Artix Connect for WCF

User's Guide

Version 1.0
May 2008

Making Software Work Together™

User's Guide

IONA Technologies

Version 1.0

Published 29 May 2008

Trademark and Disclaimer Notice

Copyright Notice

Table of Contents

List of Figures

List of Tables

List of Examples

Preface

Table of Contents

The Artix Connect for WCF Library

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

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

Document Conventions

Typographical conventions

This book uses the following typographical conventions:

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

Keying conventions

This book uses the following keying conventions:

No prompt	When a command's format is the same for multiple platforms, the command prompt is not shown.
`>`	The notation `>` represents the MS-DOS or Windows command prompt.
`. . .`	Horizontal or vertical ellipses in format and syntax descriptions indicate that material has been eliminated to simplify a discussion.
`[ ]`	Brackets enclose optional items in format and syntax descriptions.
`{ }`	Braces enclose a list from which you must choose an item in format and syntax descriptions.
`|`	In format and syntax descriptions, a vertical bar separates items in a list of choices enclosed in `{}` (braces).

Admonition conventions

This book uses the following conventions for admonitions:

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

What is Artix Connect for WCF?

Summary

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.

Table of Contents

Use Cases

Product description

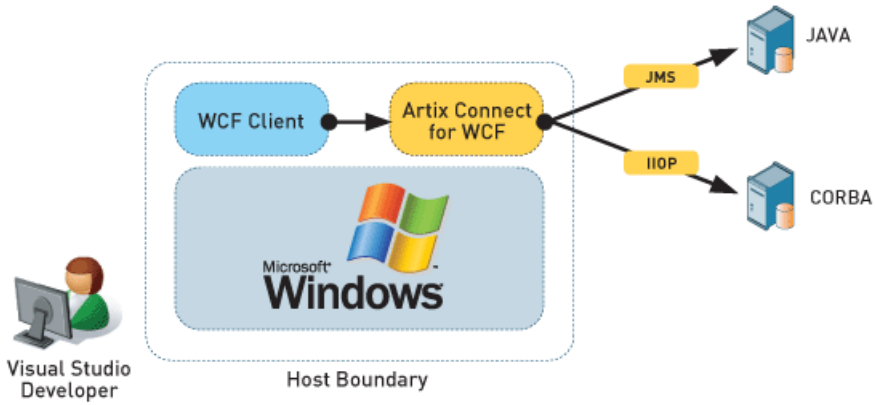
Artix Connect for WCF integrates Microsoft's new .NET communication technology, Windows Communications Foundation (WCF), with many diverse middleware and messaging technologies, including CORBA and Java (see Figure 1, "Artix Connect for WCF Use Cases". It offers you, as a .NET developer, the ability to communicate with these systems without having to leave your Visual Studio environment or without being forced to use unfamiliar techniques.

Supported technologies

This release enables you to connect to CORBA services and Java Messaging Services (JMS) queues and topics. Support for enterprise Java beans (EJBs) is planned for a future release.

Graphical representation

Figure 1. Artix Connect for WCF Use Cases

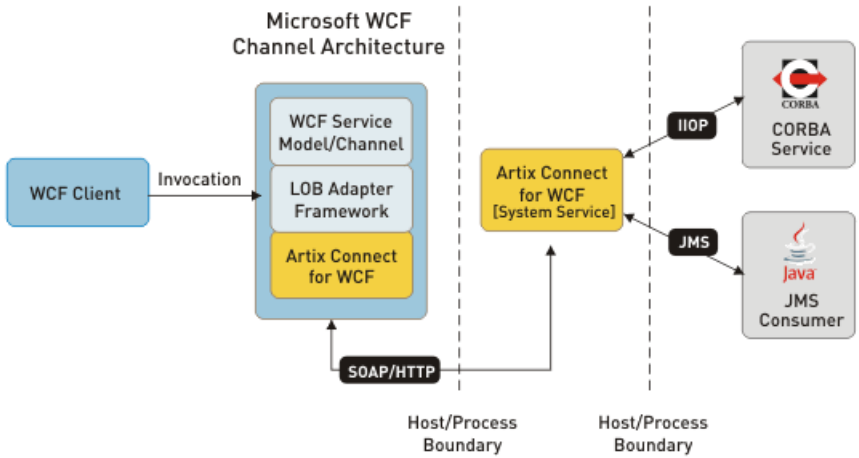


Architecture

Graphical representation

Artix Connect for WCF's architecture is shown in Figure 2, “Artix Connect for WCF Architecture”.

Figure 2. Artix Connect for WCF Architecture



Components

As shown in Figure 2, “Artix Connect for WCF Architecture”, Artix Connect for WCF consists of two main components:

- An LOB Adapter, which plugs in to the Microsoft LOB Adapter Framework. The LOB Adapter includes a wizard that enables you to design and configure LOB services such as CORBA and Java.

  For more information and instructions on how to use the LOB Adapter and its wizard, see *Connecting to CORBA Services* and *Connecting to JMS Queues and Topics*.

- An Artix Service, which runs as Windows system service and is responsible for monitoring deployed LOB 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*.

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 Services

Summary

This chapter describes how to use Artix Connect for WCF to connect to a CORBA service.

Table of Contents

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 CORBA service to which you want to connect.

Example 1, "Stock Quote System—IDL" is taken from the sample application described in the Getting Started Guide. Clients of the service pass a stock symbol string, such as MSFT or IONA, as a parameter to the `price` operation and receive a return value simulating the market value of that stock.

Example 1. Stock Quote System—IDL

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

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 service 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, "Example of a CORBA Stringified IOR").

Example 2. Example of a CORBA Stringified IOR

```
IOR:010000003200000049444c3a696f6e612e636f6d2f49545f4f54535f536572766963654164696e2f5
365727669636541646d696e3a312e3000000001000000000000008a000000010102000800000066626f
6c74616e00030c00003f0000003a3e0232310f73696d706c652e6c6f63617469696f6e11694f5453006f7473
746d0061646d696e00175472616e73616374696f6e536572766963654164696e00020000000100000
018000000010000000100010000000000000101000100000009010100060000000600000001000000260
```

- *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, “Example of a CORBALoc”).

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, “Example of a CORBAName”).

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.

Using a factory pattern

The factory pattern is commonly used when designing CORBA services. 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 service that is responsible for creating and managing accounts. The banking service 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 services that use such a pattern. For more information, see Using the Factory Pattern.

More information

To use Artix Connect for WCF to connect to a CORBA service, 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 Service

Before you Begin

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

1. Access to the CORBA service 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

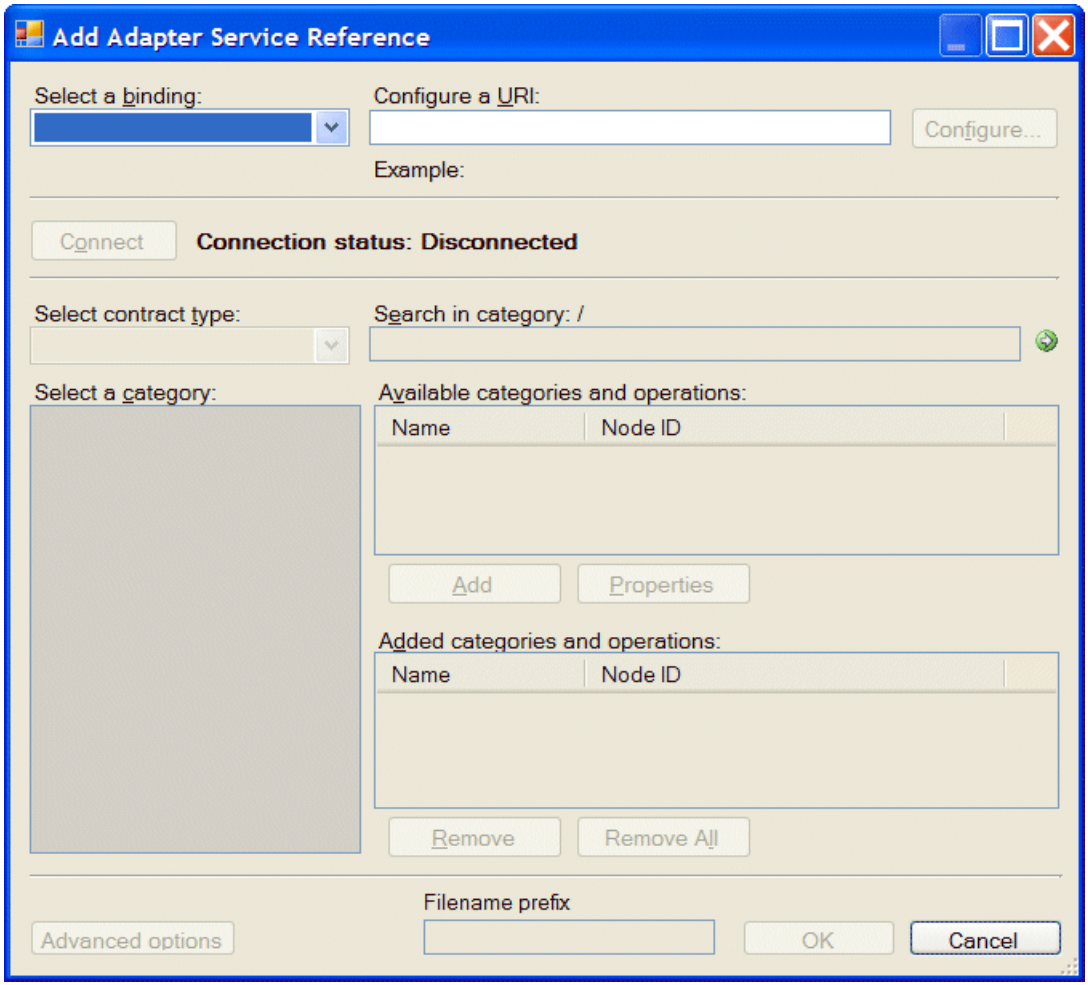
Artix Connect for WCF is a plug-in to the Microsoft LOB adapter framework. The first step in using Artix Connect for WCF is to launch this framework:

