IONA®

Artix™

Designing Artix Solutions

Version 2.0, March 2004

Making Software Work Together™

Updated: 11-Mar-2004

M 3 1 8 8

Contents

List of Figures

Preface

What is covered in this book

Designing Artix Solutions outlines how to design, develop, and deploy integration solutions with Artix using only the graphical user interface (GUI), also referred to as the Artix Designer. It also guides you through producing Web Services Description Language (WSDL), source code, and runtime configuration files for your Artix integration solution.

Note: This book does not contain information about using Artix from the command line interface - this is documented in *Designing Artix Solutions from the Command Line*.

Who should read this book

This guide is intended for all users of the Artix Designer. This guide assumes that you have a working knowledge of the middleware transports that are being used to implement the Artix system. It also assumes that you are familiar with basic software design concepts, and that you have a basic understanding of WSDL.

If you would like to know more about WSDL concepts, see the Introduction to WSDL in the *Artix Getting Started Guide*.

How to use this book

This section suggests ways to use this book according to your Artix knowledge level.

If you're new to Artix and/or WSDL

Chances are you want to do one or more of the following:

- Learn about Artix - see page 3
- Read a walkthrough of how to create a Web Service client, or server, or both - see page 157

If you find any terms you aren't familiar with, turn to the "Glossary" on page 169 for a list of Artix terms and definitions.

If you've worked with Artix before

You probably already have some WSDL knowledge, and have a clear idea of what you want to use Artix to do. In this case, you should proceed straight to the relevant section from the list provided here:

- If you are creating a new workspace, see page 19
- If you're adding or editing a collection, see page 25
- If you're adding or editing resources, see page 37
- If you're adding or editing a binding, see page 94
- If you're adding or editing a service, see page 109
- If you're adding or editing a route, see page 137
- If you're ready to deploy your collection, see page 145

If you are migrating from Artix 1.3 to Artix 2.0

We have made considerable changes to the Artix Designer in 2.0, in an effort to make it more intuitive and easier to use.

Changes include:

- A Project is now called a Workspace. For more information, see "What is a Workspace?" on page 14
- The Client, Server, and Artix nodes in the Designer Tree have been replaced by a more generic (and non-deployment specific) entity - a *Collection*. For more information, see "What is a Collection?" on page 26
- The Artix Contract still exists, but it can be created from various types of *Resources*. It is possible to have resources that are based on existing WSDL or IDL files, or data sets such as COBOL Copybooks. For more information see "What are Resources?" on page 38
- Deployment is a 3-step process - for more information, see "Deployment Explained" on page 146.

If you have Projects that were created using Artix 1.3.x, they will not be able to be opened in the 2.0 version of the user interface. You will need to re-create them.

Finding your way around the Artix library

The Artix library contains several books that provide assistance for any of the tasks you are trying to perform. The remainder of the Artix library is listed here, with a short description of each book.

If you're new to Artix

You may be interested in reading:

- *Getting Started with Artix* - this book describes basic Artix and WSDL concepts. It also helps you decide which of the other Artix books you need to use to guide you through Artix while developing your solution.
- *Artix Tutorial* - this book guides you through programming Artix applications against all of the supported transports.

To design and develop Artix solutions from the command line

You should read one or more of the following:

- *Designing Artix Solutions from the Command Line* - this book provides detailed information about the WSDL extensions used in Artix contracts, and explains the mappings between data types and Artix bindings
- *Developing Artix Applications in C++* - this book discusses the technical aspects of programming applications using the Artix C++ API
- *Developing Artix Applications in Java* - this book discusses the technical aspects of programming applications using the Artix Java API

To manage and configure your Artix solution

You should read one or more of the following:

- *Deploying and Managing Artix Solutions* - describes how to deploy Artix-enabled systems, and provides detailed examples for a number of typical use cases.
- *IONA Tivoli Integration Guide* - explains how to integrate Artix with IBM Tivoli.
- *IONA BMC Patrol Integration Guide* - explains how to integrate Artix with BMC Patrol.
- *Artix Security Guide* - provides detailed information about using the security features of Artix, and assumes a working knowledge of C++ or Java, as well as basic security concepts.

Have you got the latest version?

The latest updates to the Artix documentation can be found at http://www.iona.com/support/docs. Compare the version details provided there with the last updated date printed on the inside cover of the book you are using (under the copyright notice).

Artix online help

While using the Artix Designer you can access contextual online help, providing:

- A description of your current Artix Designer screen
- Detailed step-by-step instructions on how to perform tasks from this screen
- A comprehensive index and glossary
- A full search feature

There are two ways that you can access the Online Help:

- Click the Help button on the Artix Designer panel, or
- Select **Contents** from the Help menu

Additional resources

The IONA knowledge base (http://www.iona.com/support/knowledge_base/index.xml) contains helpful articles, written by IONA experts, about all of IONA's products.

The Artix Tech Zone (http://www.iona.com/devcenter/artix/) contains information, code samples, free downloads and support services.

The Artix Interactive Demo is available to walk you through a basic demonstration of the features included in the Artix Designer. You'll find a link to this demo under the Help menu.

