

Version 1.5
October 2008

User's Guide

Progress Software

Version 1.5

Published 15 Jan 2009

Legal Notices

Table of Contents

List of Figures

List of Tables

List of Examples

Preface

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

  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 on page 1

  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:

`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.
`%`	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.

What is Artix Connect for WCF?

This chapter describes, at a high-level, Artix Connect for Windows Communication Foundation (WCF). It includes a brief description of some typical use cases and describes the product architecture.

Use Cases

Product description

Artix Connect for WCF integrates Microsoft's new .NET communication technology, Windows Communications Foundation (WCF), with many diverse middleware and messaging technologies, including CORBA , Java, and EJBs deployed in Java EE application servers (see Figure 1 on page 17). It offers you, as a .NET developer, the ability to communicate with these systems without having to:

- Leave your Visual Studio environment.
- Use unfamiliar techniques.
- Touch back-end server systems.

Supported technologies

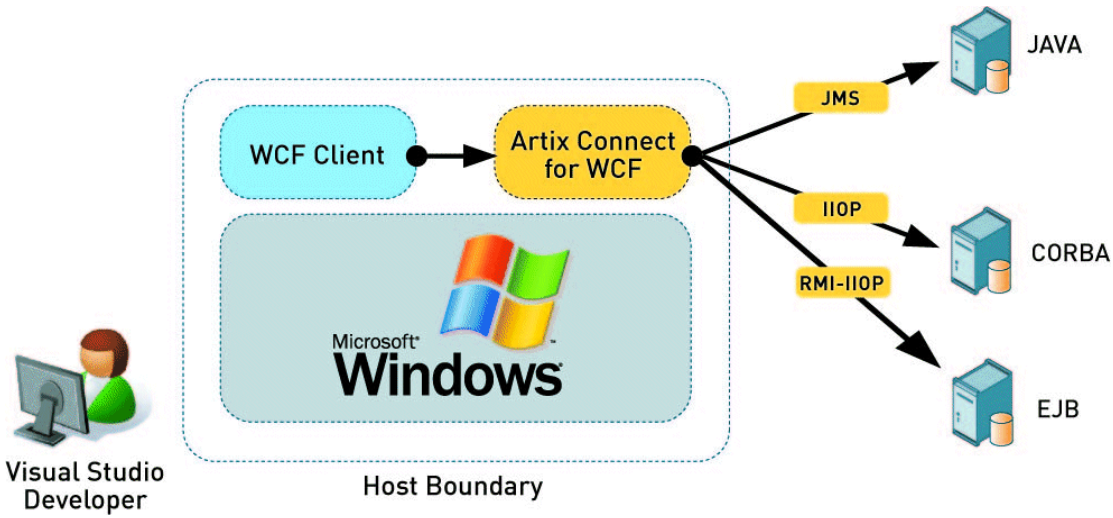
This release enables you to connect to:

- CORBA services: both secure and insecure.
- Java Messaging Services (JMS) queues and topics: both in terms of sending JMS messages and consuming JMS messages.

- Enterprise Java Beans (EJBs): supports session beans deployed in Java EE application servers such as JBoss, WebSphere and WebLogic.

Graphical representation

Figure 1. Artix Connect for WCF Use Cases

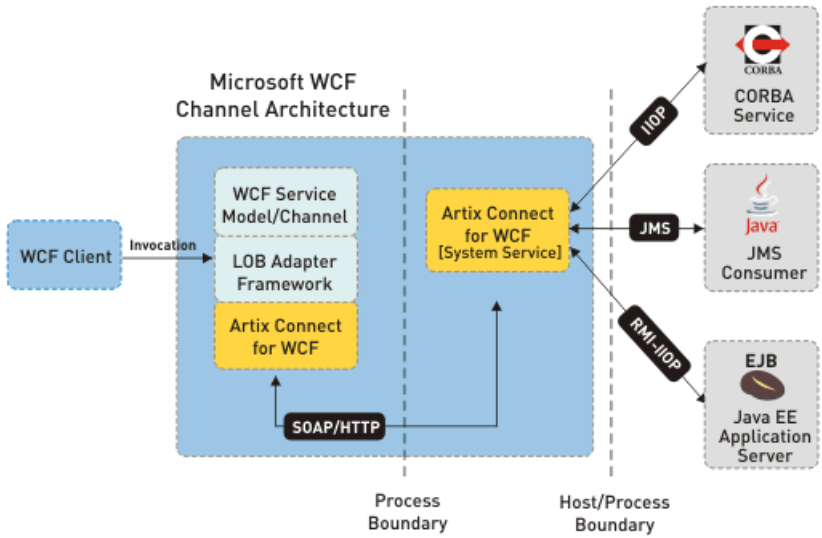


Architecture

Graphical representation

Artix Connect for WCF's architecture is shown in Figure 2 on page 18.

Figure 2. Artix Connect for WCF Architecture



Components

Artix Connect for WCF consists of three main components:

- An *Artix Adapter Framework*, which plugs in to the Microsoft LOB Adapter Framework. The Artix Adapter Framework includes a wizard that enables you to design and configure LOB clients and services for such back-end systems as CORBA, JMS and EJB.

  For more information and instructions on how to use the Artix Adapter Framework and its wizard, see:

  - *Connecting to CORBA Servers* on page 21
  - *Connecting to JMS Queues and Topics* on page 53
  - *Connecting to Enterprise Java Beans* on page 77

- An *Artix Service*, which runs as Windows system service and is responsible for monitoring deployed LOB clients and services. It supports different payload formats and translates messages between endpoints that use different messaging transports. For example, it can consume a SOAP message over HTTP from a .NET client and dispatch it to a Java service that uses the JMS transport.

  You can stop, start and reset the Artix service using the Artix Administration tool. For more information, see *Using the Artix Administration Tool* on page 101.

- An *Artix Administration tool*, which is a graphical tool that can interact with the Artix Service and the Artix Adapter Framework.

  For more information, see *Using the Artix Administration Tool* on page 101.

Running the getting started tutorial

If you have not already done so, run the getting started tutorial described in the Getting Started Guide. It will help you learn how to use Artix Connect for WCF.

Connecting to CORBA Servers

This chapter describes how to use Artix Connect for WCF to connect to CORBA servers, both with or without security.

Introduction to CORBA

What is CORBA?

The Common Object Request Broker Architecture (CORBA) is a standard defined by the Object Management Group (OMG). It enables software components written in multiple computer languages and running on multiple computers to work together.

What is IDL?

In CORBA, an IDL file defines the public application programming interface (API) that is exposed by objects in a server application. A CORBA object type is called an interface, which is similar to the concept in C++ of a class or an interface in Java. You do not need to understand IDL to use Artix Connect for WCF, but you do need to have access to the IDL file that defines the interface to the CORBA object to which you want to connect.

Example 1 on page 22 is taken from the sample application described in the Getting Started Guide. Clients of the object 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);
 };
```

What is an object reference?

In CORBA, an object reference specifies the contact details that a client application uses to communicate with a CORBA object. It is often referred to as an interoperable object reference (IOR). You do not need to understand object references to use Artix Connect for WCF, but you do need to know the object reference of the CORBA object to which you are trying to connect.

Artix Connect for WCF supports the following object reference types:

- *Stringified IOR*—this can be stored in a file or simply copied and pasted directly into the Artix Connect for WCF wizard (see Example 2 on page 23).

Example 2. Example of a CORBA Stringified IOR

```
IOR:010000003200000049444c3a696f6e612e636f6d2f49545f4f54535f536572766963654164696e2f5
365727669636541646d696e3a312e3000000001000000000000008a000000010102000800000066626f
6c74616e00030c00003f0000003a3e0232310f73696d706c652e6c6f636174696f6e11694f5453006f7473
746d0061646d696e00175472616e73616374696f6e536572766963654164696e00020000000010000000
0180000000100000001000100000000000001010001000000090101000600000006000000010000002600
```

- *CORBALoc*—a URL that specifies the location of a CORBA object in a human-readable format with the minimum amount of information necessary (see Example 3 on page 23).

Example 3. Example of a CORBALoc

```
corbaloc::localhost:3075/john
```

- *CORBAName*— a URL, similar to a `CORBALoc`, but specifies how to contact a *CORBA Naming Service*. A CORBA Naming Service associates abstract names with CORBA objects and allows clients to find those objects by looking up the corresponding names. To obtain a reference to an object, a client requests the naming service to look up the object associated with a specified name. In the `CORBAName` URL the naming service is followed by "*#*" and the name of the object within the naming service (see Example 4 on page 23).

Example 4. Example of a CORBAName