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, “Add Adapter Service Reference Wizard”.

Figure 3. Add Adapter Service Reference Wizard



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

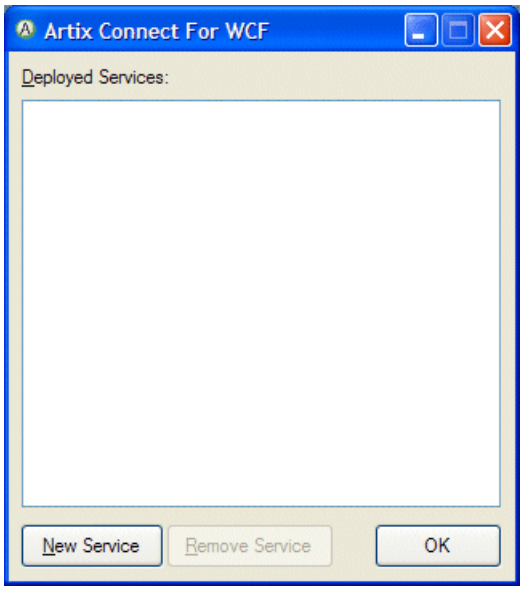
ii. Click Configure.

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, “Artix Connect for WCF Wizard”. If you have already deployed services using Artix Connect for WCF, they will be listed in under Deployed Services.

Figure 4. Artix Connect for WCF Wizard



Step 2: Adding a CORBA service

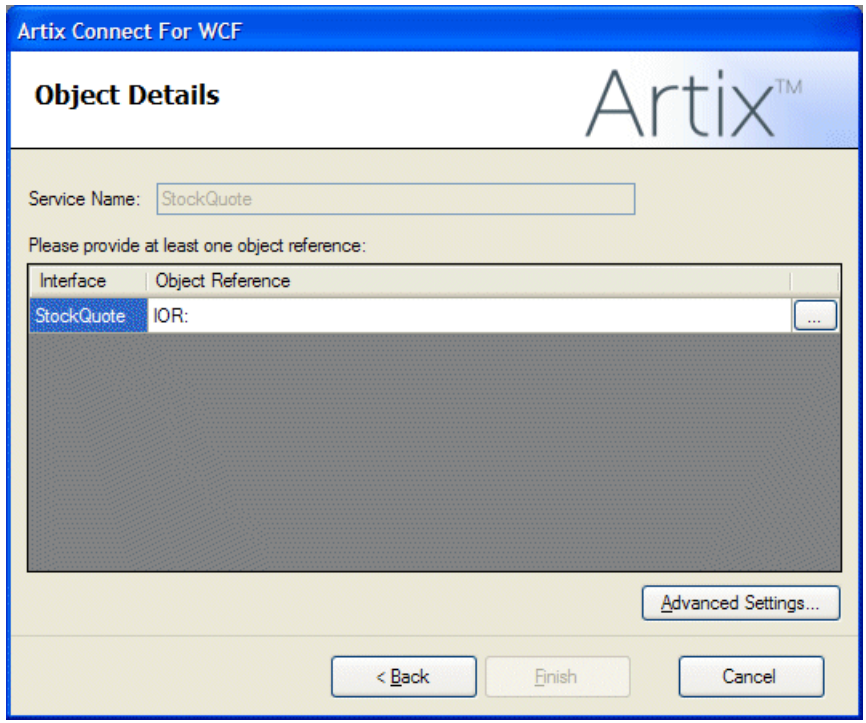
To add a CORBA service:

1. In the Artix Connect for WCF wizard, click New Service.
2. In the New Service wizard, select the CORBA radio button and click Next.
3. In the IDL File Selection window, enter the location of your CORBA service IDL file.
4. 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, “CORBA Object Details Window”).

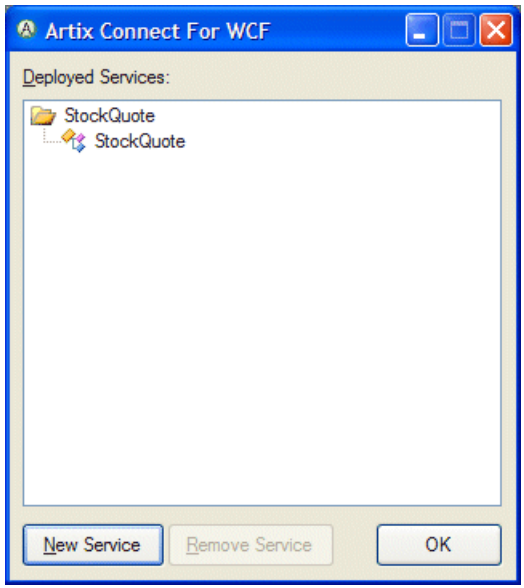
Figure 5. CORBA Object Details Window



5. Select the interface that you want to use and click ... to browse to the location of your object reference.
6. If the CORBA object that you want to connect to is persistent:
   i. Click Advanced Settings.
   ii. In the Advanced Setting window, tick the Persistent check box.
   iii. Click OK.
7. Click Finish.

The CORBA service is added to the list of deployed services (see, for example, Figure 6, “CORBA Service Deployed”).

Figure 6. CORBA Service Deployed



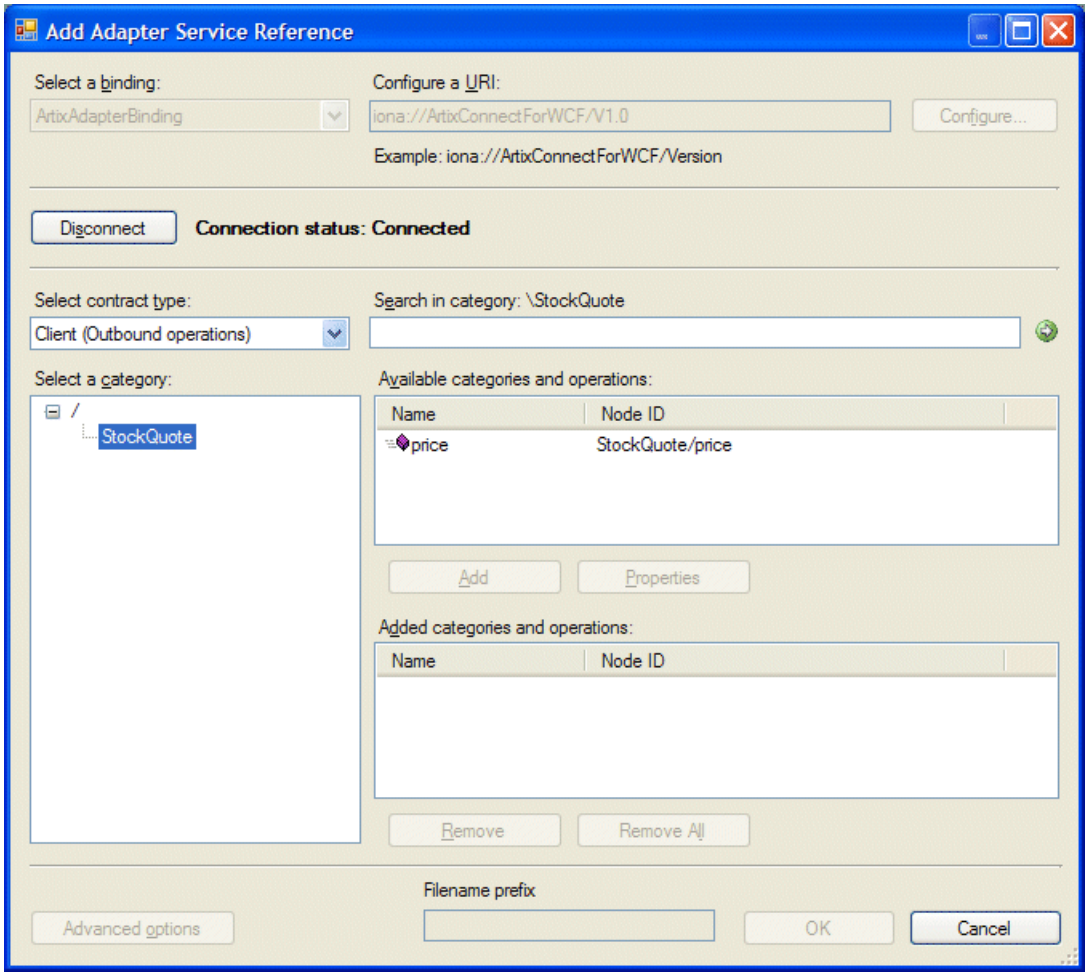
8. Click OK.