The IONA update center (http://www.iona.com/support/updates/index.xml) contains the latest releases and patches for IONA products.

If you need help with this or any other IONA products, contact IONA at support@iona.com. Comments on IONA documentation can be sent to docs-support@iona.com .

Conventions used in this book

Artix Designer procedures

This guide contains procedures explaining how to perform tasks with the Artix Designer. The Artix Interface has been designed to cater for the different ways users like to work, and so there is usually more than one way to perform every task.

For example, every option listed in the main menu options (File, Edit, Contract, Tools, Help) can also be accessed as a contextual (or pop-up) menu option by right-clicking on objects within the Designer. Conversely, every contextual menu option can be found under one of the main menus.

In the interest of keeping this guide to a manageable size, only the main menu path through procedures is described.

Typographical conventions

This guide uses the following typographical conventions:

`Constant width`	Constant 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 `CORBA::Object` class. Constant width paragraphs represent code examples or information a system displays on the screen. For example: `#include <stdio.h>`
Italic	Italic words in normal text represent *emphasis* and *new terms*. 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/`*`your_name`* **Note:** Some command examples may use angle brackets to represent variable values you must supply. This is an older convention that is replaced with *italic* words or characters.

Keying conventions

This guide may use the following keying conventions:

No prompt	When a command's format is the same for multiple platforms, a prompt is not used.
`%`	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 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.
|	A vertical bar separates items in a list of choices enclosed in {} (braces) in format and syntax descriptions.

CHAPTER 1

Introduction to Artix

With the Artix Designer, you can design, develop, and deploy integration solutions that are middleware-neutral.

In this chapter

This chapter discusses the following topics:

Overview

Artix is a flexible and easy-to-use tool for integrating your existing applications across a number of different middleware platforms. Artix also makes it easy to expose your existing applications as Web services or as a service for any number of applications using other middleware transports. In addition, Artix provides a flexible programming model that allows you to create new applications that can communicate using any of the protocols that Artix supports.

Despite the flexibility and power of Artix, designing solutions using Artix is a straightforward process which requires a minimum of coding. The Artix Designer provides a full suite of wizards to guide you through the modeling of your systems, the generation of Artix components, and the deployment of your system. Artix also ships with a number of command line tools that can be used to generate Artix components. For more information about working with the command line, see *Designing Artix Solutions from the Command Line.*

Artix Designer components

When you start working with the Artix Designer, you will see the following components listed in the Designer Tree and in the menus:

- Workspace
- Shared Resource
- Collection
- Contract

Each component is documented in detail throughout this book, but following is a brief description of what they are and how they relate to each other.

Workspace

The Workspace defines your Artix solution. It contains collections and resources, and all the required deployment information to build your solution. In Artix 1.3, the Workspace was referred to as the Project.

Shared Resource

Shared Resources are WSDL contracts that are stored at a workspace level, and are included, by default, in every collection in that workspace. When creating a workspace, you are given the option of also creating shared resources, or else you can add them to the workspace later.

Resources that are not shared, and that exist only in one collection, are collection-specific. A collection-specific resource can be changed to a shared resource, and therefore added to all existing collections, if required.

Collection

A group of related WSDL contracts that can be deployed as one or more physical entities such as Java, C++, or CORBA based applications. A collection can also be deployed as a switch process.

Contracts

An Artix *contract* defines the interaction of an endpoint with the Artix bus. Contracts are written using a superset of the standard Web Service Definition Language (WSDL). Following the procedure described by W3C, IONA has extended WSDL to support the bus' advanced functionality, and to use transports and formats other than HTTP and SOAP.

In Artix, contracts can be created from a variety of resources including:

- Existing WSDL files
- Existing IDL files
- WSDL URLs
- Existing Data Sets, such as a COBOL Copybook

A contract consists of two parts:

Logical - defines the namespaces, messages, and operations that the collection exposes. This part of the contract is independent of the underlying transports and wire formats. It fully specifies the data structures and possible operation/interaction with the application interface. It is made up of the WSDL tags `<type>`, `<message>`, and `<portType>`.

Physical - defines the transports, wire formats, and routing information used to deliver messages to and from collections, over the bus. This portion of the contract also defines which messages use each of the defined transports and bindings. The physical portion of the contract is made up of the standard WSDL tags `<binding>`, `<service>`, and `<route>`. It is also the portion of the contract that may contain IONA WSDL extensions.

For more information, see "What is a Contract?" on page 39.

5 easy steps

Regardless of the complexity of your Artix project or the tools you chose to develop it, there are five basic steps in developing a solution using Artix:

1. Create an Artix workspace to define the structure your proposed solution.
2. Create an Artix collection to manage the resources that define the Artix contract.
3. Create an Artix contract to describe how you intend to integrate or expose your systems.
4. Deploy the solution.
5. Develop any application level code needed to complete the solution.

Using Artix for the first time

Getting started dialog

The first time you start the Artix Designer, you will see the Getting Started dialog, as shown in Figure 1.