```
corbaname::localhost:3075/NameService#staff/john.person
```

Using multiple interfaces

A single IDL file can define multiple interfaces. Artix Connect for WCF supports multiple interfaces and lets you to choose the interfaces that you want to use. For more information, see Using Multiple Interfaces on page 35.

Using a factory pattern

The factory pattern is commonly used when designing CORBA servers. Essentially one object, a factory, provides access to one or more additional

objects. The factory object can represent a focal point for clients. The object reference of the factory object is all that the client needs to gain access to other objects in the system. A simple example is a banking object that is responsible for creating and managing accounts. The banking object could have one operation, `get_account`, that returns references to account objects that handle the more low-level operations for depositing or withdrawing money from an account. In this case, the bank implementation object is a factory for account objects. A factory constructs an object and returns a reference to it based on parameters passed to the factory.

Artix Connect for WCF makes it easy for you to connect to CORBA servers that use such a pattern. For more information, see Using the Factory Pattern on page 37.

Using Security

Artix Connect for WCF enables you to connect to CORBA servers using any one of the following:

- **No security**

  If the CORBA server to which you are trying to connect does not use security, follow the instructions outlined in Connecting to a CORBA Server on page 26.

- **Transport Layer Security / Secure Sockets Layer (TLS/SSL) Mutual Authentication**

  If the CORBA server to which you are trying to connect requires TLS/SSL mutual authentication:

  i. Apply the required security settings by completing the steps outlined in Configuring TLS/SSL Mutual Authentication on page 43; and

  ii. Access the CORBA server by following the instructions outlined in Connecting to a CORBA Server on page 26.

- **User/Password Authentication over SSL (CSIv2)**

  If the CORBA server to which you are trying to connect requires user and password authentication over SSL (CSIv2):

  i. Apply the required security settings by completing the steps outlined in Configuring User Name and Password Authentication over SSL (CSIv2) on page 48; and

ii. Access the CORBA server by following the instructions outlined in Connecting to a CORBA Server on page 26.

Note

When connecting to a CORBA server, you can use only one security mode at a time. If you deploy a secure client, you cannot subsequently deploy an insecure client, and vice versa. All CORBA clients must use the same security mode.

More information

To use Artix Connect for WCF to connect to a CORBA server, you do not need to understand CORBA. If, however, you are interested in learning more about the technology, visit the OMG site at:

www.omg.org [http://www.omg.org/]

Connecting to a CORBA Server

Before you Begin

Before you begin connecting to a CORBA server you must have:

1. Access to the CORBA server IDL file.
2. Access to the CORBA object reference. Artix Connect for WCF supports the following object reference formats:

   a. `IOR` in the form of a string or a file

   b. `CORBALoc`

   c. `CORBAName`

If you do not have this information, ask your CORBA administrator to provide it.

Step 1: Launching the Artix Connect for WCF wizard

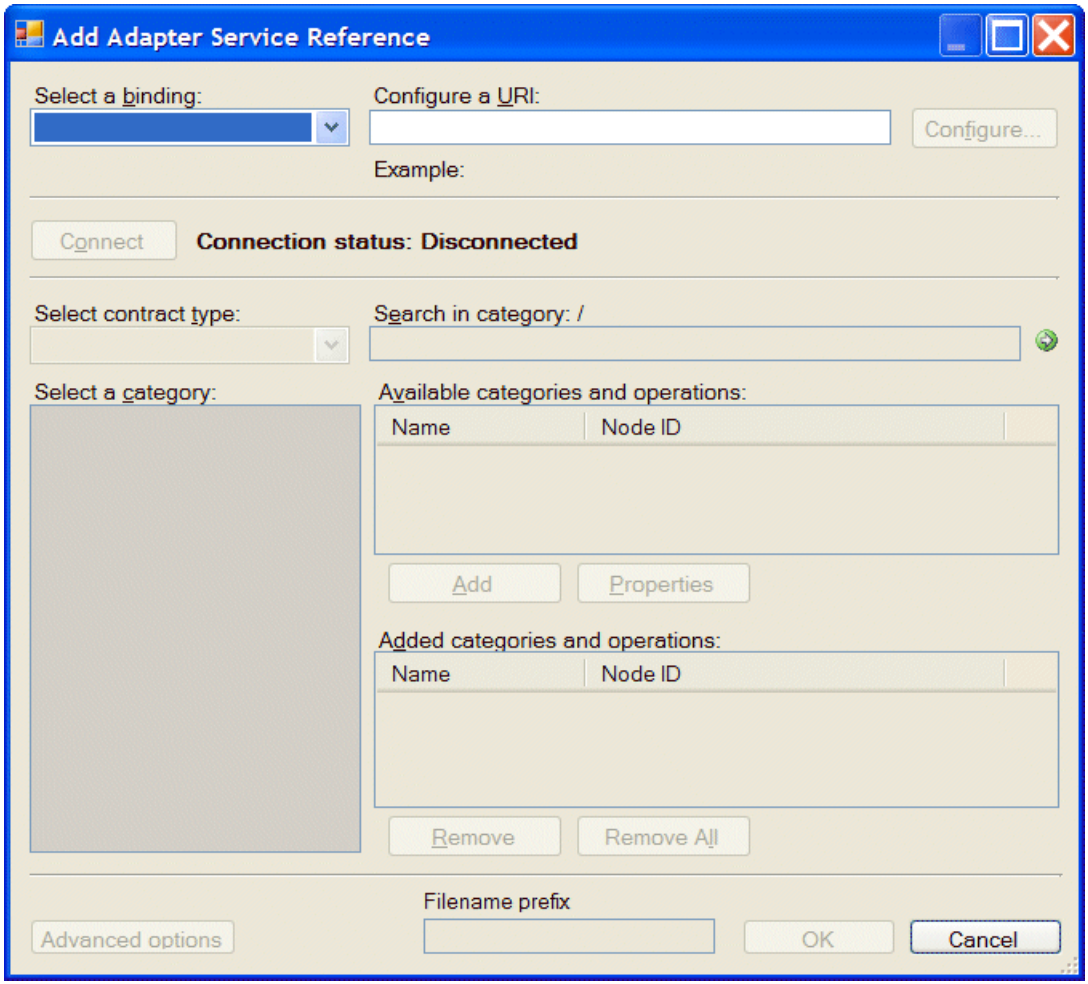
Artix Connect for WCF is a plug-in to the Microsoft LOB adapter framework. The first step in using Artix Connect for WCF is to launch this framework:

1. In the Solution Explorer window, right-click on your project and select **Add Adapter Service Reference** from the context menu.

   This launches the Microsoft LOB Adapter framework.

2. In the Add Adapter Service Reference wizard, shown in Figure 3 on page 27.

Figure 3. Add Adapter Service Reference Wizard



i. In the **Select a binding** field, choose **ArtixAdapterBinding** from the drop-down list of bindings.

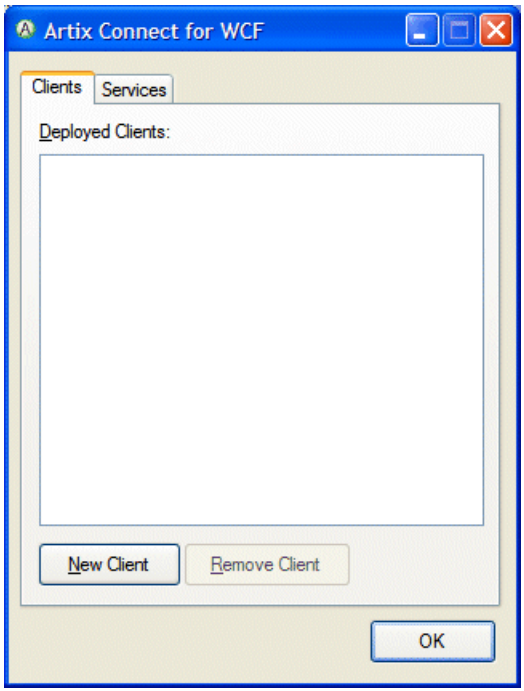
ii. Click **Configure**.

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

iv. In the Add Adapter Service Reference wizard, click **Connect**.

The Artix Connect for WCF wizard opens as shown in Figure 4 on page 28. If you have already deployed clients using Artix Connect for WCF, they will be listed in under Deployed Clients.

Figure 4. Artix Connect for WCF Wizard



Step 2: Adding a CORBA client

To add a CORBA client:

1. In the Artix Connect for WCF wizard, click **New Client**.
2. In the New Client wizard, select the CORBA radio button and click **Next**.
3. The CORBA Security Settings window only appears if you have not already used the Artix Administration tool to choose the security mode that you need.

If the CORBA Security Settings window appears, choose one of the following from the drop-down list:

Table 1. CORBA Security Settings

Setting	Description	Steps
None	The CORBA server to which you want to connect does not require security.	Click **Next** and proceed to step 4 below.
TLS/SSL Mutual Authentication	The CORBA server to which you want to connect is secure and requires you to use TLS/SSL mutual authentication.	Click **Next** and enter the requested security settings (for details see steps 4–7 in Configuring TLS/SSL Mutual Authentication on page 45). Click **Next** and proceed to step 4 below.
User/Password Authentication over SSL (CSIv2)	The CORBA server to which you want to connect is secure and requires you to use user name and password authentication over SSL (CSIv2).	Click **Next** and enter the requested security settings (for details see steps 4–8 in Configuring user name and Password Authentication over SSL (CSIv2) on page 49). Click **Next** and proceed to step 4 below.

Note

When connecting to a CORBA server, you can use only one security mode at a time. If you deploy a secure client, you cannot subsequently deploy an insecure client, and vice versa. All CORBA clients must use the same security mode.

If you want to subsequently change the CORBA security mode, please use the Artix Administration tool to enter the new details (see Configuring TLS/SSL Mutual Authentication on page 43 or Configuring User Name and Password Authentication over SSL (CSIv2) on page 48 for instructions).

4. In the IDL File Selection window, click **Browse** and enter the location of your CORBA server IDL file.

   If the IDL file includes other IDL files:

   i. Click **Add**.

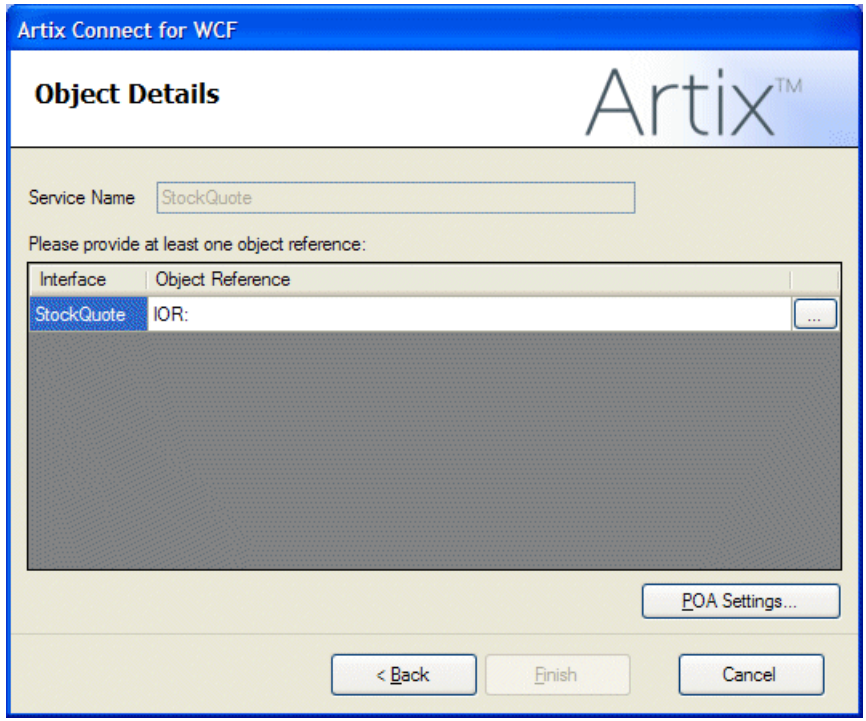
ii. In the Add Search Path window, click **Browse** and enter the location of the included IDL files.

5. Click **Next**.

   The wizard checks that the IDL file is valid.

   The interfaces defined in the IDL file are listed in the Object Details window (see, for example, Figure 5 on page 30).

Figure 5. CORBA Object Details Window



6. Select the interface that you want to use and click **...** to browse to the location of your object reference.

7. *Optional Step:* If the CORBA object that you want to connect to is persistent (that is, the object references are persisted and continue to be valid even

if the server stops and restarts), you can use the POA settings dialog to indicate this to Artix Connect for WCF.

i. Click **POA Settings**.

ii. In the Advanced Setting window, tick the **Persistent** check box.

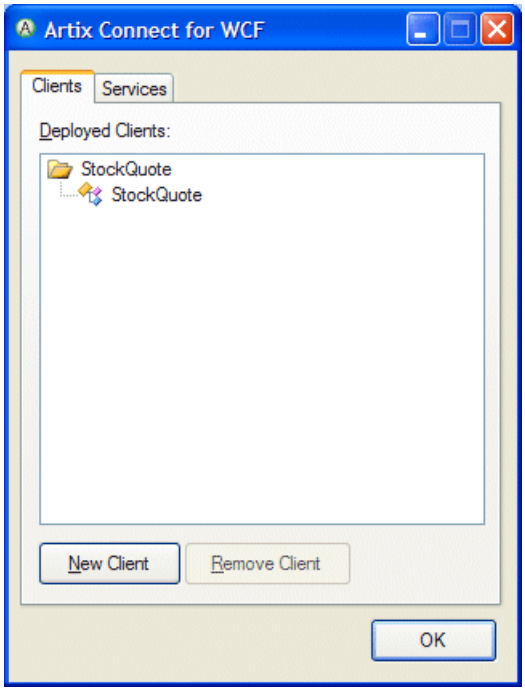
iii. Enter the name of the POA.

iv. Click **OK**.

8. Click **Finish**.

   The CORBA client is added to the list of deployed clients (see, for example, Figure 6 on page 31).

Figure 6. CORBA Client Deployed



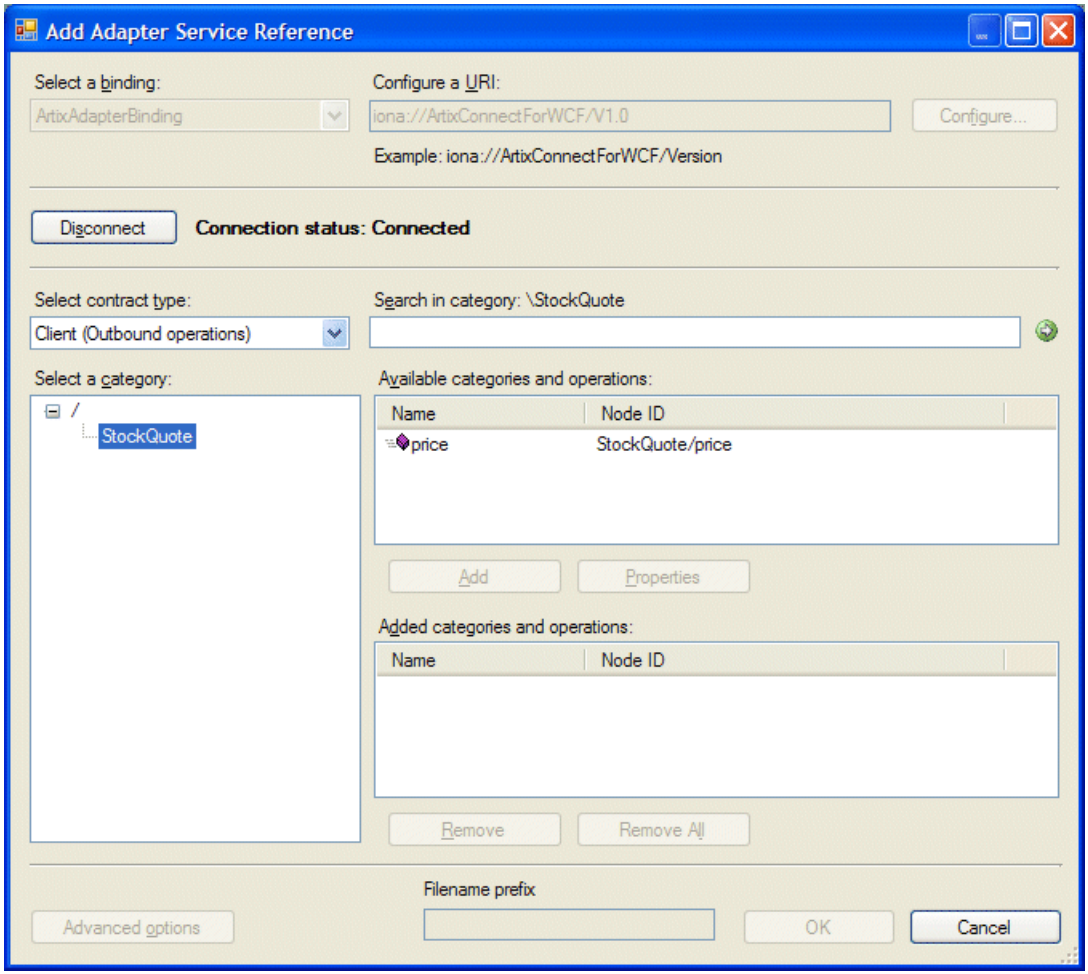
9. Click **OK**.

The wizard completes and returns to the Add Adapter Service Reference wizard. This may take a few moments while the CORBA system details are processed.

Step 3: Making CORBA Operations Available to Your WCF Application

The CORBA client is listed in the **Select a category** panel of the Add Adapter Service Reference wizard (see, for example, Figure 7 on page 33). The **OK** button is disabled. It will remain so until you specify which operations you want to use within your WCF application code.

Figure 7. Making CORBA Operations Available to WCF



To make CORBA operations available to .NET, complete the following steps:

1. In the Add Adapter Service Reference wizard, under the **Select a category** panel, select the CORBA client whose operations you want to use.
2. In the **Available categories and operations panel**, select the operations you want to use.

3. Click **Add** to add the operations to the **Added categories and operations** panel.

4. Click **OK**.

The wizard generates the code and configuration needed to enable your WCF application to use the operations.

Step 4: Adding Code to Call the CORBA Server

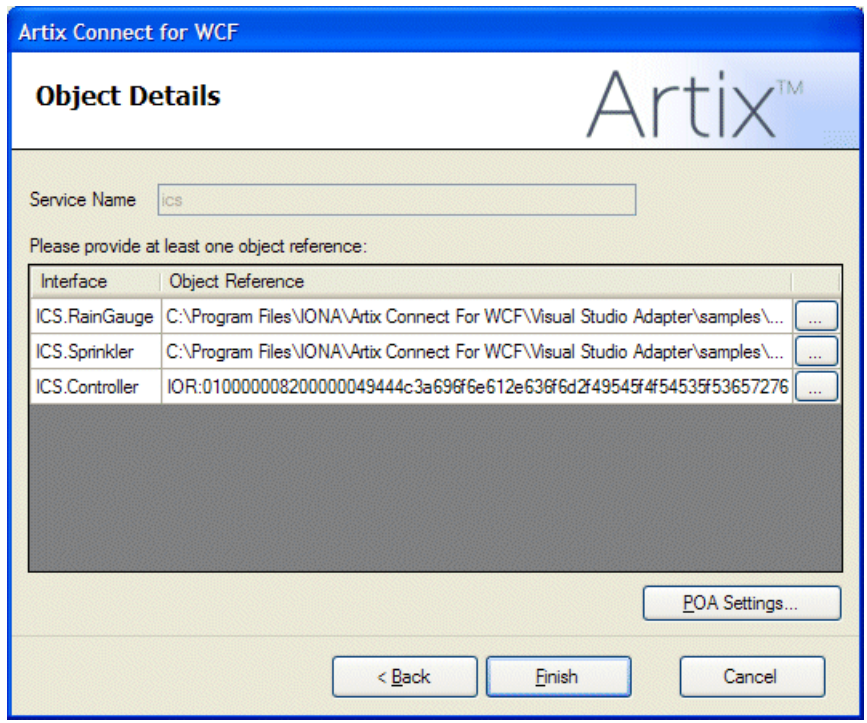
You will notice after clicking the **OK** button that your project has some new files in it, and also that your Visual Studio IntelliSense offers new symbols relating to the CORBA operations that you just added. Your project has been modified to include new code that presents the CORBA server as a native WCF endpoint. Now you can write .NET code to call the CORBA server. To see an example, run the tutorial outlined in the Getting Started Guide.

Using Multiple Interfaces

Overview

If the IDL file for the CORBA server that you are trying to access defines more than one interface, the interfaces are listed in the CORBA Object Details window of the Artix Connect for WCF wizard. For example, Figure 8 on page 35 lists the interfaces defined in an `ics.idl` file.

Figure 8. Using IDL with Multiple Interfaces

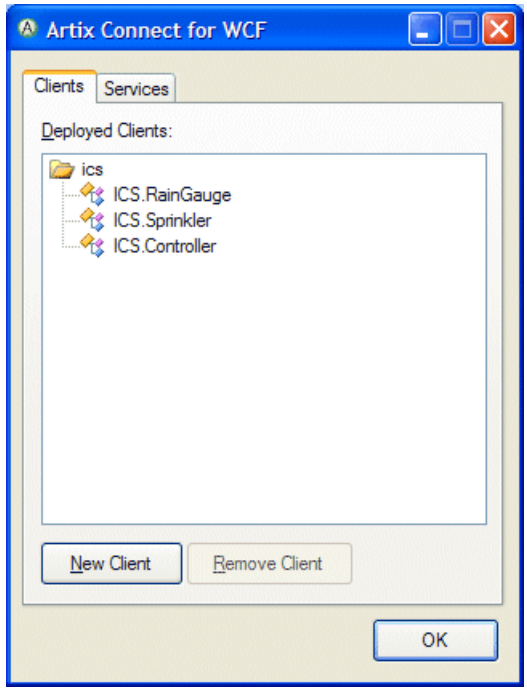


Steps

To use more than one interface, select each interface, click **...** and browse to the location of the object reference for that interface.

When you click **Finish**, the clients are deployed and displayed in a tree that takes the name of the IDL file. For example, Figure 9 on page 36 shows the list of deployed clients as defined in the `ics.idl` file.

Figure 9. CORBA: Multiple Clients Deployed



Note

The factory pattern is a special case in which only the IOR of the factory object should be used. See Using the Factory Pattern on page 37 for more details.

Using the Factory Pattern

Introduction

The factory pattern is a special case in which multiple interfaces are defined in a single IDL file, but one of the interfaces is a factory interface. It provides access to objects defined by the other interfaces. In this case, instead of providing object references for all of the interfaces that you want to use, you only need to provide an object reference for the factory interface.

Sample application

Artix Connect for WCF includes a sample application in which a factory pattern is used. It is located in the following directory of your Artix Connect for WCF installation:

```
InstallDir\Visual Studio Adapter\samples\corba\factory
```

The CORBA server implements a typical factory pattern, where a `Bank` object acts as the factory and is responsible for creating and returning references to `Account` objects. As a result, you only need to provide the CORBA object reference (IOR) for the `Bank` object when deploying the CORBA client using the Artix Connect for WCF wizard.

Running the sample application

To run the sample application:

1. Start the CORBA server by navigating to the *InstallDir*`\Visual Studio Adapter\samples\corba\factory\bin` directory and double-clicking the `start_corba_server.bat` file.
2. Open the .NET solution file by navigating to the *InstallDir*`\Visual Studio Adapter\samples\corba\factory\dotnet` directory and double-clicking on the `BankApplication.sln` file.
3. Run the Artix Connect for WCF wizard as described in Connecting to a CORBA Server on page 26, but:
   i. When you get to the IDL File Selection window, browse for the `bank.idl` file, which is located in the *InstallDir*`\Visual Studio Adapter\samples\corba\factory\etc` directory.

ii. In the Object Details window, only provide an object reference for the `bankServer.Bank` interface by:

a. Selecting the `bankServer.Bank` interface.

b. Clicking **...** and browsing to the `Bank.ior` file, which is located in

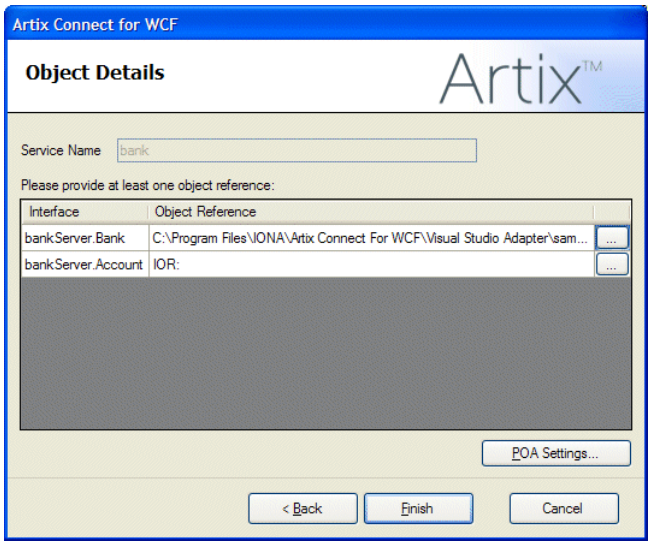
`InstallDir``\Visual Studio Adapter\samples\corba\factory\etc`

Important

Only provide an object reference for the factory interface. Do not provide object references for the other interfaces.

The resulting Object Details window should appear as shown in Figure 10 on page 38:

Figure 10. Selecting the Bank Factory IOR



c. The `Bank` object is persistent. That means it outlives the process in which it is created and a client can contact the CORBA server even

if it stopped and restarted. For the C# client to treat the `Bank` object as persistent, you must:

i. Click **POA Settings**.

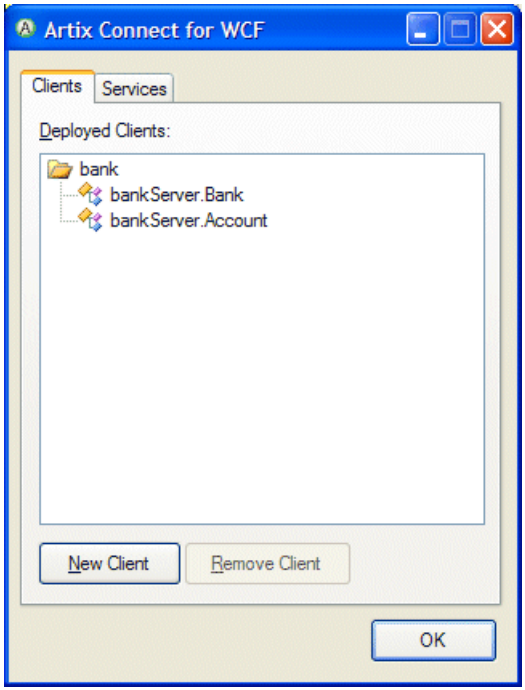
ii. In the Advanced Setting window, tick the **Persistent** check box.

iii. Click **OK**.

d. Click **Finish**.

The `Bank` and `Account` clients appear in the list of deployed clients, as shown in Figure 11 on page 39.

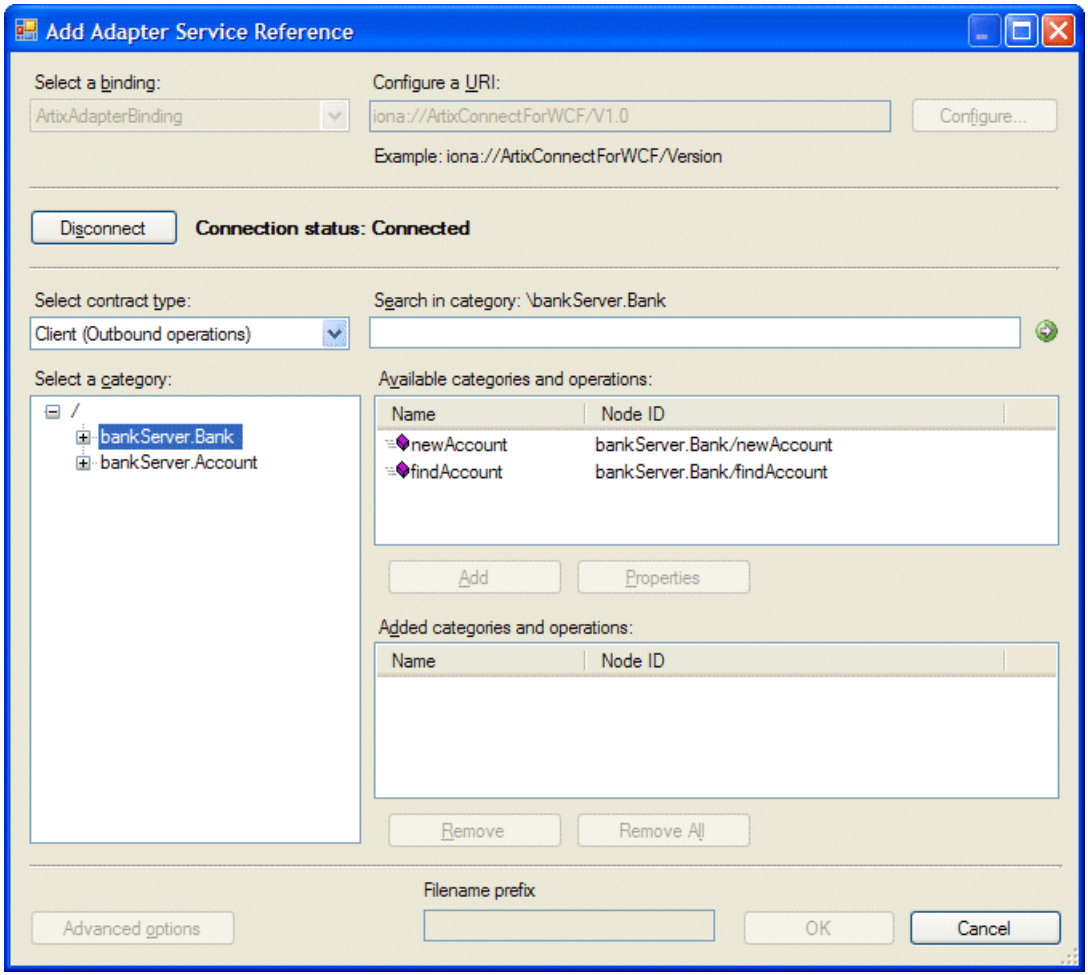
Figure 11. Factory Pattern: Deployed Clients



e. Click **OK**.

iii. The wizard returns to the Add Adapter Service Reference wizard, where the CORBA client is listed in the **Select a category** panel (see Figure 12 on page 40).

Figure 12. Add Adapter Service Reference Wizard: CORBA clients



Choose all of the CORBA operations for use in your WCF application as follows:

i. In the Add Adapter Service Reference wizard, under the **Select a category** panel, select the `bankServer.Bank` interface.

ii. In the **Available categories and operations panel**, select all of the operations.

iii. Click **Add** to add the operations to the **Added categories and operations** panel.

iv. Repeat steps (i) to (iii) for the `bankServer.Account` interface.

v. Click **OK**.

The wizard generates the code and configuration needed to enable the WCF application to use the operations. You can now build and run the client application and watch the account balance grow.

Factory Pattern—.NET Client

The code in the `Program.cs` file illustrates how you would write a .NET client to connect to a CORBA server that makes use of the factory pattern. The basic steps are:

1. Create a factory object.
2. Use the factory object to get a reference to the object that you want.
3. Connect to the object that you want using the reference that you get back from the factory object.

Example 5 on page 41 shows the relevant code from the `Program.cs` file.

Example 5. .NET Client of CORBA Server that Uses a Factory Pattern

```
using System;
using System.Collections.Generic;
using System.Text;
using System.ServiceModel;

namespace BankApplication
{
    class Program
    {
        static void Main(string[] args)
        {
            string name = "ArtixWcf";
```

```
            EndpointReferenceType account_ref;
❶           bankServerBankClient bank = new bankServerBankClient();

             try
            {
❷                account_ref = bank.newAccount(name);
            }
...

         string remote_address = account_ref.Address.Value;
            try
            {
❸                WSHttpBinding binding = new WSHttpBinding(SecurityMode.None);

                 bankServerAccountClient account = new bankServerAccountClient(
                                            binding,
                                            new EndpointAddress(remote_address)
                );
❹                  float current_balance = account._get_balance();
...
                }
         }
     }
}
```

The code shown in Example 5 on page 41 can be explained as follows:

❶ Creates a factory object; in this case, a bank object.

❷ Obtains a reference to the account object.

❸ Creates an account object using the .NET native WSHttp binding and the reference that comes back from the bank factory object.

❹ Invokes on the account object's `get_balance` operation.

Configuring TLS/SSL Mutual Authentication

Introduction

Transport Level Security (TLS) and its predecessor, Secure Sockets Layer (SSL), are cryptographic protocols that provide security and data integrity for network communications. Mutual authentication requires the server to authenticate itself to the client and the client to authenticate itself to the server as a prerequisite to communicating.

X.509 certificates

TLS authentication uses X.509 certificates. The role of the certificate is to associate a public key with the identity contained in the X.509 certificate. Authentication of a secure application depends on the integrity of the public key value in the application's certificate. If an impostor replaced the public key with its own public key, it could impersonate the true application and gain access to secure data.

Certification Authorities

To prevent this form of attack, all certificates must be signed by a certification authority (CA). A CA is a trusted node that confirms the integrity of the public key value in a certificate.

Digital signatures

A CA signs a certificate by adding its digital signature to the certificate. A digital signature is a message encoded with the CA's private key. The CA's public key is made available to applications by distributing a certificate for the CA. Applications verify that certificates are validly signed by decoding the CA's digital signature with the CA's public key.

Security handshake

The server sends its certificate to the client and the client sends its certificate to the server in what is called the security handshake. Each one decides whether or not to trust the received certificate by checking whether the issuer CA is one of a predefined set of trusted CA certificates. If the received X.509 certificate is validly signed by one of the application's trusted CA certificates, the certificate is deemed trustworthy; otherwise, it is rejected.

Sample application

Artix Connect for WCF includes a sample application that demonstrates how use TLS/SSL mutual authentication when communicating with a secure CORBA server. It is located in the following directory of your Artix Connect for WCF installation:

```
InstallDir\Visual Studio Adapter\samples\security\corba
```

To run the sample application, see the `README.txt` file in this directory.

Warning

The sample application uses demonstration certificates. These should not be used in a real-life deployed system.

Prerequisites

Before configuring TLS/SSL mutual authentication, you must have the following:

- A trusted CA certificate, the format of which must be Privacy Enhanced Mail (PEM) (see, for example, `cert.pem` located in *`InstallDir`*`Visual Studio Adapter\samples\security\certs`).
- A client certificate, the format of which must be PKCS#12 (see, for example, `cert.p12` located in *`InstallDir`*`Visual Studio Adapter\samples\security\certs`).
- A password for the client certificate.

If you do not have these, ask your security administrator for them.

Trusted Root Certification Authority list

In addition, you must ensure that the CA certificate is added to the Trusted Root Certification Authority list on your computer. If this is not already done, complete the following steps:

1. From the Windows Start menu, open the Control Panel as follows:

   Start | Control Panel

2. Double-click the **Internet Options** icon.

   This opens the Internet Properties dialog.

3. Select the **Content** tab.

4. Click **Certificates**.

   This opens the Certificates dialog.

5. Select the **Trusted Root Certification Authorities** tab.

6. Click **Import**.

   This opens the Certificate Import Wizard.

7. Click **Next**.
8. Click **Browse**.
9. In the Browse dialog, select All Files (*.*) from the Files of Type drop-down menu.
10. Browse to and select the CA certificate.
11. Click Next.
12. Ensure that **Place all certificates in the following store** is selected and click **Next**.
13. Click **Finish**.
14. You are prompted are you sure you want to install the certificate. Click **Yes**.
15. Close the Certificates dialog.
16. In the Internet Properties dialog, click **OK**.

Configuring TLS/SSL Mutual Authentication

To set TLS/SSL Mutual Authentication, complete the following steps:

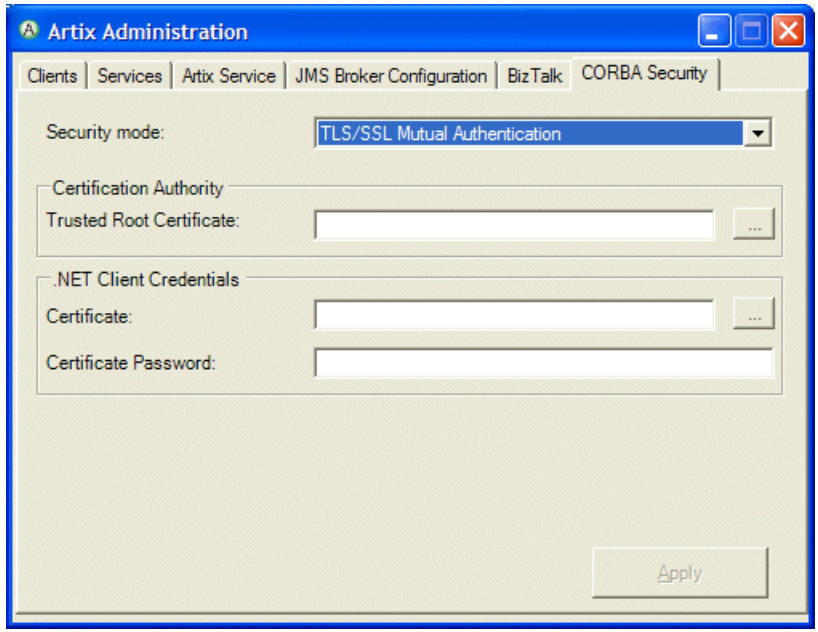
1. Launch the Artix Administration tool from the Windows Start menu:

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

2. Select the **CORBA Security** tab.
3. Select **TLS/SSL Mutual Authentication** from the Security mode drop-down menu.

   The TLS/SSL Mutual Authentication panel of the Artix Administration tool appears as shown in Figure 13 on page 46.

Figure 13. TLS/SSL Mutual Authentication Settings



4. In the Certification Authority panel, select the trusted root certificate by clicking **...**, browsing to and selecting the CA certificate.
5. In the .NET Client Credentials panel, select **...**, browse to and select your client certificate.
6. In the Certificate Password field, type the client certificate password.
7. Click **Apply** to save your security settings.
8. Exit the Artix Administration tool.

Warning

The client certificate password, user name and password are stored in plain text along with the other security settings in the following files in your Artix Connect for WCF installation:

InstallDir\Visual Studio Adapter\artifacts\descriptors\GlobalCredentials.pwf

InstallDir\Visual Studio Adapter\artifacts\domains\security.cfg

You must set the file permissions appropriately to ensure that both the confidentiality and the integrity of the password data are protected.

Configuring User Name and Password Authentication over SSL (CSIv2)

Introduction

The Common Secure Interoperability Protocol Version 2 (CSIv2) defines an authorization over transport mechanism that passes user name and password credentials to the server. The server uses the credentials to authenticate the client.

Sample application

Artix Connect for WCF includes a sample application that demonstrates how use CSIv2 when communicating with a secure CORBA server. It is located in the following directory of your Artix Connect for WCF installation:

```
InstallDir\Visual Studio Adapter\samples\security\corba
```

To run the sample application, see the `README.txt` file in this directory.

Warning

The sample application uses demonstration certificates. These should not be used in a real-life deployed system.

Prerequisites

Before configuring user name and password authentication over SSL (CSIv2), you must have the following:

- A trusted CA certificate, the format of which must be Privacy Enhanced Mail (PEM) (see, for example, `cert.pem` located in *InstallDir*`Visual Studio Adapter\samples\security\certs`).
- A client certificate, the format of which must be PKCS#12 (see, for example, `cert.p12` located in *InstallDir*`Visual Studio Adapter\samples\security\certs`).

  This is required to secure communications between the client and the Artix service.
- A password for the client certificate.

  This is required to secure communications between the client and the Artix service.

- A valid client user name.
- A valid client password.

If you do not have these, ask your security administrator for them.

In addition, you must ensure that the CA certificate is added to the Trusted Root Certification Authority list on your computer. If this is not already done, see Trusted Root Certification Authority list on page 44 for more detail.

Configuring user name and Password Authentication over SSL (CSIv2)

To configure CSIv2 settings complete the following steps:

1. Launch the Artix Administration tool from the Windows Start menu:

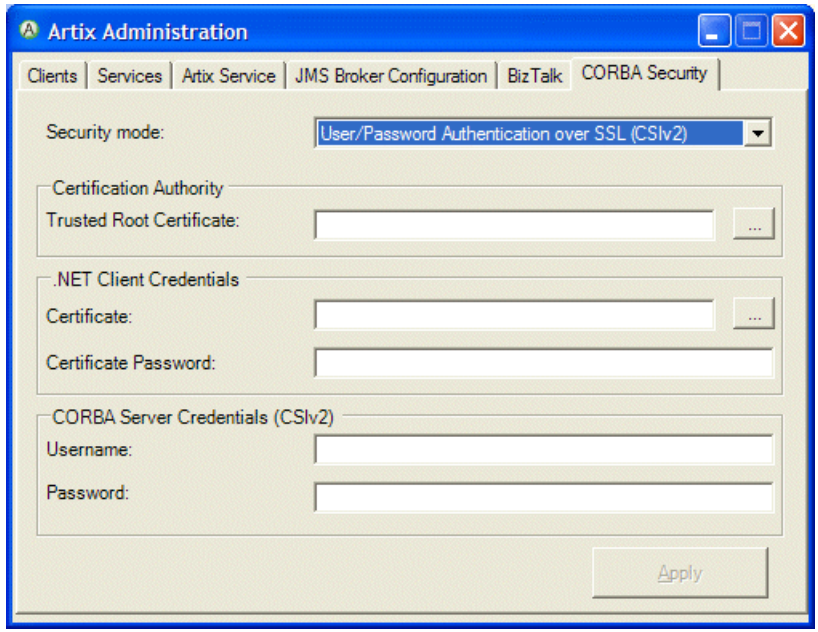
   (All) Programs | IONA | Artix Connect For WCF | Artix Administration

2. Select the **CORBA Security** tab.

3. Select **User/Password Authentication over SSL (CSIv2)** from the Security mode drop-down menu.

   The User/Password Authentication over SSL (CSIv2) panel of the Artix Administration tool appears as shown in Figure 14 on page 50.

Figure 14. User Name and Password Authentication over SSL (CSIv2) Settings



4. In the Certification Authority panel, select the trusted root certificate by clicking **...**, browsing to and selecting the certificate.
5. In the .NET Client Credentials panel, select **...**, browse to and select your client certificate.
6. In the Certificate Password field, type the client certificate password.
7. In the CORBA Server Credentials (CSIv2) panel:
   i. Under username, type the client user name that is required by the CORBA server.
   ii. Under Password, type the client password that is required by the CORBA server.
8. Click **Apply** to save your security settings.

9. Exit the Artix Administration tool.

Warning

The client certificate password, user name and password are stored in plain text along with the other security settings in the following files in your Artix Connect for WCF installation:

InstallDir\Visual Studio Adapter\artifacts\descriptors\GlobalCredentials.pwf

InstallDir\Visual Studio Adapter\artifacts\domains\security.cfg

You must set the file permissions appropriately to ensure that both the confidentiality and the integrity of the password data are protected.

Connecting to JMS Queues and Topics

This chapter describes how to use Artix Connect for WCF to connect to JMS queues and topics.

Introduction

Overview

Artix Connect for WCF enables you to both send JMS messages and consume JMS messages from within the .NET environment. It supports both asynchronous and synchronous RPC-style JMS communications.

What is JMS?

The Java Message Service (JMS) API is a Java Message Oriented Middleware (MOM) API for sending messages between two or more clients. JMS is a part of the Java Platform, Enterprise Edition, and is defined by a specification developed under the Java Community Process.

Queues and topics

A JMS queue is a staging area that contains messages that have been sent and are waiting to be read. The messages are delivered in the order sent. A message is removed from the queue once it has been read.

A JMS topic is a distribution mechanism for publishing messages that are delivered to multiple subscribers.

JNDI

Java Naming and Directory Interface (JNDI) is a set of APIs that assist Java applications to interface with multiple naming and directory services. Artix Connect for WCF uses JNDI to locate and connect to JMS queues and topics.

More information

To use Artix Connect for WCF to send or consume messages from a JMS queue or topic you do not need to understand JMS in detail. If, however, you are interested in learning more about this technology, visit the following website:

http://java.sun.com/products/jms/

Before You Begin

Prerequisites

Before you use Artix Connect for WCF to connect to a JMS queue or topic, you need to know:

- The JMS implementation of the system to which you are trying to connect.
- Connection details for the JMS broker you are using.
- Details of the queue or topic that you are using.

These details are requested by the Artix Connect for WCF wizard. See Figure 17 on page 59 and Figure 20 on page 63 for more detail.

Sending JMS Messages

Sample application

Artix Connect for WCF includes a sample application that demonstrates how to develop a .NET application that sends messages to a JMS queue. It is located in the following directory of your Artix Connect for WCF installation:

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

To run the sample application, see the `README.txt` file in this directory or walk through the steps outlined in this section, which include instructions for, and example settings from, the sample application.

Launching the Artix Connect for WCF wizard

Artix Connect for WCF is a plug-in to the Microsoft LOB adapter framework. The first step in using Artix Connect for WCF is to launch this framework.

1. Open your Visual Studio project.

   Sample: Double-click on the `Messenger.sln` solution file located in:

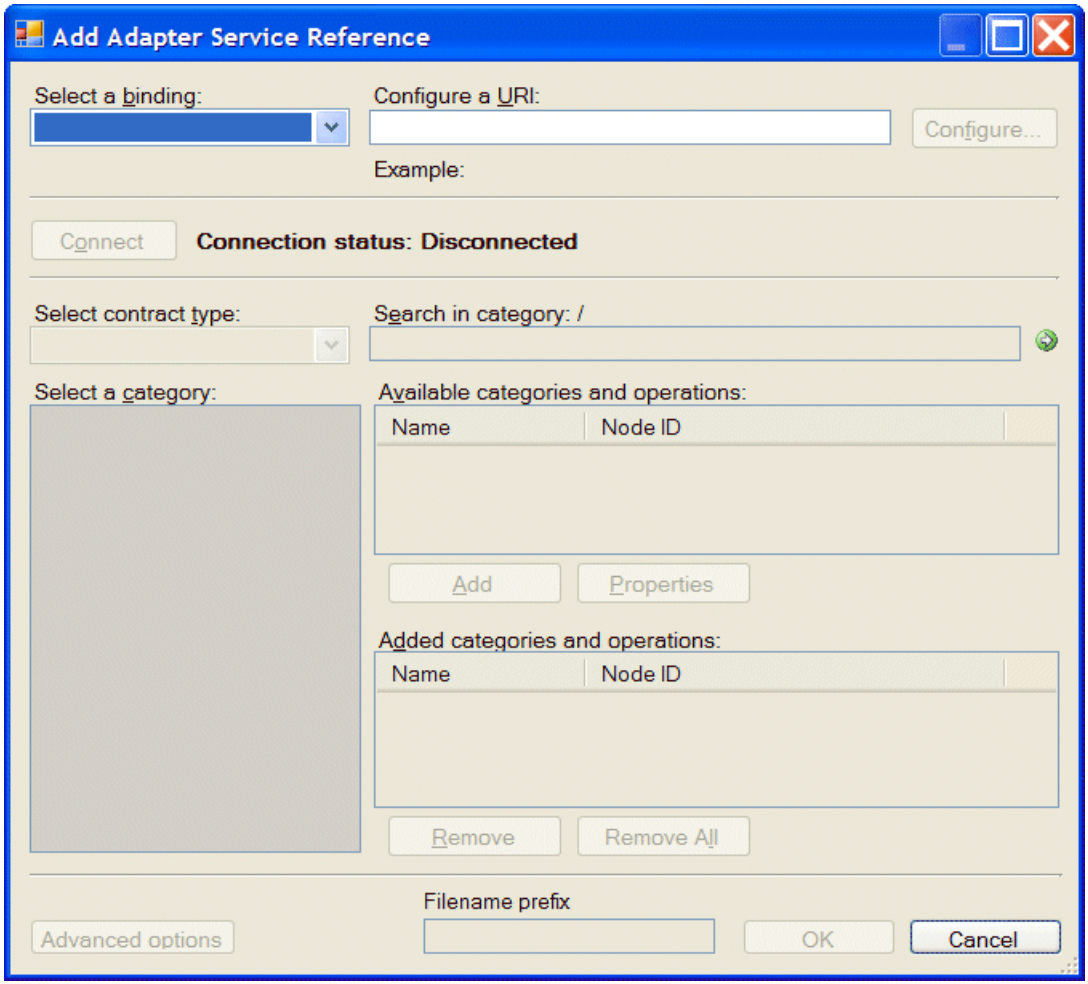
   ```
   InstallDir\Visual Studio Adapter\samples\jms\simple\dotnet
   ```

2. In the Solution Explorer window, right-click on your project and select **Add Adapter Service Reference** from the context menu. This launches the Microsoft LOB Adapter framework.

   Sample: Right-click on `Messenger`.

3. In the Add Adapter Service Reference wizard, shown in Figure 15 on page 57.

Figure 15. Add Adapter Service Reference Wizard



i. In the **Select a binding** field, choose **ArtixAdapterBinding** from the drop-down list of bindings.

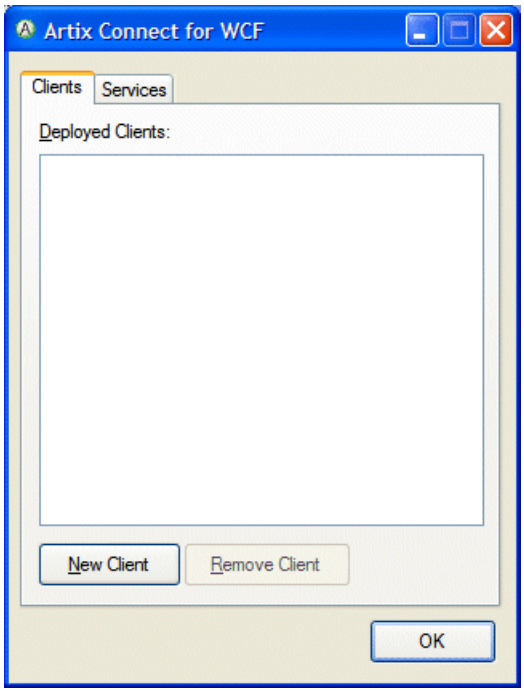
ii. Click **Configure**.

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

iv. In the Add Adapter Service Reference wizard, click **Connect**.

The Artix Connect for WCF wizard opens as shown in Figure 16 on page 58. The deployed clients list is empty if you have not already deployed any clients.

Figure 16. Artix Connect for WCF Wizard

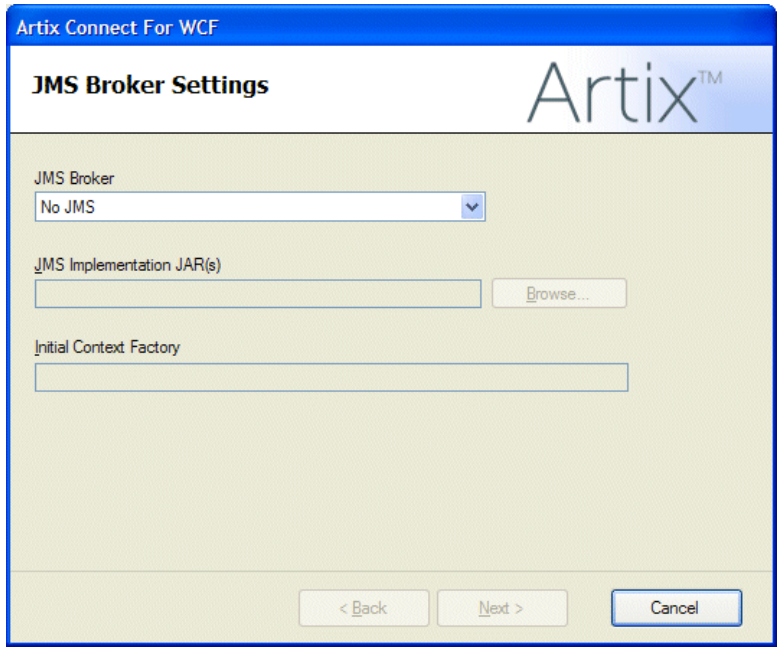


Connecting to JMS

To connect to a JMS queue or topic:

1. In the Artix Connect for WCF wizard, click **New Client**.
2. In the New Client window, select the JMS radio button.
3. Click **Next**.
4. In the JMS Broker Settings window, shown in Figure 17 on page 59:

Figure 17. Adding JMS Broker Settings



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

   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 broker that you selected.

   For a complete list of JMS implementation JARs, see JMS Broker Implementation JARs in the *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

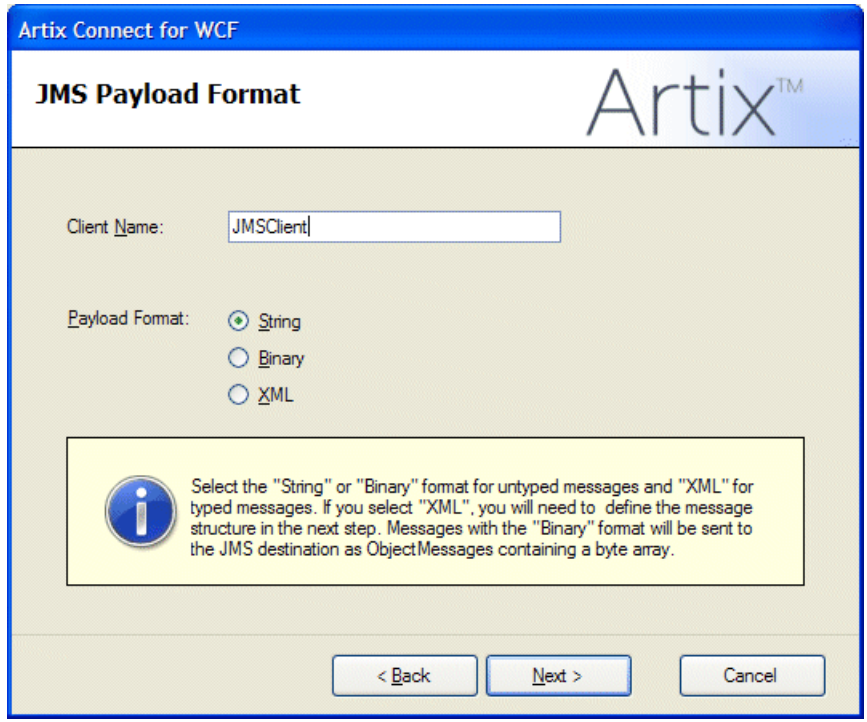
the JMS broker that you are using, please use the Artix Administration tool to enter details of the new broker. For instructions, see Configuring a JMS Broker on page 108.

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. You can send any of the following message types:

- *String*: Untyped messages.
- *Binary*: Untyped messages. Sent to the JMS destination as `ObjectMessages` containing a byte array.
- *XML*: Typed messages. If you select XML, you must define the message structure.

Figure 18. JMS Payload Format



In the JMS Payload Format window:

1. Type the name of your client in the Client Name field.

   Sample: Type `JMSMessenger`.

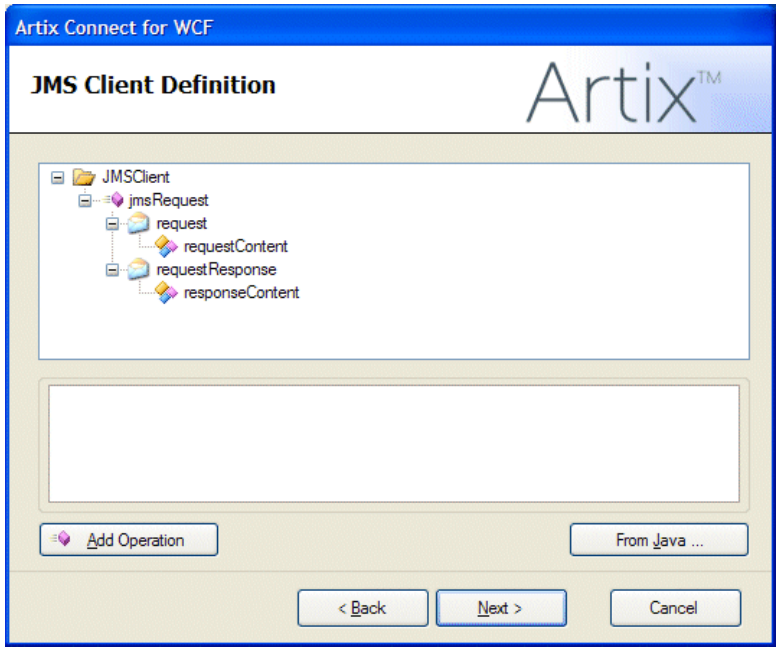
2. Under Payload Format, select which payload you want to use.

   Sample: Select `String`.

3. Click **Next**.

4. If you select XML as the payload format, the JMS Client Definition window appears as shown in Figure 19 on page 62.

Figure 19. Defining XML Message



You can:

- Use the tree and the buttons below the client panel to manually define the XML; or
- Use a Java class file that represents the interface to which you are trying to connect by:

  i. Clicking **From Java**.

  ii. Navigating to the directory that contains the Java class file.

  iii. Selecting the Java class file and clicking **Open**.

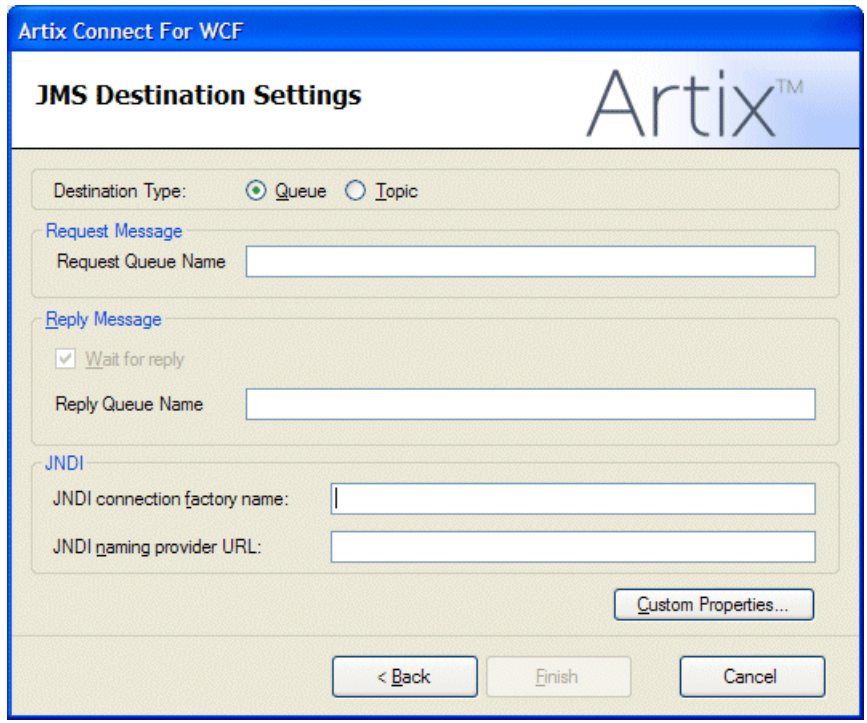
  The wizard examines the Java class and extracts the relevant interface information from it. This information is displayed in the top panel the JMS Client Definition window.

5. Click **Next**.

Specifying JMS destination settings

In the JMS Destination Settings window you need to set JMS destination information (see Figure 20 on page 63. This information is specific to the JMS queue or topic to which you want to send messages and the JMS broker that you are using.

Figure 20. JMS Destination Settings



1. Fill in the JMS Destination Settings window. The fields are described in Table 2 on page 63.

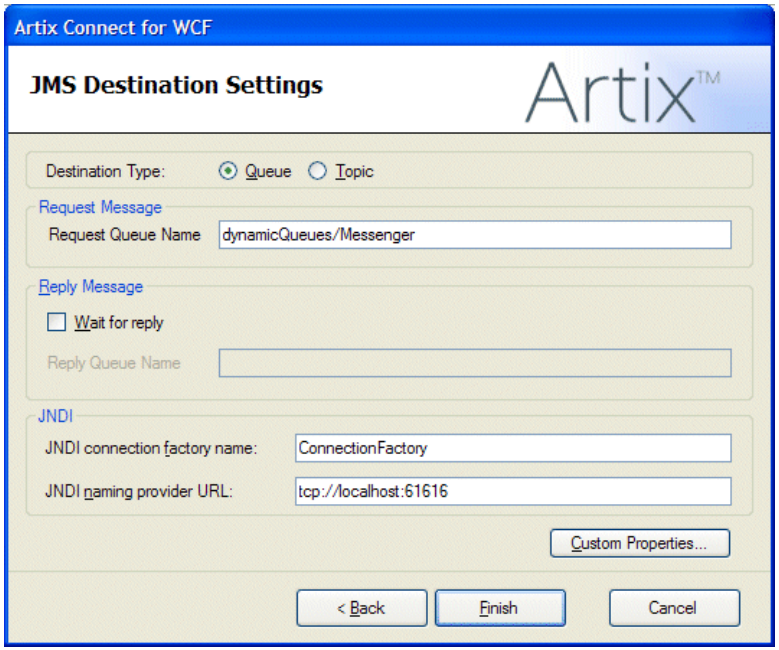
Table 2. JMS Destination Settings

Field	Description
Destination Type	Specifies whether you are connecting to a JMS queue or topic.

Field	Description
Request Queue/Topic Name	Specifies the name of the JMS queue or topic to which you are trying to connect.
Reply Queue/Topic Name	Specifies the name of the JMS queue or topic to which the reply, if there is one, is sent. This is used for synchronous RPC-style communications.
JNDI connection factory name	Specifies the name of the JMS broker connection factory.
JNDI naming provider URL	Specifies the URL used to locate and connect to the JMS broker.

Sample: Settings are shown in Figure 21 on page 64.

Figure 21. JMS Simple Sample: JMS Destination Settings



For example settings for each of the supported JMS brokers, see the Using Other JMS Brokers in the *Getting Started Guide*.

2. If you want to set custom properties; for example, if the JMS client requires an access user name and password:

   i. Click **Custom Properties**.

ii. In the Custom Properties window, under **Name** type the name of your custom property, and under **Value** type the value of your custom property.

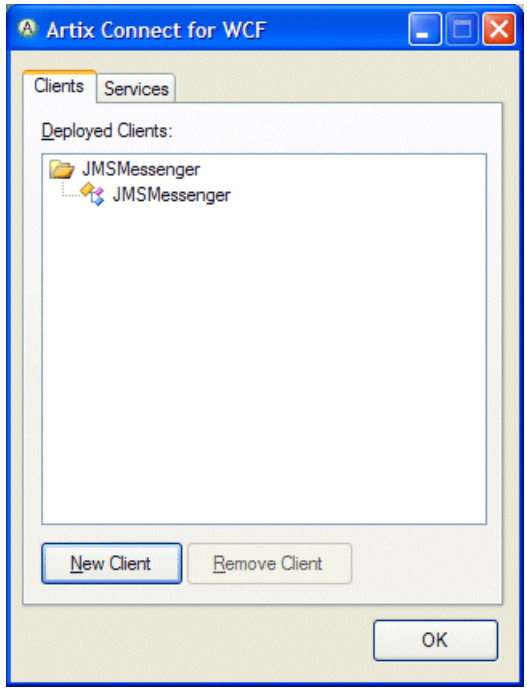
iii. Click **OK**.

Sample: Has no custom settings.

3. Click **Finish**.

   The wizard completes its tasks and the JMS client is listed under Deployed Clients, as shown, for example, in Figure 22 on page 65.

Figure 22. JMS Sample Client Deployed



4. Click **OK**.

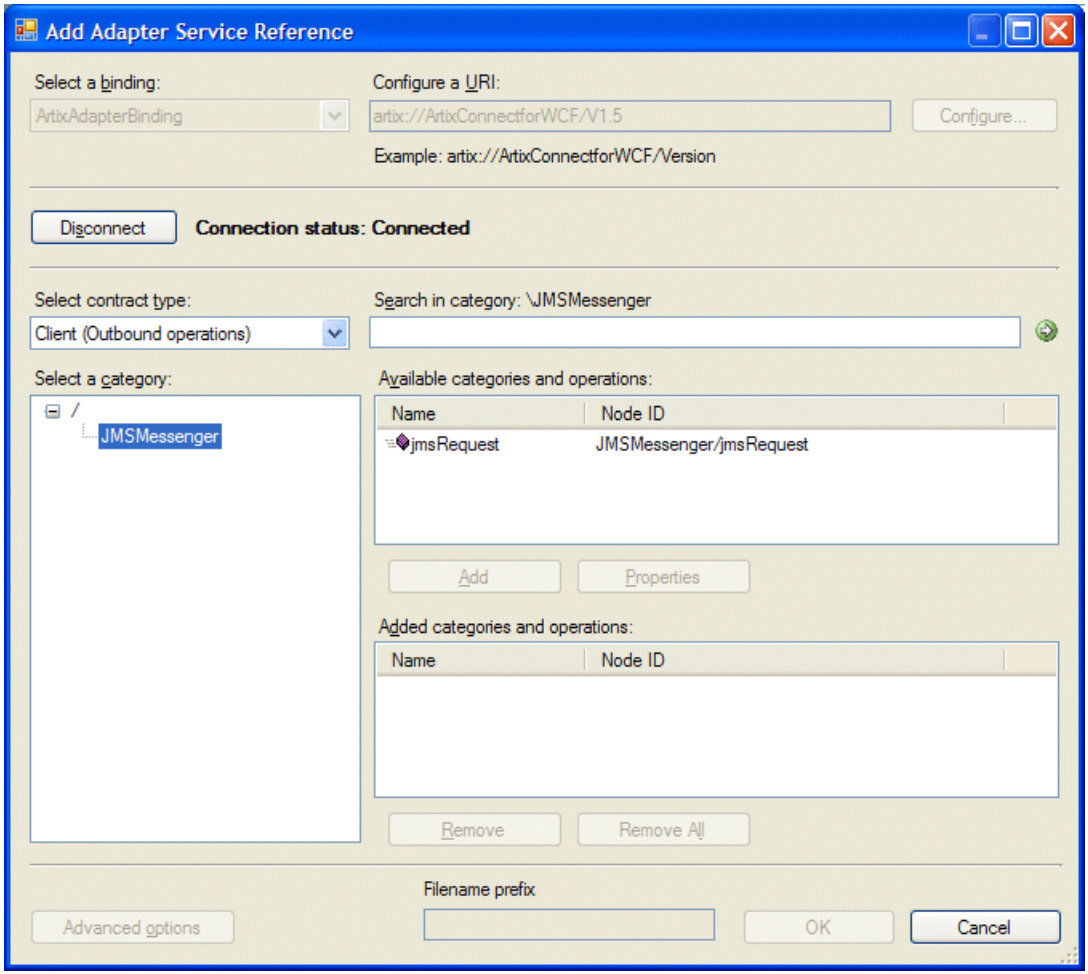
   The Artix Connect for WCF wizard completes and returns to the Add Adapter Service Reference wizard. It may take a few moments for the Add Adapter

Service Reference wizard to become responsive while the JMS system details are processed.

Making JMS operations available to your WCF application

The Add Adapter Service Reference wizard lists the JMS client in the **Select a category** panel (see, for example, Figure 23 on page 67. The **OK** button is disabled. It remains so until you specify which operations you want to use within your WCF application code.

Figure 23. Making JMS Operations Available to .NET Applications: Sample Application



To make JMS operations available to .NET, complete the following steps:

1. In the Add Adapter Service Reference wizard, under the **Select a category** panel, select the JMS client that you just deployed.

   Sample: Select `JMSMessenger`.

2. In the **Available categories and operations** panel, select the operations that you want to use.

   Sample: Select `jmsRequest`.

3. Click **Add** to add the operations 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. Your project is modified to include new code that presents the JMS system as native WCF endpoints. Now you can simply write .NET code to send messages to the JMS system.

Running the sample application

Complete the following steps to run the JMS simple sample application:

1. Open the `Windows.cs` file.

2. Uncomment the two lines of code shown in Example 6 on page 68.

   Example 6. JMS Simple Sample—Uncomment Code