   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 service is listed in the Select a category panel of the Add Adapter Service Reference wizard (see, for example, Figure 7, “Making CORBA Operations Available to WCF”). 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 service 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.

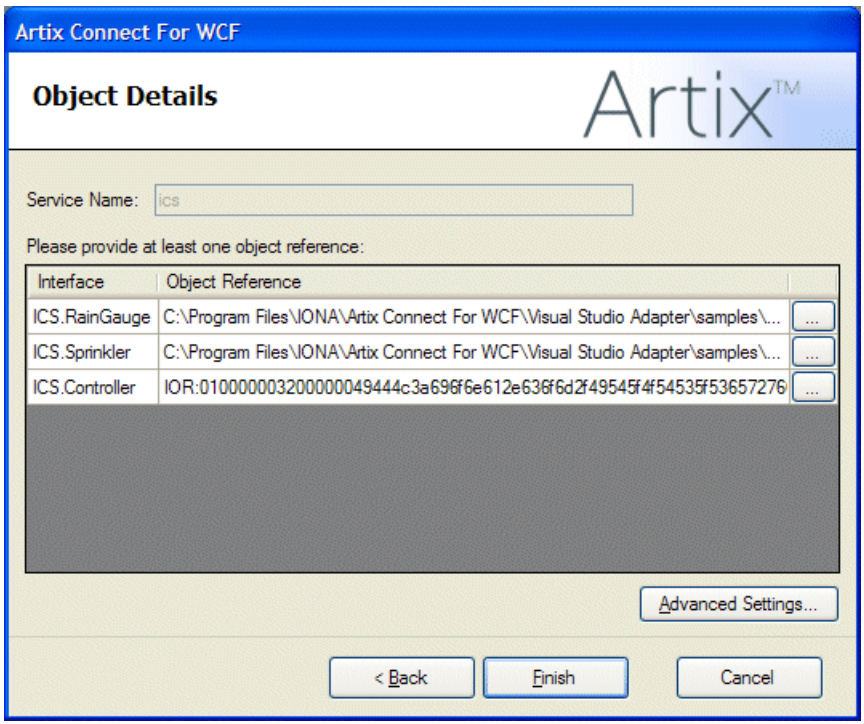
Step 4: Adding Code to Call to the CORBA Service

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 service as a native WCF endpoint. Now you can write .NET code to call the CORBA service. To see an example, run the tutorial outlined in the Getting Started Guide.

Using Multiple Interfaces

If the IDL file for the CORBA service 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, “Using IDL with Multiple Interfaces” lists the interfaces defined in an `ics.idl` file.

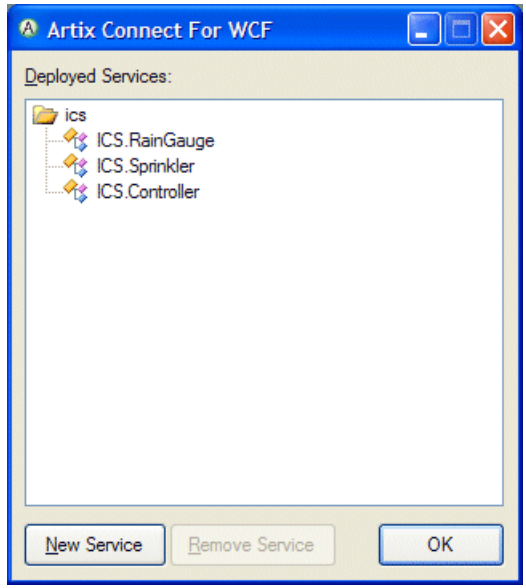
Figure 8. Using IDL with Multiple Interfaces



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 services are deployed and displayed in a tree that takes the name of the IDL file. For example, Figure 9, “CORBA: Multiple Services Deployed” shows the list of deployed services defined in the `ics.idl` file.

Figure 9. CORBA: Multiple Services Deployed



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 service 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 Service, 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

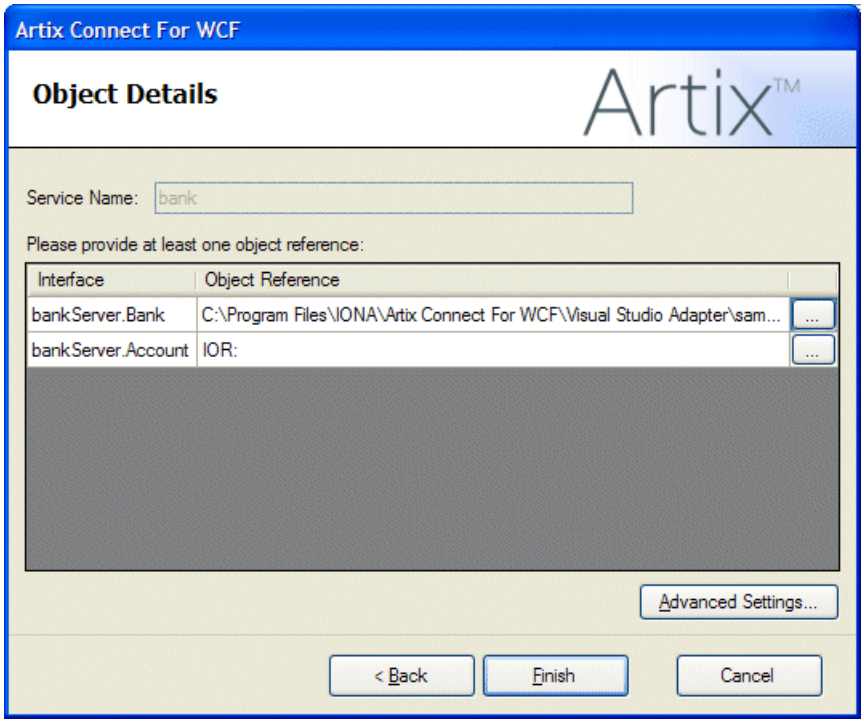
InstallDir`\Visual Studio Adapter\samples\corba\factory\etc`

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, “Selecting the Bank Factory IOR”:

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

   i. Click Advanced Settings.

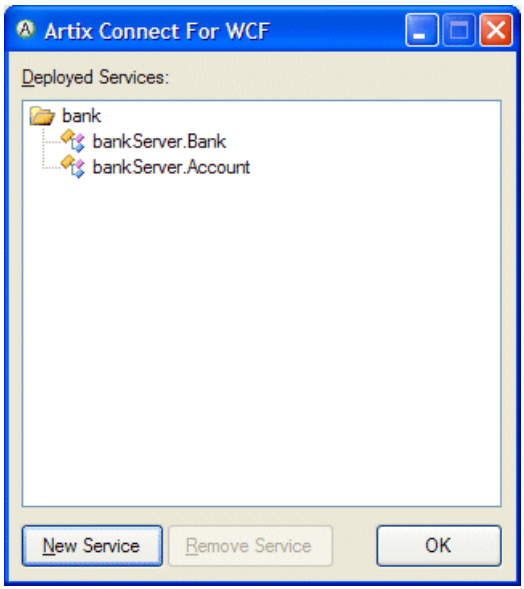
   ii. In the Advanced Setting window, tick the Persistent check box.

   iii. Click OK.

d. Click Finish.

The `Bank` and `Account` services appear in the list of deployed services, as shown in Figure 11, “Factory Pattern: Deployed Services”.

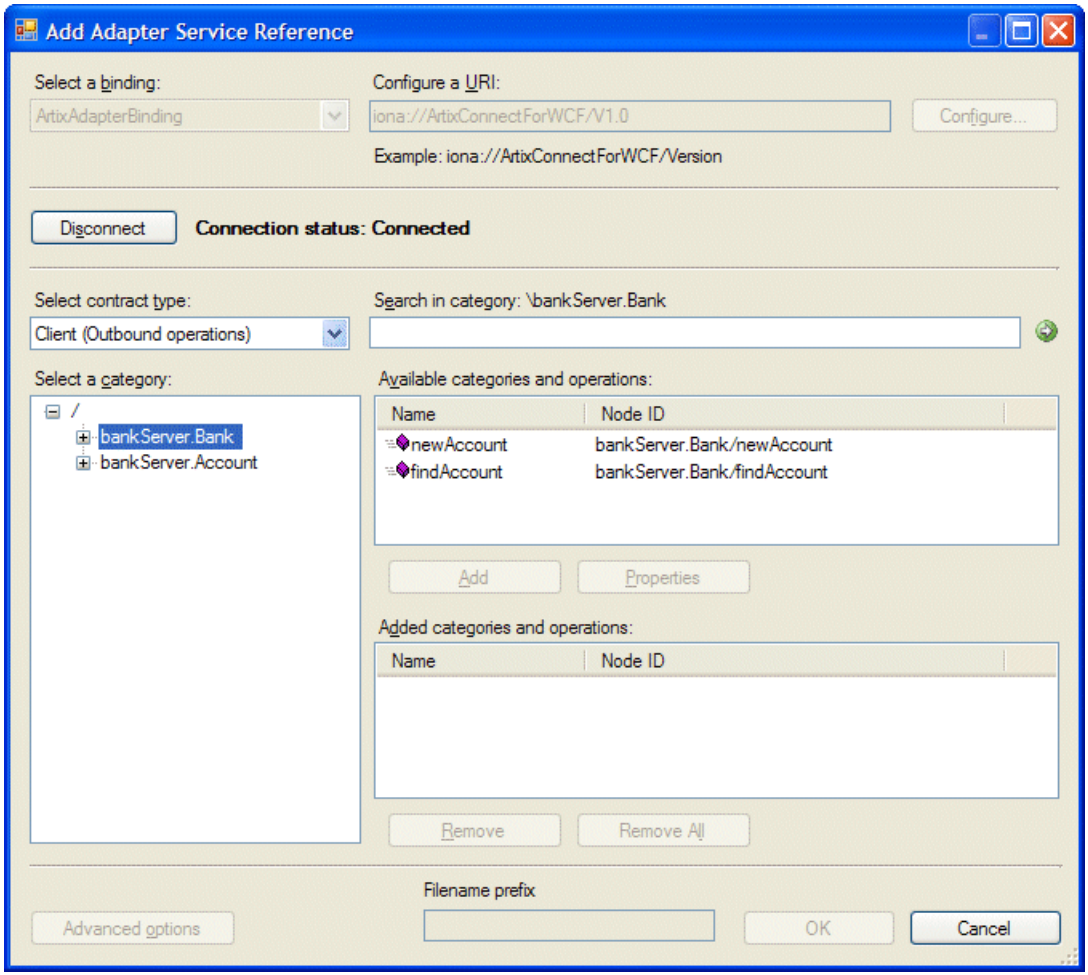
Figure 11. Factory Pattern: Deployed Services



e. Click OK.

iii. The wizard returns to the Add Adapter Service Reference wizard, where the CORBA service is listed in the Select a category panel (see Figure 12, “Add Adapter Service Reference Wizard: CORBA Services”).

Figure 12. Add Adapter Service Reference Wizard: CORBA Services



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 service 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, ".NET Client of CORBA Service that Uses a Factory Pattern" shows the relevant code from the `Program.cs` file.

Example 5. .NET Client of CORBA Service 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, ".NET Client of CORBA Service that Uses a Factory Pattern" 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.

Removing a Service

To remove a service, in the Deployed Services window of the Artix Connect for WCF wizard (see Figure 6, “CORBA Service Deployed”), select the service that you want to delete and click Remove Service.

Connecting to JMS Queues and Topics

Summary

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

Table of Contents

Introduction to JMS

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 connect to a JMS queue or topic you do not need to understand JMS in any detail. If, however, you are interested in learning more about this technology, visit the following website:

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

Connecting to JMS

Before you begin

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 15, “Adding JMS Broker Settings” and Figure 18, “JMS Destination Settings” for more detail.

Launching the Artix Connect for WCF wizard

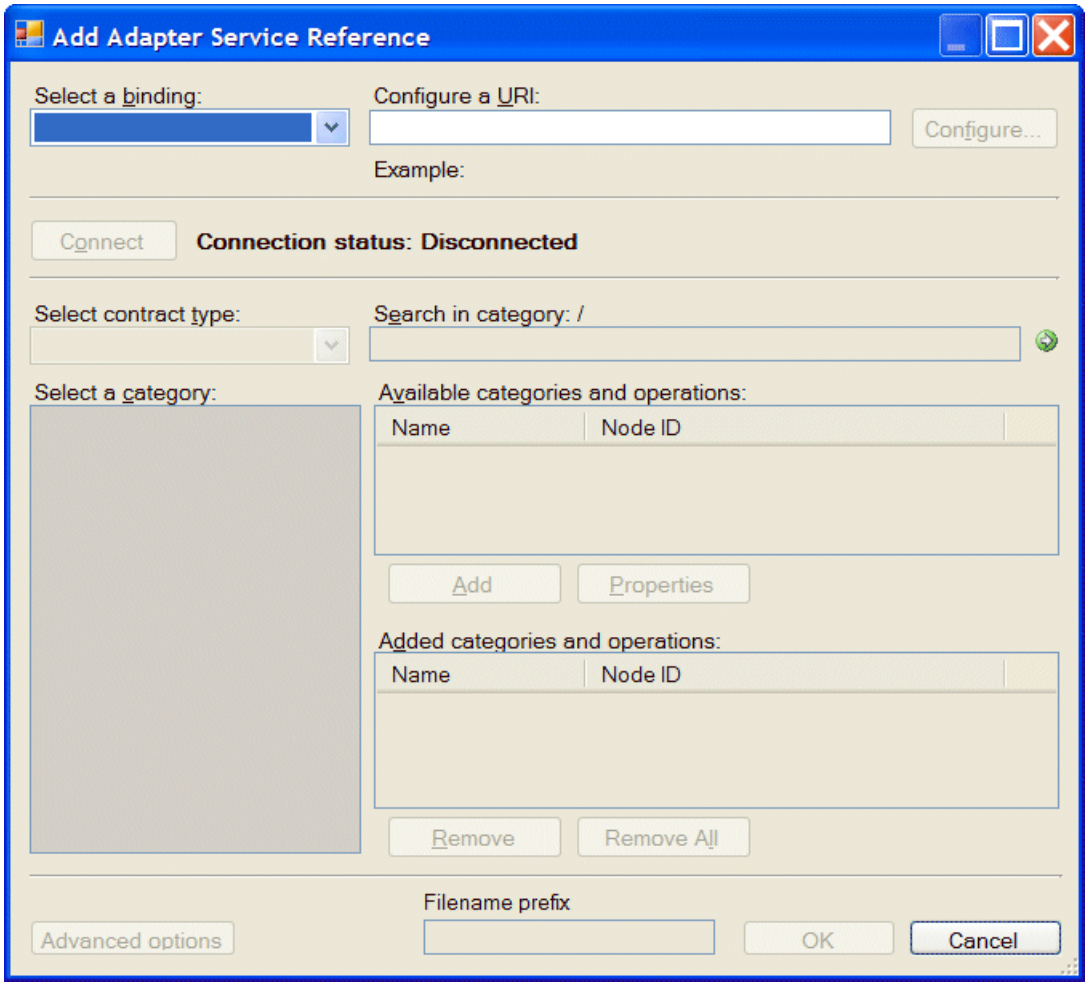
Artix Connect for WCF is a plug-in to the Microsoft LOB adapter framework. The first step in using Artix Connect for WCF is to launch this framework.

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 13, “Add Adapter Service Reference Wizard”.