```
JMSMessengerClient client = new JMSMessengerClient();
client.jmsRequest(msgBuffer.Text);
```

3. Build the sample application.

4. Start your JMS broker if it is not already running.

5. Start the Java consumer by running the `start_java_server.bat` in the following folder of the JMS simple sample application:

   InstallDir`\Visual Studio Adapter\samples\jms\simple\bin`

   Once the Java consumer is running, it will wait for JMS requests from the C# client.

6. Run the sample C# client and click **Send** to send the `Hello World!` message.

The message that you send should appear in the Java consumer's console window as follows:

```
Message received:
Hello World!
```

Note

Your system `CLASSPATH` must to include your JMS vendor's implementation JARs or, alternatively, you can run the Java consumer from a command prompt that has the `CLASSPATH` set.

Consuming JMS Messages

Overview

Artix Connect for WCF enables you to develop a .NET application that can consume JMS messages.

Sample application

Artix Connect for WCF includes a sample application that demonstrates how to use Artix Connect for WCF to develop a.NET application that consumes messages from a JMS queue. It is located in the following directory of your Artix Connect for WCF installation:

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

To run the sample application, see the `readme.htm` file in this directory.

Launching the Artix Connect for WCF wizard

Artix Connect for WCF is a plug-in to the Microsoft LOB adapter framework. The first step in using Artix Connect for WCF is to launch this framework.

See Launching the Artix Connect for WCF wizard on page 56 for details. Ignore the sample instructions as they refer to the JMS simple sample application that demonstrated how to use Artix Connect for WCF to develop a .NET application that sends JMS messages.

Connecting to JMS

To use Artix Connect for WCF to connect to JMS:

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 17 on page 59:
   i. Under **JMS Broker**, select the broker that you want to use from the drop-down list.

      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 broker that you selected.

      For a complete list of JMS implementation JARs, see JMS Broker Implementation JARs in the *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 instructions, see Configuring a JMS Broker on page 108.

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 consuming. You can consume any of the following message types:

- *String*: Untyped messages.
- *Binary*: Untyped messages. Sent to the JMS destination as `ObjectMessages` containing a byte array.
- *XML*: Typed messages. If you select XML, you must define the message structure.

In the JMS Payload Format window:

1. Type the name of your service in the Service Name field.
2. Under Payload Format, select which payload you want to use.
3. Click **Next**.
4. If you select XML as the payload format, the JMS Service Definition window appears. You can:
   - Use the tree and the buttons below the client panel to manually define the XML; or
   - Use a Java class file that represents the interface:
     i. Clicking **From Java**.
     ii. Navigating to the directory that contains the Java class file.

iii. Selecting the Java class file and clicking **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.

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 queue or topic from which you want to consume messages and the JMS broker that you are using.

1. Fill in the JMS Destination Settings window. The fields are described in Table 2 on page 63.

   For example settings for each of the supported JMS brokers, see the Using Other JMS Brokers in the *Getting Started Guide*.

2. If you want to set custom properties; for example, if the JMS service requires an access user name and password:

   i. Click **Custom Properties**.

   ii. In the Custom Properties window, under **Name** type the name of your custom property, and under **Value** type the value of your custom property.

   iii. Click **OK**.

3. Click **Finish**.

   The wizard completes its tasks and the JMS service is listed under Deployed Services.

4. Click **OK**.

   The Artix Connect for WCF wizard completes and returns to the Add Adapter Service Reference wizard. It may take a few moments for the Add Adapter Service Reference wizard to become responsive while the JMS system details are processed.

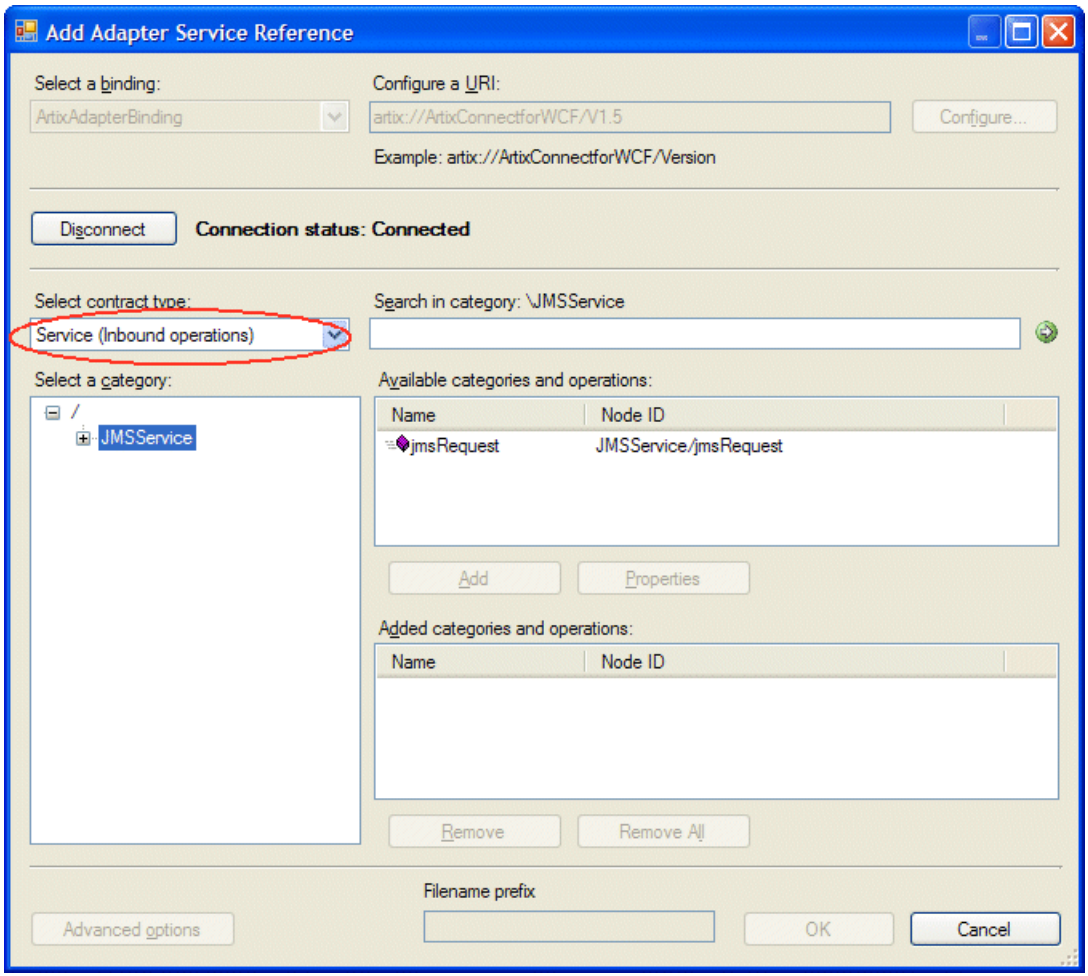
Making JMS operations available to your WCF application

To see the service that you just deployed and to select the operations associated with that service:

1. In the Add Adapter Service Reference wizard, under Select contract type, select Service (inbound operations) from the drop-down list. This makes visible the service that you just deployed.

   See, for example, Figure 24 on page 73

Figure 24. JMS Service in Add Adapter Service Reference Wizard



2. Under the **Select a category** panel, select the JMS service.

3. In the **Available categories and operations** panel, select the operations that you want to use.
4. Click **Add** to add the operations to the **Added categories and operations** panel.
5. Click **OK**.

The wizard starts to generate code and configuration to enable your WCF application to use these operations. Your project is modified to include new code that presents the JMS system as native WCF endpoints.

Implementing a JMS listener

You now need to implement the JMS listener. The stub is contained in the `ArtixAdapterBindingService.cs` file that is generated in the previous step. See, for example, Example 7 on page 74. You must provide an implementation of the `jmsRequest` method.

Example 7. JMS Listener Stub Code