Figure 13. Add Adapter Service Reference Wizard



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

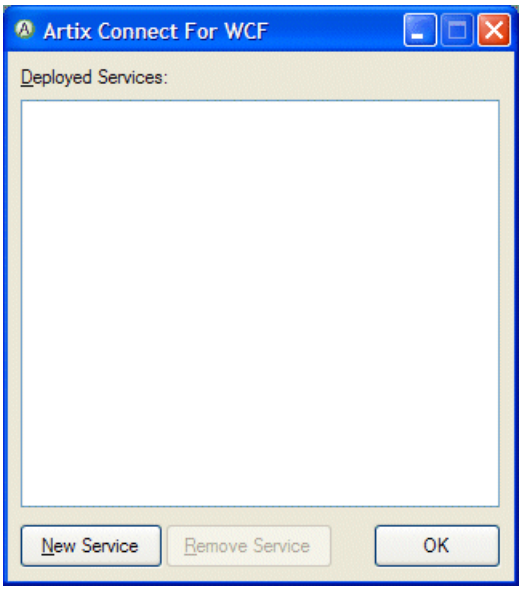
ii. Click Configure.

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

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

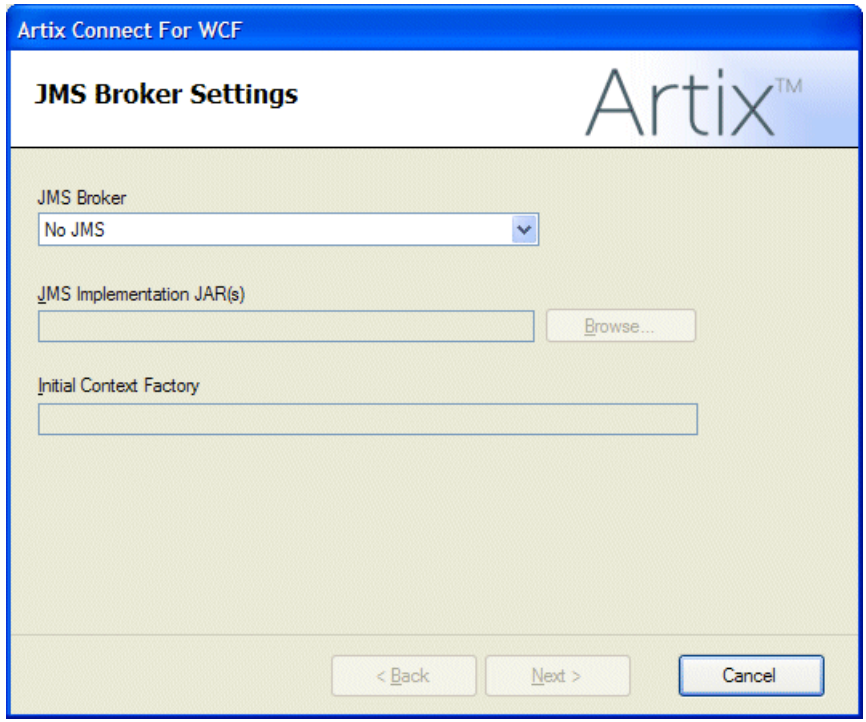
The Artix Connect for WCF wizard opens as shown in Figure 14, “Artix Connect for WCF Wizard”. The deployed services list is empty if you have not already deployed any services.

Figure 14. Artix Connect for WCF Wizard



3. In the Artix Connect for WCF wizard, click New Service.
4. In the New Service window, select the JMS radio button.
5. Click Next.
6. In the JMS Broker Settings window, shown in Figure 15, “Adding JMS Broker Settings”:

Figure 15. 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 *Installation Guide*.

iii. Click Next.

Note

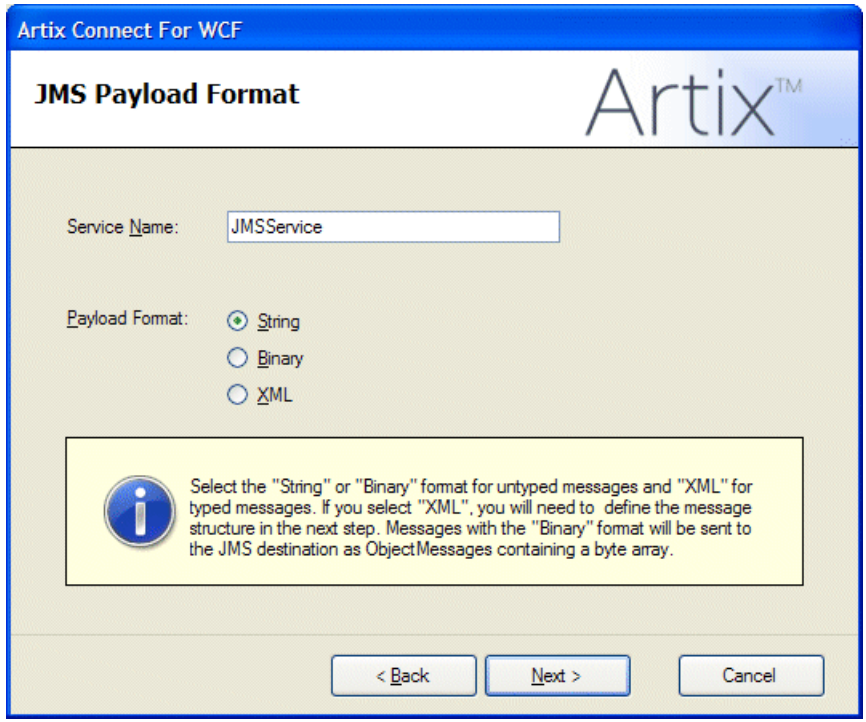
You are only asked to set JMS broker settings once. The JMS Broker Settings window does not appear when you run the Artix Connect for WCF wizard again. If you want to subsequently change the JMS broker that you are using, please use the Artix Administration tool to enter details of the new broker. For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

Selecting a payload format

The JMS Payload Format window enables you to give the service a name and to select the type of message that you are sending. 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.

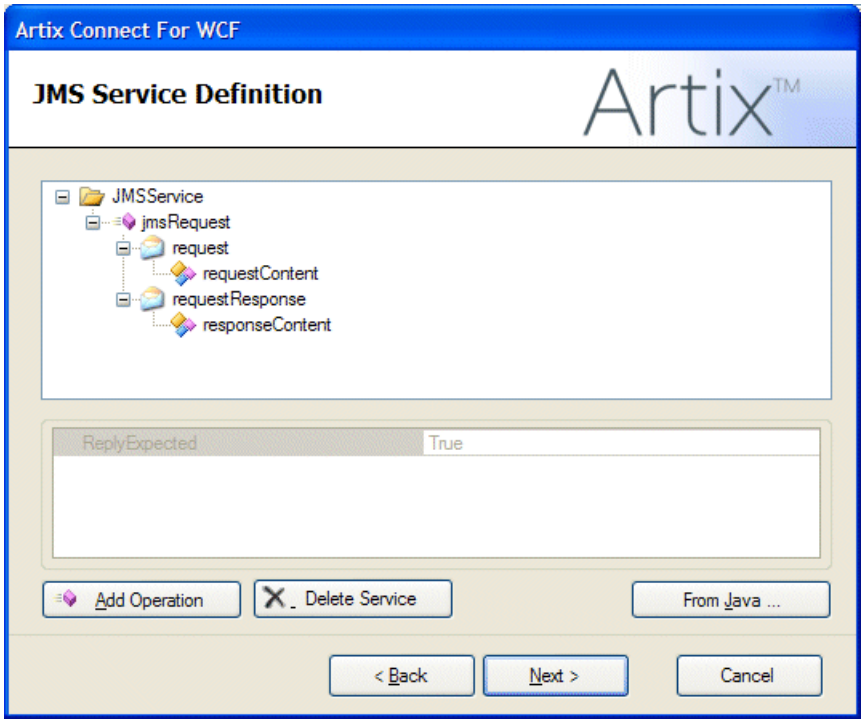
Figure 16. JMS Payload Format



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 as shown in Figure 17, “Defining XML Message”.

Figure 17. Defining XML Message



You can:

- Use the tree and the buttons below the service 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.