```
namespace ArtixAdapterBindingNamespace {

     public class ArtixAdapterBindingService : JMSService {

          public virtual void jmsRequest(string requestContent) {
               throw new System.NotImplementedException("The method or operation is not
implemented.");
               }
         }
}
```

For an example implementation see the `ArtixAdapterBindingsService.cs` file in the sample JMS listener, which is shown in Example 8 on page 74 and can be found in the following directory of your Artix Connect for WCF installation:

InstallDir`\Visual Studio Adapter\samples\jms\listener\dotnet\WhiteBoard`

Example 8. Implementing a JMS Listener: Sample Code

```
using Samples.WhiteBoard;

namespace ArtixAdapterBindingNamespace {
```

```
    public class ArtixAdapterBindingService : WhiteBoard {

         public virtual void jmsRequest(string updateString)
        {
             WhiteBoardForm.Instance.Update(updateString);
         }
    }
}
```

Connecting to Enterprise Java Beans

This chapter describes how to use the Artix Connect for WCF to connect to Enterprise Java Beans (EJBs).

Introduction

Overview

Artix Connect for WCF enables you to invoke on Enterprise Java Beans (EJBs) from within the .NET environment. It allows you to write standard .NET clients that can communicate with EJBs that have been deployed in any of the following application servers:

- Red Hat JBoss 4.3.2
- IBM WebSphere MQ 6.1
- BEA WebLogic 10

You do not have to edit or change the server-side configuration and you do not have to write any product-specific code.

How it works

Artix Connect for WCF uses industry standard RMI-IIOP to communicate with EJB containers. Using the Artix Connect for WCF wizard, you provide the EJB's JNDI details, including the host and port on which the application server's naming service is running, and name under which the EJB is registered with JNDI. Artix Connect for WCF uses this information when designing the EJB client to create a corbaname URL. At runtime, the underlying Artix CORBA binding accesses the application server's naming service using the corbaname URL to get the EJB Home object reference and EJB Remote object reference. The EJB Remote object reference is used to invoke on the EJB's operations.

All of the required type information for the EJB Home and Remote interfaces is made available in WSDL and the Artix runtime uses this information to marshall and unmarshall the data on the wire.

Restrictions

Artix Connect for WCF has the following restrictions:

1. *CORBA IDL case insensitivity*

   The Java-to-IDL reverse mapping does not support type names that differ only by case. In addition, package names map to an IDL construct (module) that is considered a type. You cannot, therefore, have RMI definitions such as a package named `greeter` that contains an interface named `Greeter`.

   The OMG IDL is case insensitive due to a requirement to support target languages that are case insensitive.

   To avoid this restriction, you should change either the package name or the interface name.

2. *Data type support*

   Artix Connect for WCF supports Java primitive types and strings. Support for arrays, object references and complex types will be made available in the next release.

Deployment Notes

Overview

To use IIOP with JBoss there are a number of deployment tasks of which the person developing and deploying the EJB needs to be aware.

Deploying on JBoss

In the case of JBoss, ensure that:

1. The EJB developer overrides the default invoker, JRMP, by configuring the use IIOP. For instance, Example 9 on page 80, shows a standard stateless session bean configuration file in which the default invoker is replaced by IIOP (see the `<invoker-bindings>` element).

Example 9. jboss.xml Deployment Descriptor

```
<?xml version="1.0" encoding="UTF-8"?>

<jboss>
     <enterprise-beans>
         <session>
             <ejb-name>GreeterBean</ejb-name>
             <jndi-name>ejb/GreeterBean</jndi-name>
             <configuration-name>Standard Stateless SessionBean</configuration-name>
             <invoker-bindings>
                 <invoker>
                     <invoker-proxy-binding-name>iiop</invoker-proxy-binding-name>
                 </invoker>
             </invoker-bindings>
         </session>
    </enterprise-beans>
</jboss>
```

2. The EJB JAR file **must** be deployed in the following JBoss deployment directory:

   JBossInstallDir\server**all**\deploy

3. The JBoss server **must** be started using the `-c all` switch:

   ```
   run -c all
   ```

The `-c all` switch instructs the run script to use the `all` configuration, which includes the IIOP module.

Connecting to an EJB

Before your begin

Before you begin connecting to an EJB you must have the following information:

1. The name of the host on which the application server's naming service is running.
2. The port number on which the application server's naming service is listening.

   The default naming service listener port numbers used by the supported application servers are:

   a. *Red Hat JBoss:* 3528

   b. *IBM WebSphere:* 2809

   c. *BEA WebLogic:* 7001

   The Artix Connect for WCF wizard displays these port numbers by default. If your application server uses a different port number, you need to enter it when you run the wizard.
3. *JBoss* and *WebLogic:* Artix Connect for WCF ùses the EJB's JNDI name as specified in the configuration file (`jboss.xml` and `weblogic-ejb-jar.xml` respectively).

   WebSphere: Artix Connect for WCF requires the full path for the EJB's JNDI name, including the qualified path and prefix. The Artix Connect for WCF wizard displays the generic format of the full path of the JNDI name. You simply need to replace the value of the `node-name`, `server-name`, and `ejb-name`.

Note

To find the `node-name` and the `server-name` you or the EJB deployer should run the WebSphere `dumpNameSpace.bat` utility, which is located in the following directory on your WebSphere machine:

WebSphereInstallDir\bin

ejb-name is specified in the EJB's configuration file, ejb-jar.xml.

Sample application

Artix Connect for WCF includes a sample application that demonstrates how to use Artix Connect for WCF to connect to an EJB. It is located in the following directory of your Artix Connect for WCF installation:

InstallDir\Visual Studio Adapter\samples\ejb

To run the sample application, see the README.txt file in this directory.

The examples shown in this chapter are taken from the sample application and the application server used is JBoss.

Launching the Artix Connect for WCF wizard

Artix Connect for WCF is a plug-in to the Microsoft LOB adapter framework. The first step in using Artix Connect for WCF is to launch this framework.

1. Open your Visual Studio project.

   Sample: Double-click on the GreeterEJBClient.sln solution file located in:

   InstallDir\Visual Studio Adapter\samples\ejb\dotnet

2. In the Solution Explorer window, right-click on your project and select **Add Adapter Service Reference** from the context menu. This launches the Microsoft LOB Adapter framework.

   Sample: Right-click on GreeterEJBClient.

3. In the Add Adapter Service Reference wizard, shown in Figure 3 on page 27.

   i. In the **Select a binding** field, choose **ArtixAdapterBinding** from the drop-down list of bindings.

   ii. Click **Configure**.

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

iv. In the Add Adapter Service Reference wizard, click **Connect**.

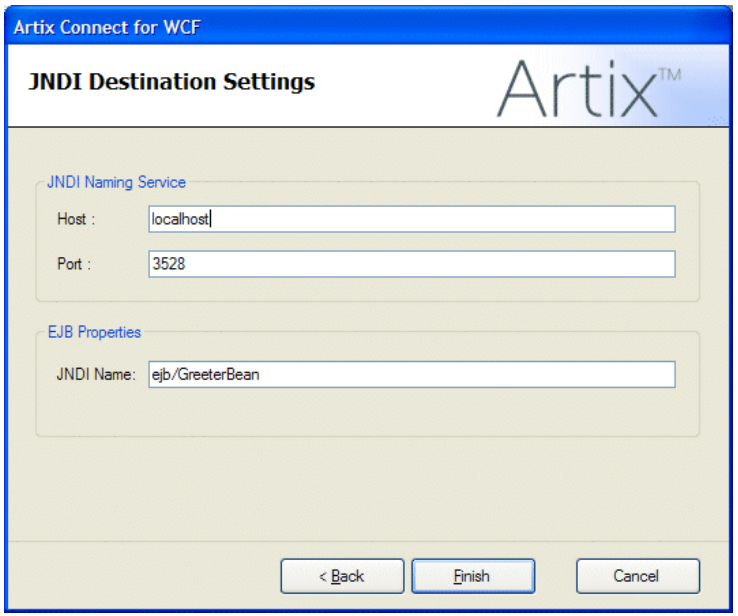
The Artix Connect for WCF wizard opens as shown in Figure 4 on page 28. The deployed clients list is empty if you have not already deployed any clients.

Connecting to an EJB

To connect to an EJB:

1. In the Artix Connect for WCF wizard, click **New Client**.
2. In the New Client window, select the **EJB** radio button.
3. In the EJB Interface Selection window, click **Browse** and browse to and select the EJB JAR file associated with the EJB that you want to invoke.
4. In the JNDI Destination Settings dialog you need to set EJB destination information (see Figure 25 on page 84. This information is specific to the EJB to which you want to connect and the application server that you are using.

Figure 25. JNDI Destination Settings: JBoss Sample



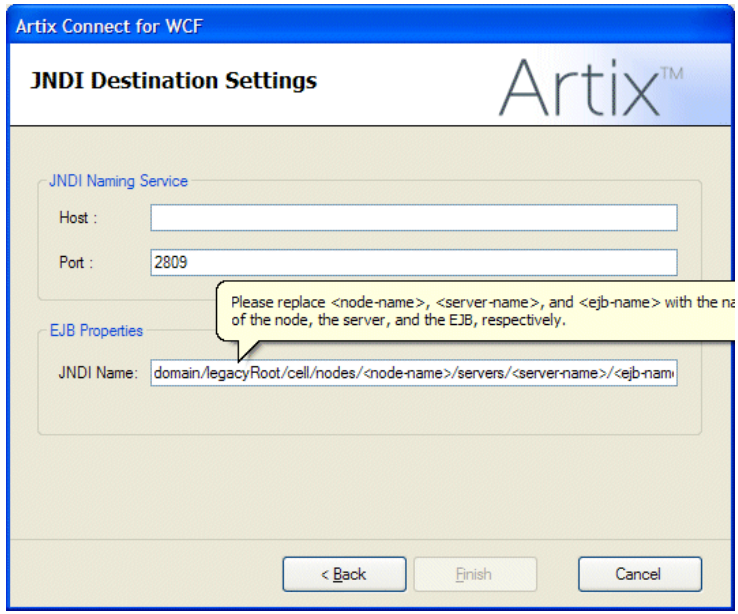
Fill in the JNDI Destination Settings window. The fields are described in Table 3 on page 85.

Sample: The settings shown in Figure 25 on page 84 are those required by the sample application using JBoss on the same host.

Table 3. EJB Destination Settings

Field	Description
JNDI Naming Service, Host:	Specifies the name of the host on which the application server's naming service is running.
JNDI Naming Service, Port:	Specifies the port on which the application server's naming service is listening.
EJB Properties, JNDI Name:	*JBoss* and *WebLogic:* specifies the JNDI name associated with the EJB, as referenced in the EJB configuration file (`jboss.xml` and `weblogic-ejb-jar.xml` respectively). *WebSphere:* specifies the JNDI name, including the qualified path and prefix; that is, the `node-name`, `server-name` and `ejb-name` (see Figure 26 on page 86 for example). **Note** `ejb-name` is specified in the EJB configuration file, `ejb-jar.xml`. To find the `node-name` and the `server-name`, run the WebSphere `dumpNameSpace.bat` utility, which is located in the following directory on your WebSphere machine: *`WebSphereInstallDir`*`\bin`

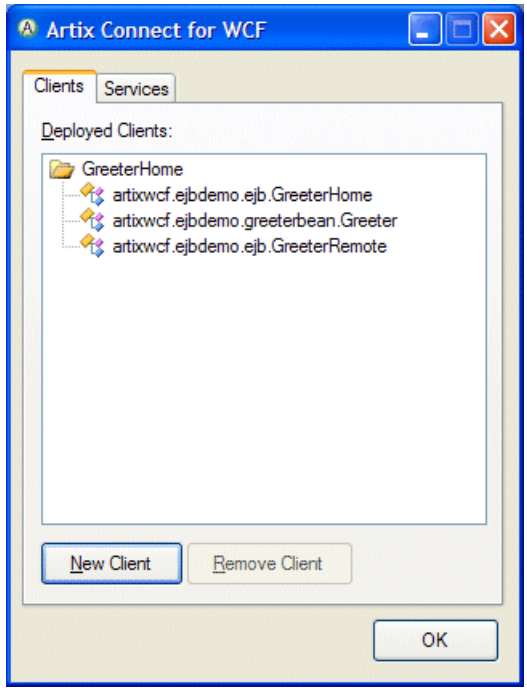
Figure 26. EJB Destination Settings: WebSphere Sample



5. Click **Finish**.

   The wizard completes its tasks and the EJB client is listed under Deployed Clients, as shown, for example, in Figure 27 on page 87.

Figure 27. EJB Sample Client Deployed



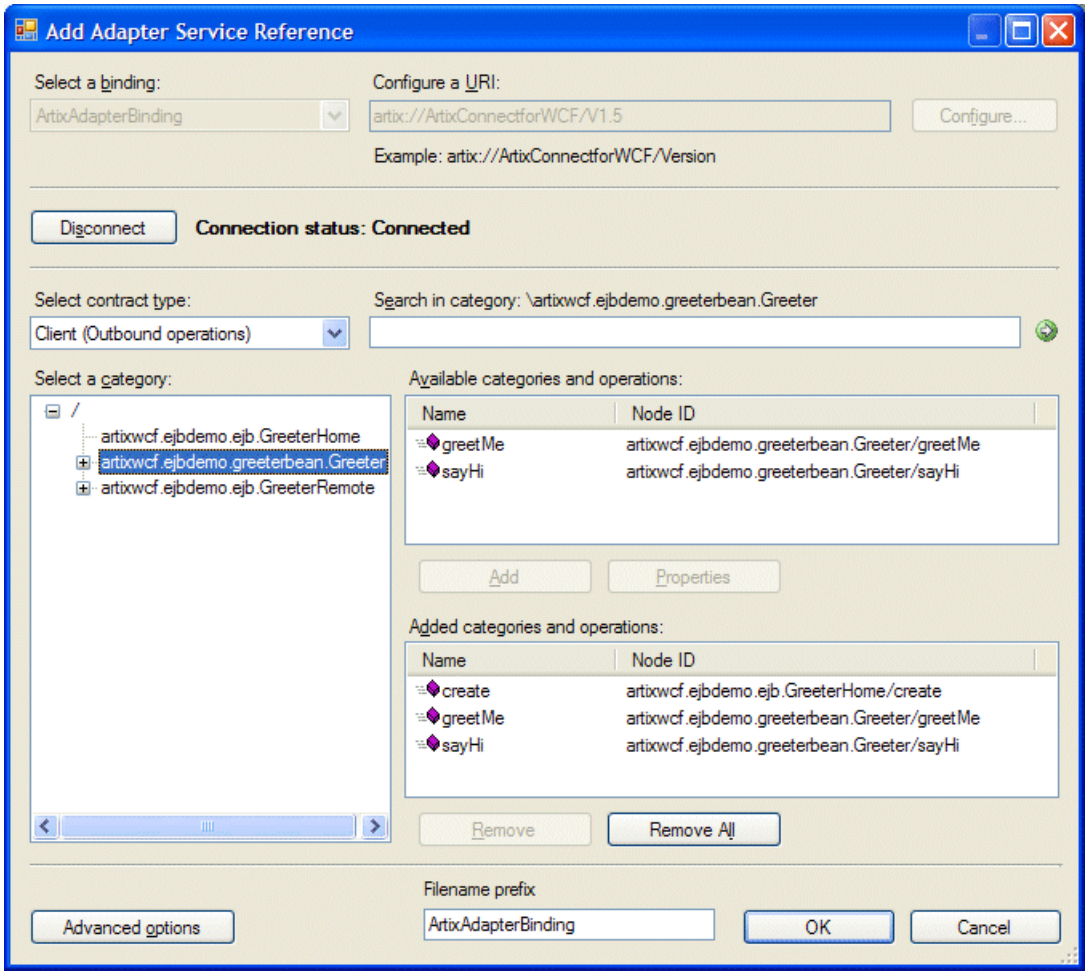
6. Click **OK**.

   The Artix Connect for WCF wizard completes and returns to the Add Adapter Service Reference wizard. It may take a few moments for the Add Adapter Service Reference wizard to become responsive while the EJB details are processed.

Making EJB operations available to your WCF application

The Add Adapter Service Reference wizard lists the EJB client in the **Select a category** panel (see, for example, Figure 28 on page 88. The **OK** button is disabled. It remains so until you specify which operations you want to use within your WCF application code.

Figure 28. Making EJB Operations Available to .NET Applications: Sample Application



To make EJB operations available to .NET, complete the following steps:

1. In the Add Adapter Service Reference wizard, under the **Select a category** panel, select the EJB interfaces that you want to use.
2. In the **Available categories and operations** panel, select the operations that you want to use.

3. Click **Add** to add the operations to the **Added categories and operations** panel.

   Sample:

   For example, if running the sample application:

   i. Select the `artixwcf.ejbdemo.ejb.GreeterHome` interface and add the `create` operation.

   ii. Select the `artixwcf.ejbdemo.greeterbean.Greeter` interface and add the `greetMe` and `sayHi` operations.

4. Click **OK**.

The wizard starts to generate code and configuration to enable your WCF application to use these operations. Your project is modified to include new code that presents the EJB as a native WCF endpoint. Now you can simply write .NET code to access the EJB.

Sample code

For example, take a look at the code in Example 10 on page 89. It is taken from the `Program.cs` file that is part of the EJB sample's `GreeterEJBClient` project. It illustrates how to write a .NET client to connect to an EJB. The basic steps are:

1. Create an EJB home interface proxy with which you can get a reference to the EJB remote interface.
2. Create an EJB remote interface proxy.
3. Invoke on the EJB.

Example 10. .NET Client Connecting to EJB