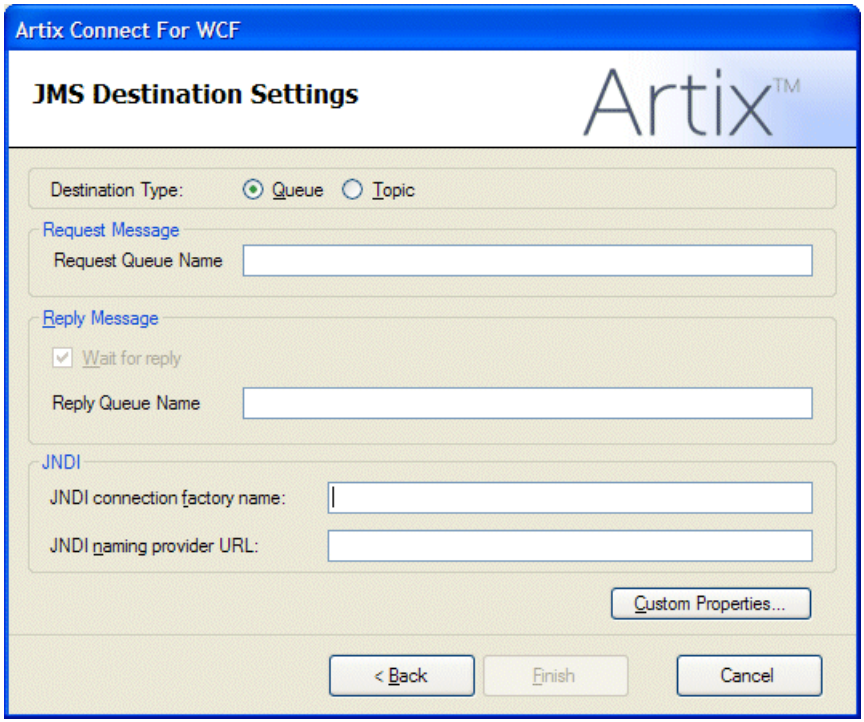
  The wizard examines the Java class and extracts the relevant interface information from it. This information is displayed in the top panel the JMS Service Definition window.

5. Click Next.

Specifying JMS destination settings

In the JMS Destination Settings window you need to set JMS destination information (see Figure 18, “JMS Destination Settings”. This information is specific to the JMS service to which you want to connect and the JMS broker that you are using.

Figure 18. JMS Destination Settings



1. Fill in the JMS Destination Settings window. The fields are described in Table 1, “JMS Destination Settings”. For example settings for each of the supported JMS brokers, see the Getting Started Guide.

Table 1. JMS Destination Settings

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

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.

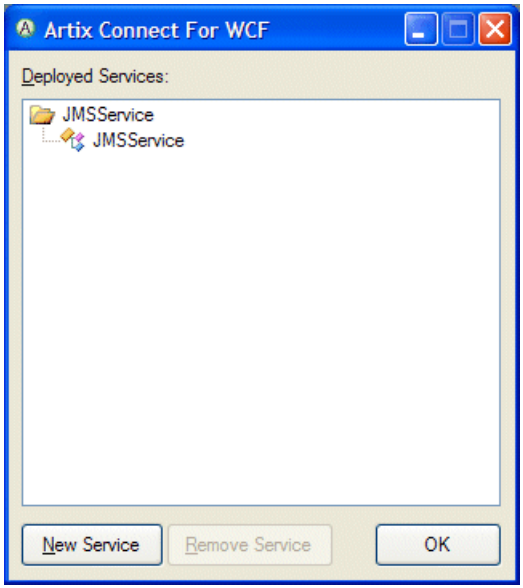
   ii. In the Custom Properties window, under Name type the name of your custom property, and under Value type the value of your custom property.

   iii. Click OK.

3. Click Finish.

   The wizard completes its tasks and the JMS service is listed under Deployed Services, as shown, for example, in Figure 19, “JMS Service Deployed”.

Figure 19. JMS Service Deployed



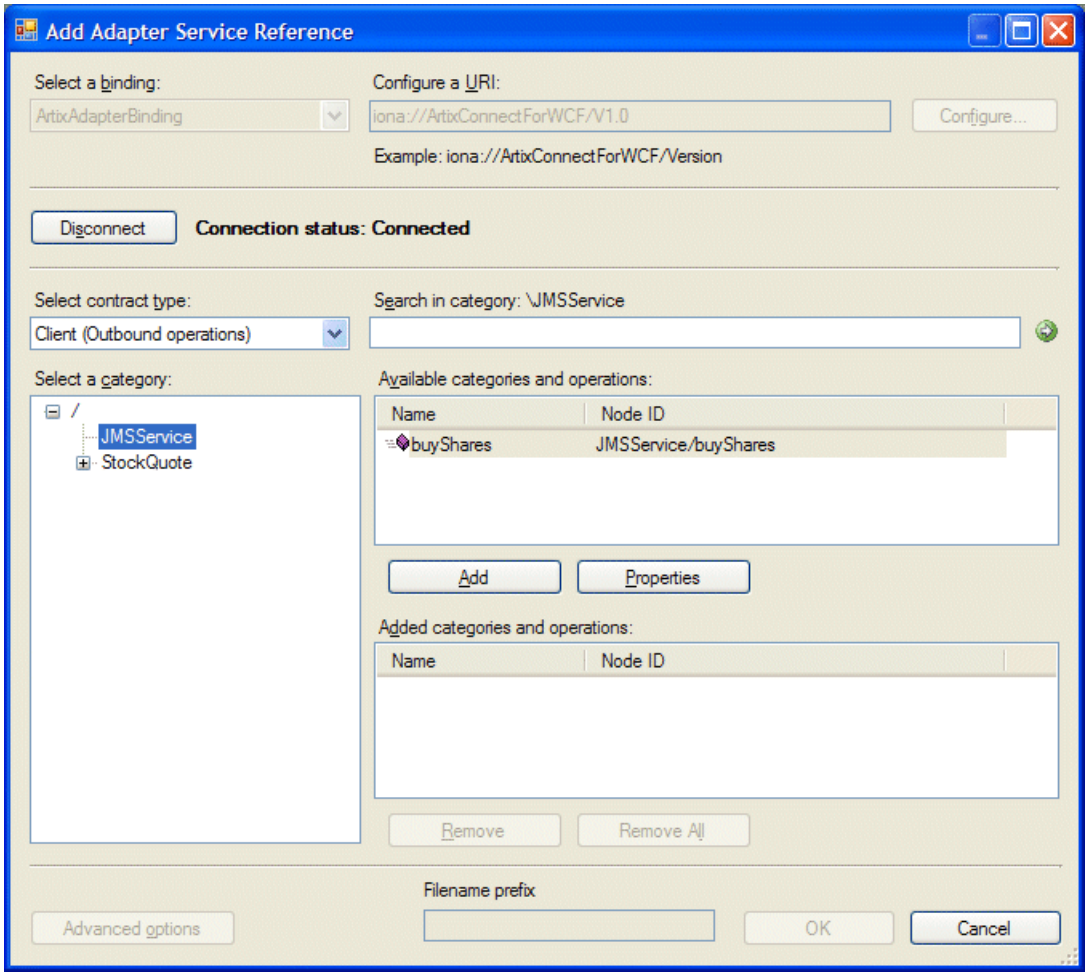
4. Click OK.

   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 service in the Select a category panel (see, for example, Figure 20, “Making JMS Operations Available to .NET Applications”. The OK button is disabled. It remains so until you specify which operations you want to use within your WCF application code.

Figure 20. Making JMS Operations Available to .NET Applications



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 service that you just deployed.

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.

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 access the JMS system. To see an example, run the tutorial described in the Getting Started Guide.

Removing a Service

To remove a service, in the Deployed Services window of the Artix Connect for WCF wizard (see Figure 19, “JMS Service Deployed”), select the service that you want to delete and click Remove Service.

Deploying Your Applications

Summary

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

Table of Contents

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

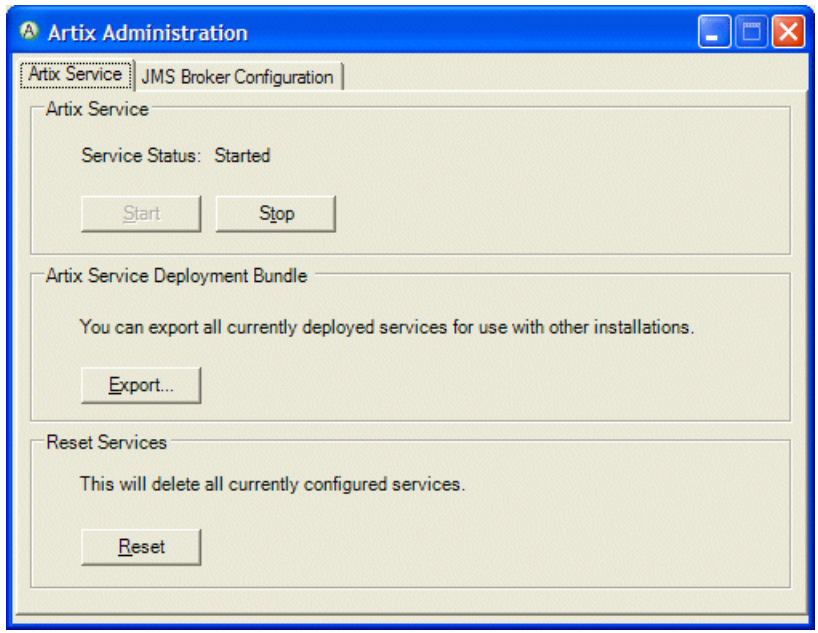
To export the applications that you have developed using Artix Connect for WCF so that they can be imported and deployed on to a runtime machine:

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 21, “Exporting Applications”.

Figure 21. 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.

Importing Your Applications

Deployment Steps: Importing

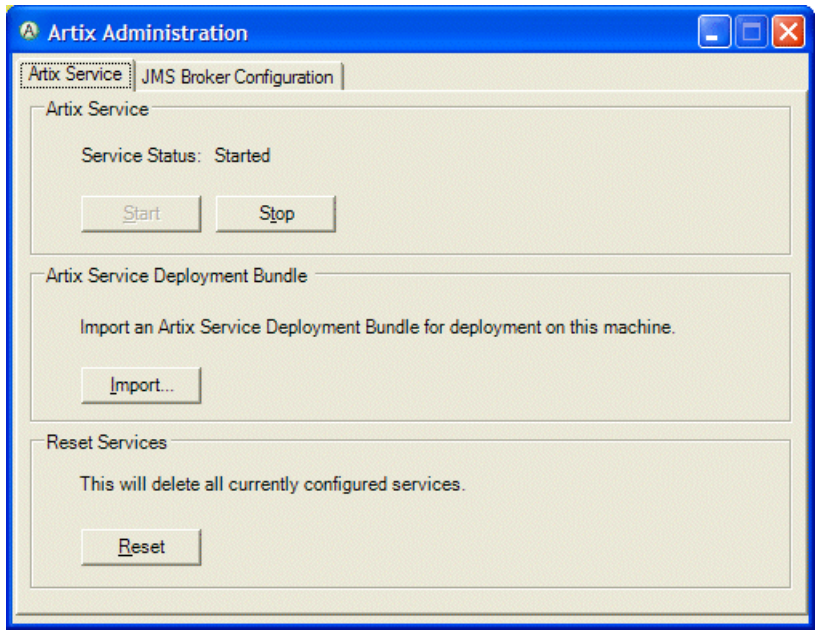
To import and deploy applications that you have developed using Artix Connect for WCF on to a runtime machine:

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 22, “Importing and Deploying Applications”.

Figure 22. 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.

Using the Artix Administration Tool

Summary

This chapter describes how to use the Artix Administration tool to stop, start and reset the Artix service. In addition, it outlines how to configure a JMS broker.

For information on how to use the Artix Administration tool to deploy your applications, see Deploying Your Applications.

Table of Contents

Stopping, Starting and Resetting the Artix Service

Steps

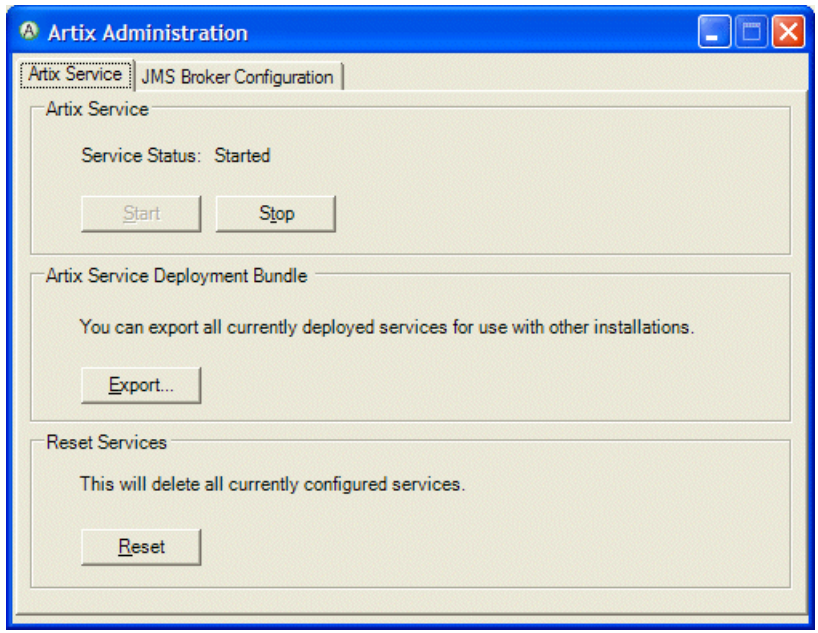
To stop, start or reset the Artix service:

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 23, “Artix Administration Tool”.

Figure 23. Artix Administration Tool



2. In the Artix Service tab, 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:

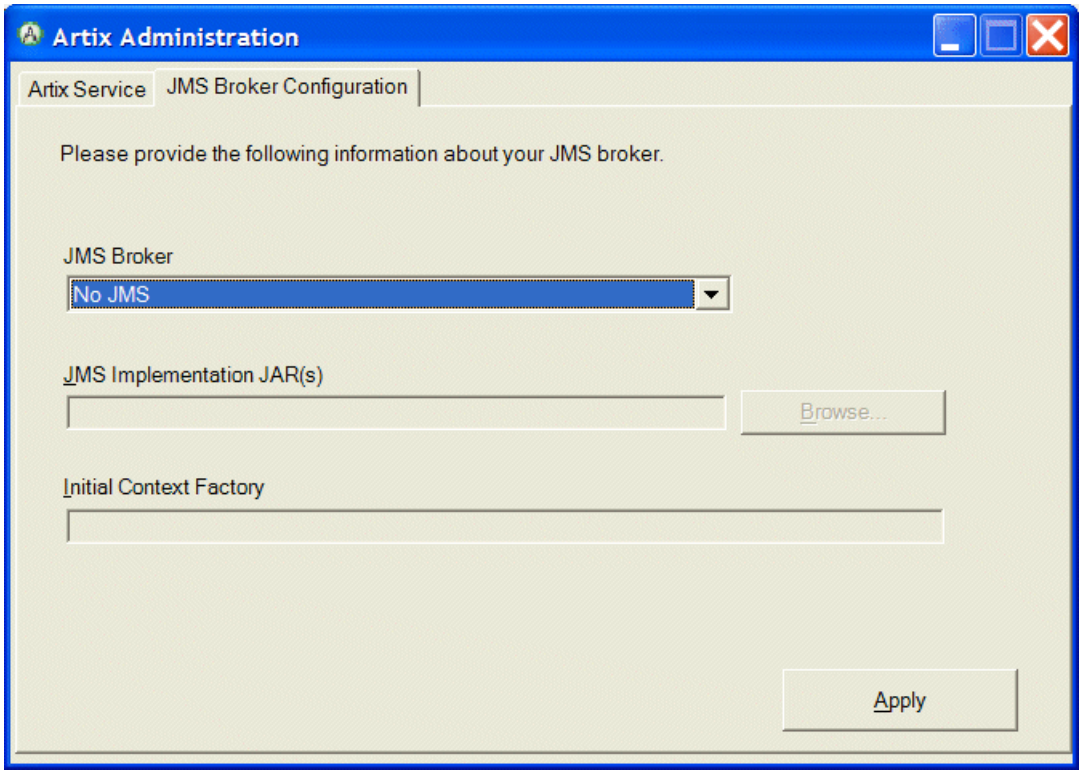
   (All) Programs | IONA | Artix Connect For WCF Beta | Artix Administration

   It appears as shown in Figure 23, “Artix Administration Tool”.