```
using System;
using System.Collections.Generic;
using System.Text;
using System.ServiceModel;

namespace GreeterEJBClient
    {
        class Program
        {
            static void Main()
```

```
            {
                try
                {
                    ❶artixwcfejbdemoejbGreeterHomeClient beanHomeClient = new artixwcfejb
demoejbGreeterHomeClient();

                    EndpointReferenceType beanHomeRef = beanHomeClient.create();

                    string beanRemoteAddress = beanHomeRef.Address.Value;

                    SecurityMode security = SecurityMode.None;
                    WSHttpBinding binding = new WSHttpBinding(security);

                    ❷artixwcfejbdemogreeterbeanGreeterClient beanRemoteClient = new artix
wcfejbdemogreeterbeanGreeterClient(
                    binding,
                    new EndpointAddress(beanRemoteAddress)
                    );

                    ❸string ret_value = beanRemoteClient.sayHi();
                    Console.WriteLine("Response to sayHi call :\n " + ret_value);

                    ret_value = beanRemoteClient.greetMe("Artix Connect for WCF");
                    Console.WriteLine("Response to greetMe call :\n " + ret_value);
                }
            }
        }
    }
```

The code shown can be explained as follows:

❶ Creates a proxy to the EJB home interface and calls the `create()` method on the EJB home interface to get a reference to the bean's remote object. The returned remote reference is a standard WS-Addressing EPR, which includes the remote bean address.

This version of Artix Connect for WCF does not support security. As a result the security mode is set to none.

❷ Creates a proxy to the EJB remote interface using the standard WSHttp binding and the remote bean address.

❸ Invokes on the EJB's operations using the remote bean reference.

Running the sample application

You can run the sample application from within Visual Studio or from the command line. You should get the response shown in Example 11 on page 91.

Example 11. Response from EJB

```
Response to sayHi call :
Hi from an EJB
Response to greetMe call :
Hi Artix Connect for WCF from an EJB
```

Deploying Your Applications

This chapter describes how to deploy your Artix Connect for WCF applications.

Introduction

Background

You can export all of the applications that you have developed on your development machine for deployment on any number of runtime machines. Deploying the applications on a runtime machine can be done when you are installing the runtime (see the Installation Guide) or after you have installed the runtime, as described in this chapter.

Exporting applications

When you export the applications, Artix Connect for WCF creates a `.zip` file that contains the applications' WSDL files and deployment descriptors. The `.zip` file is saved to the following directory of your Artix Connect for WCF installation:

InstallDir`\Visual Studio Adapter\artifacts\deploymentbundles`

Importing applications

When you import the applications onto your runtime machine, Artix Connect for WCF unzips the contents of the `.zip` file to the following directory of your Artix Connect for WCF installation:

InstallDir`\Visual Studio Adapter\artifacts\descriptors`

Exporting Your Applications

Deployment Steps: Exporting

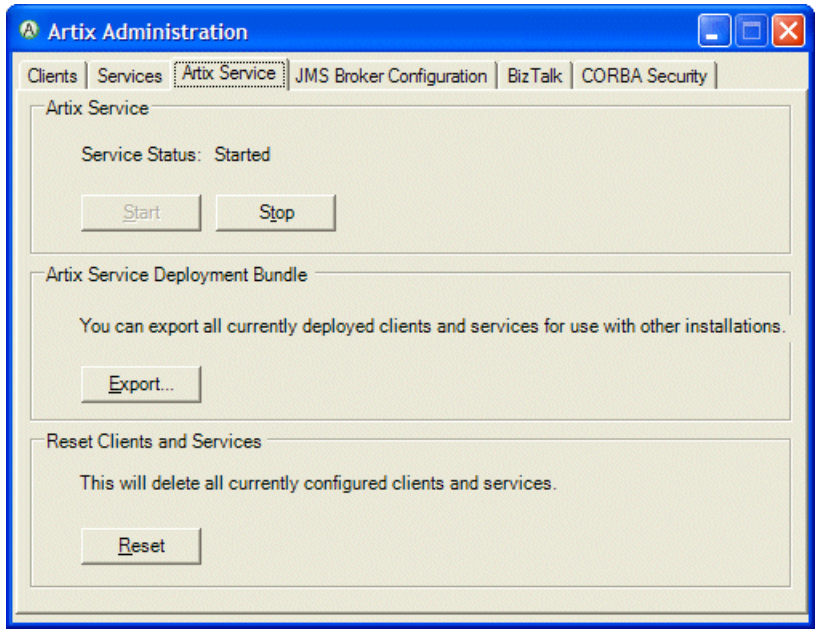
To export the applications that you have developed using Artix Connect for WCF so that they can be imported and deployed on to a runtime machine:

1. Launch the Artix Administrator tool from the Windows **Start** menu as follow:

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

   The Artix Administrator tool launches as shown in Figure 29 on page 95.

   Figure 29. Exporting Applications



2. Click **Export**.

3. In the Export Artix Service Deployment Bundle window:

   i. In the **Artix Service Deployment Bundle Name:** field, type the name that you want to use for the deployment bundle.

ii. Click **OK**.

4. You are prompted that the deployment bundle is saved to the following directory of your Artix Connect for WCF installation:

   ```
   InstallDir\Visual Studio Adapter\artifacts\deploymentbundles
   ```

   Click **OK**.

Exporting secure CORBA clients

If your are exporting a secure CORBA client, please be aware of the following warnings:

Warning

- The certificates must be stored in the same location on the deployment machine as they are on the development machine.

  The CORBA security sample application uses demonstration certificates. These should not be used in a real-life deployed system.

- The client certificate password, user name and password are stored in plain text along with the other security settings in the `GlobalCredentials.pwf` and `security.cfg` files, which are contained in the deployment zip file. You must ensure that the zip file is stored in a secure location.

Importing Your Applications

Deployment Steps: Importing

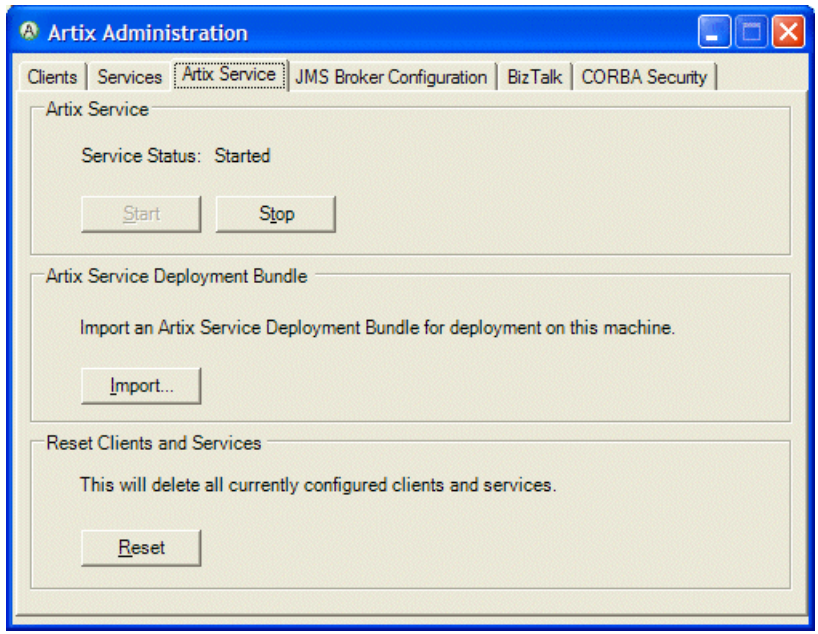
To import and deploy applications that you have developed using Artix Connect for WCF on to a runtime machine:

1. On the runtime machine, launch the Artix Administrator tool from the Windows **Start** menu as follow:

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

   The Artix Administration tool launches as shown in Figure 30 on page 97.

   Figure 30. Importing and Deploying Applications



2. Click **Import**.

3. In the Import Artix Service Deployment Bundle window, click **...** and navigate to the following Artix Connect for WCF directory on the development machine:

```
InstallDir\Visual Studio Adapter\artifacts\deploymentbundles
```

4. Select the deployment bundle and click **Open**.
5. In the Import Artix Service Deployment Bundle window, click **OK**.

   Artix Connect for WCF unzips the deployment bundle into the following directory of your Artix Connect for WCF installation:

   ```
   InstallDir\Visual Studio Adapter\artifacts\descriptors
   ```

6. You are prompted when the deployment bundle has been successfully imported. Click **OK**.
7. If the Artix service is running before you import your applications, it automatically restarts when your applications have been imported and you can exit the Artix Administration tool.

   If the Artix service is not running before you import your applications, start it manually by clicking **Start** in the Artix Service panel of the Artix Administration tool and exit the tool.

Importing secure CORBA clients

If your are importing a secure CORBA client, please be aware of the following warnings:

Warning

- The certificates must be stored in the same location on the deployment machine as on the development machine.

  The CORBA security sample application uses demonstration certificates. These should not be used in a real-life deployed system.

- Once extracted, the client certificate password, user name and password are stored in plain text along with the other security settings in the following files in your Artix Connect for WCF installation:

  ```
  InstallDir\Visual Studio
  Adapter\artifacts\descriptors\GlobalCredentials.pwf
  ```

InstallDir\Visual Studio Adapter\artifacts\domains\security.cfg

You must set the file permissions appropriately to ensure that both the confidentiality and the integrity of the password data are protected.

Using the Artix Administration Tool

This chapter describes how to use the Artix Administration tool.

Introduction

Overview

Artix Connect for WCF includes an Artix Administration tool that enables you to:

- View, add and remove clients and services. on page 104
- Export your applications. on page 95
- Import your applications. on page 97
- Stop, start and reset the Artix Service. on page 107
- Configure JMS broker settings. on page 108
- Configure Artix Connect for WCF for use with BizTalk. in the *BizTalk Integration Guide*
- Configure CORBA security. on page 24

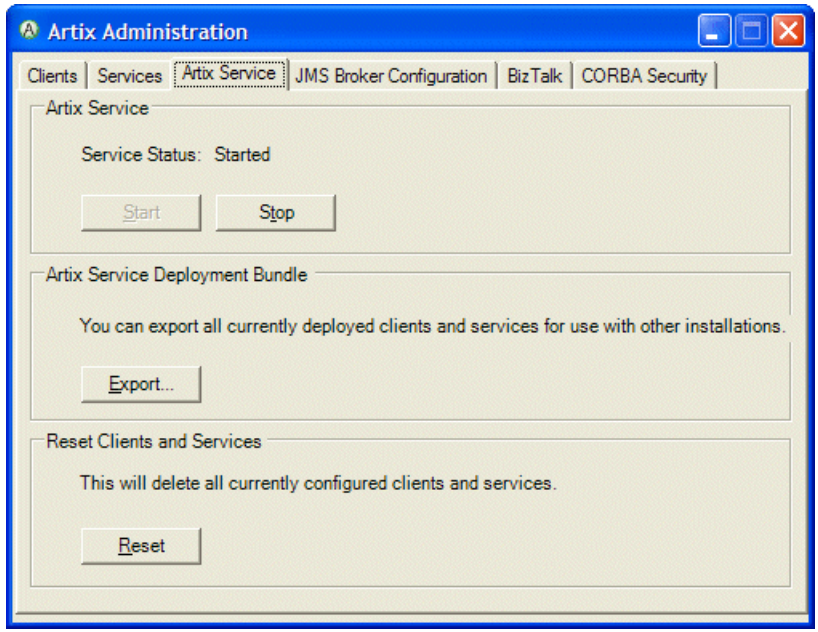
Launching Artix Administration tool

1. Launch the Artix Administration tool from the Windows Start menu:

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

   It appears as shown in Figure 31 on page 103.

Figure 31. Artix Administration Tool



Viewing, Adding and Removing Clients and Services

Viewing clients and services

To view the clients and services that you have already deployed:

1. Launch the Artix Administration tool from the Windows Start menu:

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

2. To view clients that you have already deployed, select the **Clients** tab.

   To view services that you have already deployed, select the **Services** tab.

Removing clients and services

To remove a client or service that you have deployed:

1. Launch the Artix Administration tool from the Windows Start menu:

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

2. To remove a client:

   i. Select the **Clients** tab.

   ii. Select the client that you want to remove.

   iii. Click **Remove Client**.

   To remove a service:

   i. Select the **Services** tab.

   ii. Select the service that you want to remove.

   iii. Click **Remove Service**.

Adding clients and services

When you run the Artix Connect for WCF wizard from within Visual Studio, as described in the other chapters in this book, it generates the code needed to connect your .NET application to the selected back-end as well as generating the deployment descriptors needed to deploy your application. You can, however, launch the Artix Connect for WCF wizard from the Artix Administrator tool and use it to create a client or service that has not already been developed or you can remove an already deployed service without

opening Visual Studio LOB Adapter. This can be used when you want to quickly design a service and deploy it.

Adding a client (outbound)

To add a client:

1. Launch the Artix Administration tool from the Windows Start menu:

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

2. Select the **Clients** tab.

3. Select **New Client**.

   This launched the Artix Connect for WCF wizard.

4. Select the type of client that you want to deploy and follow the instructions outlined in Table 4 on page 105.

Table 4. Artix Administration Tool: Deploying Clients

Client Type	Instructions
JMS	Complete steps 2–4 in Connecting to JMS on page 58, all of the steps in Selecting a payload format on page 60, and steps 1–3 in Specifying JMS destination settings on page 63. The JMS client is added to the list of deployed clients in the Artix Administration tool.
CORBA	Complete steps 2–8 in Step 2: Adding a CORBA client on page 28. The CORBA client is added to the list of deployed clients in the Artix Administration tool.
EJB	Complete steps 2–5 in Connecting to an EJB on page 84. The EJB client is added to the list of deployed clients in the Artix Administration tool.

Adding a service (Inbound)

To add a service:

1. Launch the Artix Administration tool from the Windows Start menu:

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

2. Select the **Services** tab.

3. Select **New Service**.

   This launched the Artix Connect for WCF wizard.

4. Select the type of service that you want to add. This release of Artix Connect for WCF supports JMS services.

5. Follow the instructions outlined in Consuming JMS Messages on page 70to add your service.

Stopping, Starting and Resetting the Artix Service

Steps

To stop, start or reset the Artix service:

1. Launch the Artix Administration tool from the Windows Start menu:

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

   It appears as shown in Figure 31 on page 103.

2. Select the Artix Service tab.

3. Click the appropriate button.

Configuring a JMS Broker

Steps

If you want to change the JMS broker configuration that you set when you were using the Artix Connect for WCF wizard for the first time, you can use the Artix Administration tool as follows:

1. Launch the Artix Administration tool from the Windows Start menu:

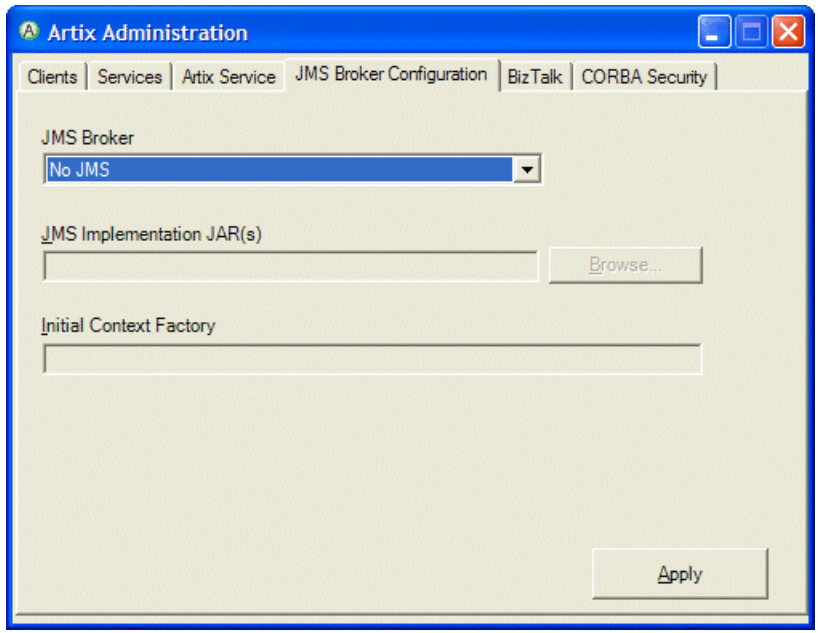
   (All) Programs | IONA | Artix Connect For WCF Beta | Artix Administration

   It appears as shown in Figure 31 on page 103.

2. Select the JMS Broker Configuration tab.

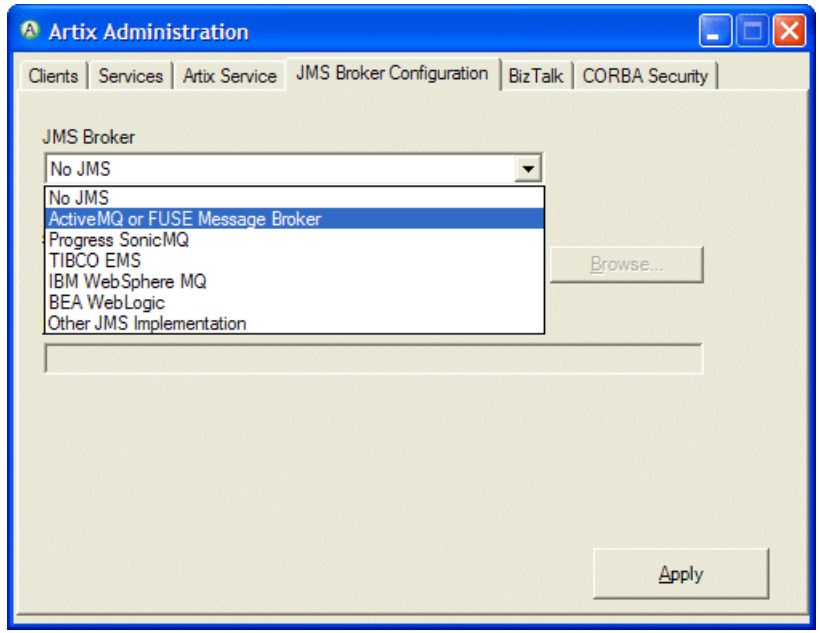
3. In the JMS Broker Configuration window, as shown in Figure 32 on page 108, complete the following steps:

   Figure 32. Artix Administration Tool: JMS Broker Configuration



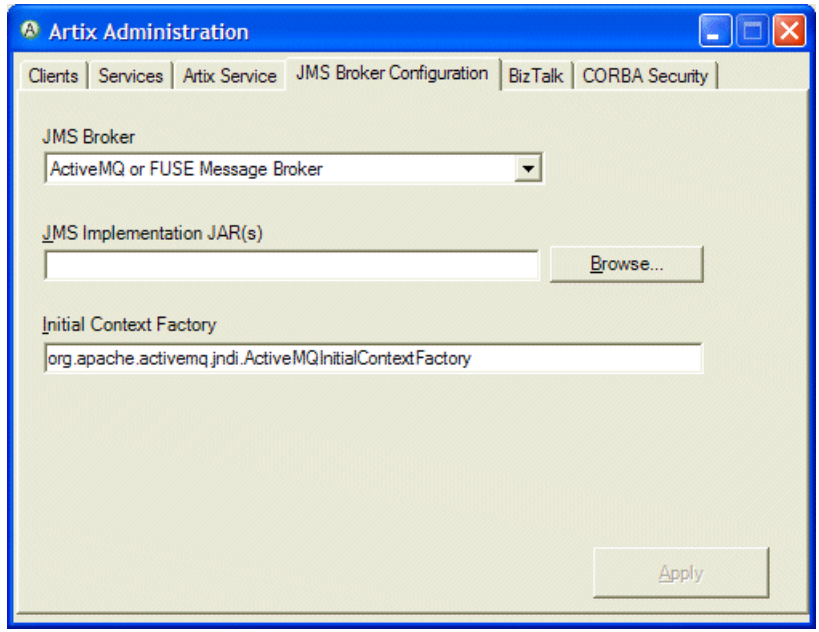
i. In the JMS Broker field, select the JMS broker that you want to use from the drop-down list (see Figure 33 on page 109).

Figure 33. Selecting a JMS Broker



The Initial Context Factory field is automatically filled in for you. For example, if you have selected ActiveMQ or FUSE Message Broker as your JMS broker, the Initial Context Factory is shown in Figure 34 on page 110.

Figure 34. Setting the Initial Context Factory



ii. Select the JMS implementation JAR associated with the JMS broker that you selected. For example, in the case of FUSE Message Broker 5.0.0.20, the implementation JAR is located in `FUSEMessageBrokerInstallDir\` and is called `activemq-all-5.0.0.20-fuse.jar`.

    For a list of the implementation JARs for all supported JMS brokers, see JMS Broker Implementation JARs in the *Installation Guide*.

iii. Click **Apply**.

Logging

This chapter describes how to configure logging for the Artix service and the Artix Adapter Framework.

Introduction

Overview

Artix Connect for WCF supports logging at the level of the:

- *Artix service:* see Artix Service Logging on page 113.
- *Artix Adapter Framework:* see Artix Adapter Framework Logging on page 120.

For more detail about these components and how they fit into the Artix Connect for WCF architecture, see Architecture on page 18.

Artix Service Logging

Configuration variables and plug-ins

Artix service logging is based on Artix logging. It is controlled by the `event_log:filters` configuration variable and the log stream plug-ins (for example, `local_log_stream` and `xmlfile_log_stream`). It is configured in the `artix_wcf.cfg` configuration file, which is located in the following directory of your Artix Connect for WCF installation:

InstallDir\Visual Studio Adapter\artifacts\domains

Default settings

Artix Connect for WCF includes some default settings for Artix service logging. Example 12 on page 113 shows the relevant section of the `artix_wcf.cfg` file.

Example 12. Default Settings for Artix Service Logging

```
❶orb_plugins = ["local_log_stream", "iiop_profile", "giop",
"iiop", "java", "mq"];
❷event_log:filters = ["*=FATAL+ERROR"];
❸plugins:local_log_stream:filename = "C:/Program
Files/IONA/Artix Connect For WCF/Visual Studio Adapter/arti
facts/log/artix_wcf.log";
```

The configuration shown in Example 12 on page 113 can be explained as follows:

❶ Adds the `local_log_stream` plug-in to the list of plug-ins used by Artix Connect for WCF. This is required for logging.

❷ Configures the level of logging to display errors only.

❸ Configures the `local_log_stream` plug-in to publish the log messages in a log file, `artix_wcf.log`, in the following directory of your Artix Connect for WCF installation:

InstallDir\Visual Studio Adapter\artifacts\domains\log

The rest of this chapter describes how you can change the default settings.

Configuring Logging Levels

Log message severity levels

Artix Connect for WCF supports the following levels of log message severity:

Table 5. Artix Service Logging Severity Levels

Severity Level	Description
Information	Information messages report significant non-error events. These include server startup or shutdown, object creation or deletion, and details of administrative actions. Information messages provide a history of events that can be valuable in diagnosing problems. Information messages can be set to low, medium, or high verbosity.
Warning	Warning messages are generated when the Artix service encounters an anomalous condition, but can ignore it and continue functioning. For example, encountering an invalid parameter and ignoring it in favor of a default value.
Error	Error messages are generated when the Artix service encounters an error. Artix might be able to recover from the error, but might be forced to abandon the current task. For example, an error message might be generated if there is insufficient memory to carry out a request.
Fatal error	Fatal error messages are generated when the Artix service encounters an error from which it cannot recover. For example, a fatal error message is generated if the Artix service cannot find its configuration file.

Log level syntax

Artix service logging is set by default to display errors (see Example 12 on page 113). You can, however, change the logging level using the syntax shown in Table 6 on page 115.

Table 6. Artix Logging Severity Levels Syntax

Severity Level Syntax	Description
INFO_LO[W]	Low verbosity informational messages.
INFO_MED[IUM]	Medium verbosity informational messages.
INFO_HI[GH]	High verbosity informational messages.
INFO[_ALL]	All informational messages.
WARN[ING]	Warning messages.
ERR[OR]	Error messages.
FATAL[_ERROR]	Fatal error messages.

Severity Level Syntax	Description
*	All messages.

Example Logging Settings

Table 7 on page 116 shows some examples:

Table 7. Artix Logging Configuration Examples

Example	Description
event_log:filters = ["=FATAL+ERROR+WARNING"];*	Displays errors and warnings only.
event_log:filters = ["=FATAL+ERROR+WARNING+INFO_MED"];*	Adding `INFO_MED` causes all request/reply messages to be logged (for all transport buffers).
event_log:filters = ["=FATAL+ERROR+WARNING+INFO_HI"];*	Displays typical trace statement output (without the raw transport buffers).
event_log:filters = ["=*"];*	Displays all logging.

Configuring Logging Output

Introduction

In addition to setting the event log filter, you must ensure that a log stream plug-in is set in your `artix_wcf.cfg` file. These include:

- `local_log_stream`, which sends logging to a text file.
- `xmlfile_log_stream`, which directs logging to an XML file.

The `local_log_stream` is set by default.

Using text log files

Artix Connect for WCF is configured by default to use the `local_log_stream`. Example 13 on page 117 shows the relevant content of the default `artix_wcf.cfg` configuration file.

Example 13. Configuring Logging Output to a Text File

```
//Ensure the local_log_stream plug-in exists in the orb_plugins
 list
orb_plugins = ["local_log_stream", ... ];
//Optional text filename
plugins:local_log_stream:filename = "C:/Program
Files/IONA/Artix Connect For WCF/Visual Studio Adapter/arti
facts/log/artix_wcf.log";
```

If you do not specify a text log file name, logging is sent to `stdout`.

Using XML log files

To configure the `xmlfile_log_stream`, set the following variables in your configuration file:

Example 14. Configuring Logging Output to an XML File

```
//Ensure the xml_log_stream plug-in is in your orb_plugins
list
orb_plugins = ["xmlfile_log_stream", ... ];
// Optional filename
plugins:xmlfile_log_stream:filename = "artix_logfile.xml";
// Optional process ID added to filename (default is false).
plugins:xmlfile_log_stream:use_pid = "false";
```

Using a rolling log file

By default, the logging plug-in creates a new log file each day to prevent the log file from growing indefinitely. In this model, the log stream adds the current date to the configured filename. This produces a complete filename, for example: `artix_wcf.log.05122008`

A new log file begins with the first event of the day, and ends each day at 23:59:59.

Specifying the date format

You can configure the format of the date in the rolling log file, using the following configuration variables:

- `plugins:local_log_stream:filename_date_format`
- `plugins:xmlfile_log_stream:filename_date_format`

The specified date must conform to the format rules of the ANSI C `strftime()` function. For example, for a text log file, use the following settings:

```
plugins:local_log_stream:rolling_file="true";
plugins:local_log_stream:filename="my_log";
plugins:local_log_stream:filename_date_format="_%Y_%m_%d";
```

On the 31st May 2008, this results in a log file named `my_log_2008_05_31`.

The equivalent settings for an XML log file are:

```
plugins:xmlfile_log_stream:rolling_file="true";
plugins:xmlfile_log_stream:filename="my_log";
plugins:xmlfile_log_stream:filename_date_format="_%Y_%m_%d";
```

Disabling rolling file behavior

To disable rolling file behavior for a text log file, set the following variable to false:

```
plugins:local_log_stream:rolling_file = "false";
```

To disable rolling file behavior for an XML log file, set the following variable to false:

```
plugins:xmlfile_log_stream:rolling_file = "false";
```

Artix Adapter Framework Logging

Introduction

Artix Connect for WCF uses `log4net` for Artix Adapter Frameowrk logging.

log4net is an open source logging tool from the Apache Logging Services Project (http://logging.apache.org/). It is part of the log4j framework for the .NET runtime.

Logging output

Log statements are written to the following file:

InstallDir`\Visual Studio Adapter\artifacts\log\ArtixAdapter.log`

Configuration

You cannot configure logging level and output type in this version of Artix Connect for WCF. This functionality will, however, be added in a future release.

Index

L

M

O

S

W