2. Select the JMS Broker Configuration tab.

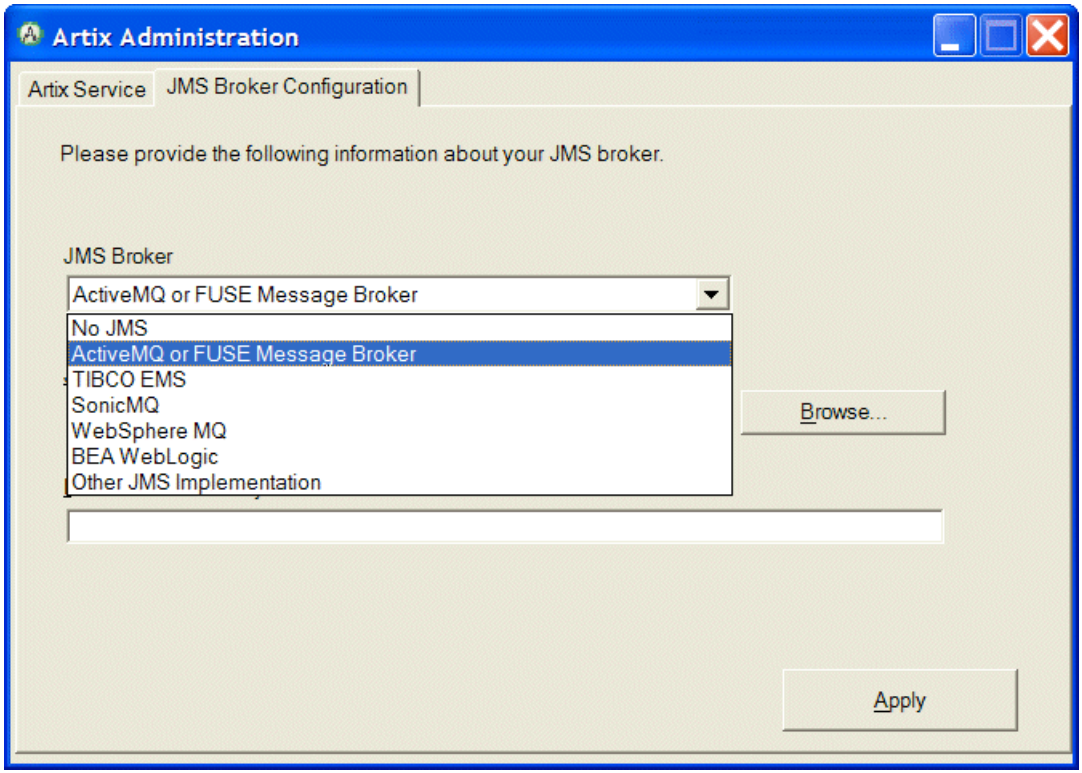
3. In the JMS Broker Configuration window, as shown in Figure 24, “Artix Administration Tool: JMS Broker Configuration”, complete the following steps:

Figure 24. Artix Administration Tool: JMS Broker Configuration



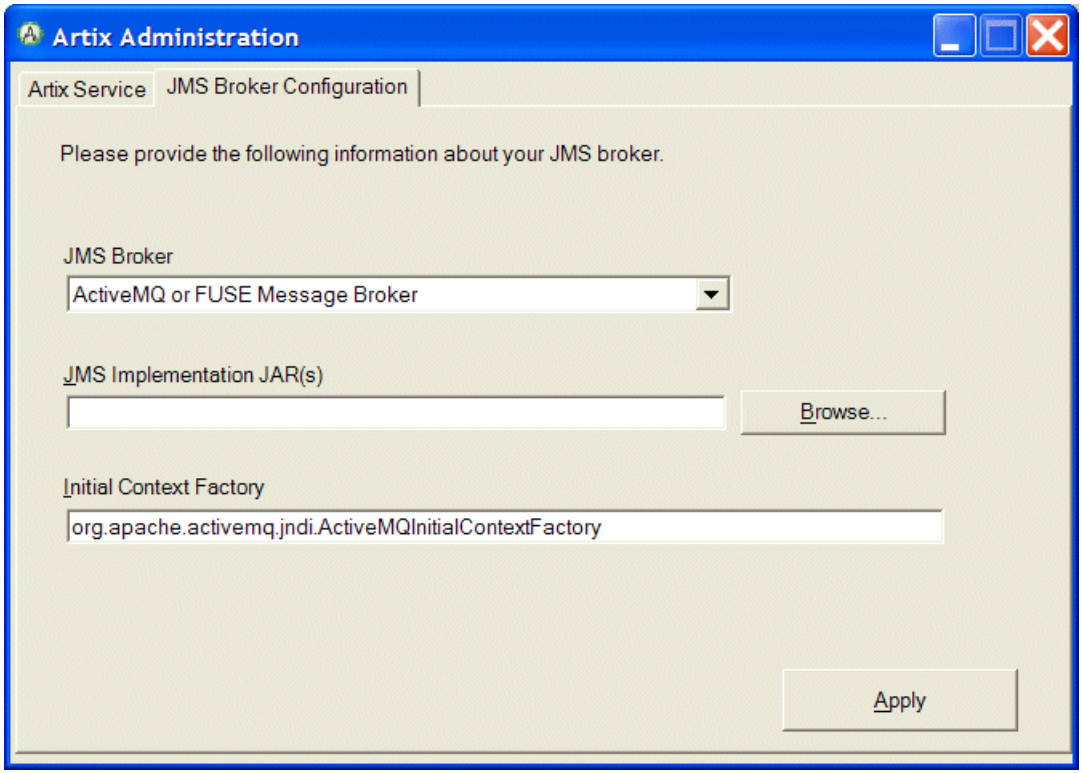
i. In the JMS Broker field, select the JMS broker that you want to use from the drop-down list (see Figure 25, “Selecting a JMS Broker”).

Figure 25. 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 26, “Setting the Initial Context Factory”.

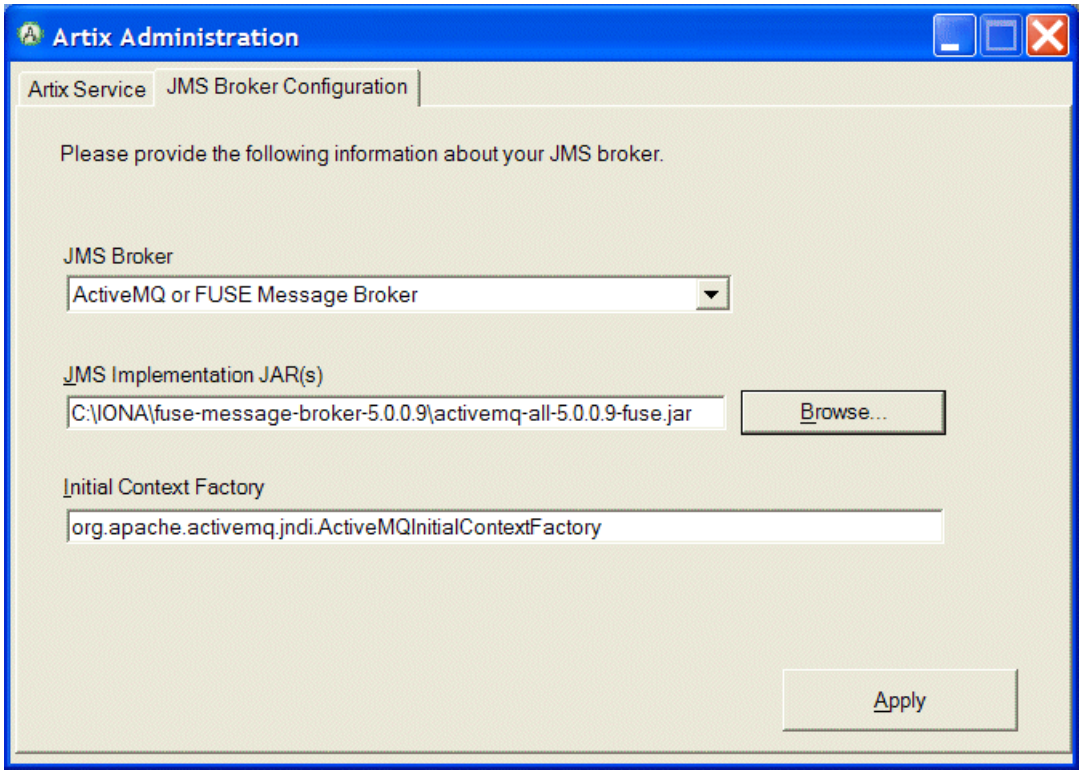
Figure 26. Setting the Initial Context Factory



ii. Select the JMS implementation JAR associated with the JMS broker that you selected. For example, in the case of FUSE Message Broker 5.0.0.9, the implementation JAR is located in `FUSEMessageBrokerInstallDir\` and is called `activemq-all-5.0.0.9-fuse.jar`, as shown in Figure 27, "Selecting the JMS Implementation JAR".

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

Figure 27. Selecting the JMS Implementation JAR



iii. Click Apply.

Artix Service Logging

Summary

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

Table of Contents

Introduction

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 6, “Default Settings for Artix Service Logging” shows the relevant section of the `artix_wcf.cfg` file.

Example 6. Default Settings for Artix Service Logging

```
❶orb_plugins = ["local_log_stream", "iiop_profile", "giop",
"iiop", "java"];
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 6, “Default Settings for Artix Service Logging” 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 2. 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 6, “Default Settings for Artix Service Logging”). You can, however, change the logging level using the syntax shown in Table 3, “Artix Logging Severity Levels Syntax”.

Table 3. 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.

Severity Level Syntax	Description
FATAL[_ERROR]	Fatal error messages.
*	All messages.

Example Logging Settings

Table 4, “Artix Logging Configuration Examples” shows some examples:

Table 4. 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 7, “Configuring Logging Output to a Text File” shows the relevant content of the default `artix_wcf.cfg` configuration file.

Example 7. 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 8. 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";
```

Index

O

S