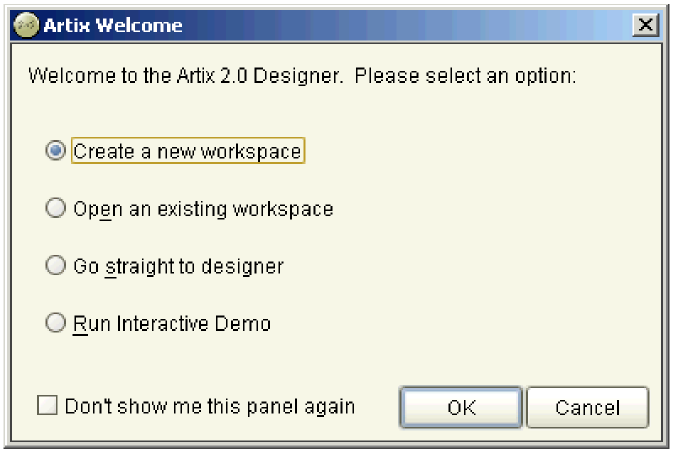

Artix Welcome
Welcome to the Artix 2.0 Designer. Please select an option:
[✓ Create a new workspace]
Open an existing workspace
Go straight to designer
Run Interactive Demo
Don't show me this panel again
OK
Cancel

Figure 1: *Getting Started dialog*

You have four options from this dialog:

- Create a new workspace - takes you to the New Workspace dialog, where you can define the structure of your Artix solution.
- Open an existing workspace - takes you to a file chooser dialog from where you can navigate to any previously created workspaces.
- Go straight to the designer - opens the Artix Designer without loading a workspace.
- Show a demo of the designer - launches an interactive demo (requires a plug-in which is available for download if necessary).

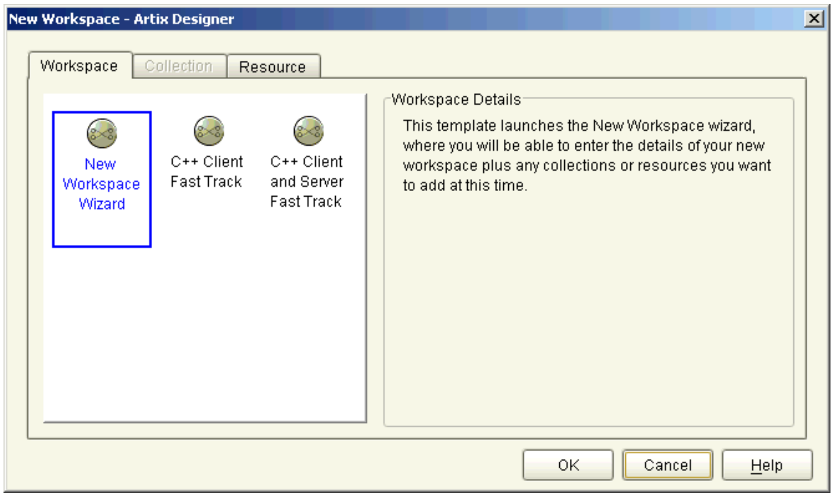
Tip: To stop the Getting Started dialog from displaying every time you start the Artix Designer, click in the check box at the bottom of the panel (Don't show me this panel again). To turn it back on, select the option in the User Preferences dialog (under the **Edit** menu).

Creating an Artix workspace

The Artix workspace defines the structure of your proposed solution, and determines what is contained in the Artix Designer Tree.

There are two ways to create a workspace:

- Follow the New Workspace wizard (from the New Workspace dialog, as shown in Figure 2) to guide you through the process - recommended for first-time users of Artix.
- Select one of the Workspace Templates provided in the New Workspace dialog to create one of the common workspaces.



This template launches the New Workspace wizard, where you will be able to enter the details of your new workspace plus any collections or resources you want to add at this time.

Figure 2: *New Workspace dialog*

Tip: To access the New Workspace dialog, select **File | New | Workspace** from the menu bar, or click the New Workspace icon in the toolbar.

Shared resources

When you have created your workspace, you have the option of adding resources at the workspace level that can be applied to every collection contained in your workspace. In Artix, these are called "Shared Resources".

For more information on workspaces and shared resources, see "Creating an Artix Workspace" on page 13.

Adding collections and resources

A *collection* is a group of *resources* that can be deployed one or more times to meet your solution requirements. As such, it defines the Artix *contract*. This contract models the services you want to integrate.

While you can only have one workspace at a time, you can have as many collections as you like. They can comprise shared resources, collection-specific resources, or a mix of both.

When you create a workspace using the New Workspace wizard you are given the opportunity to create a collection, but you can also add collections to workspaces by selecting **File | New | Collection** from the menu bar.

For more information about Artix collections, see "Working with Artix Collections" on page 25.

Collection-specific resources

After you have created a collection you can add *collection-specific resources.* Resources can be created from existing WSDL files, or from WSDL generated from IDL files. They can also include contracts generated from data sets such as COBOL Copybooks, or contracts created from scratch using the Contract Editor.

Regardless of your mix of resources, the process of creating the Artix contract involves creating logical descriptions of the data and the operations you want the services to share, and mapping them to the physical payload formats and transports used by the services to expose themselves to the network. Artix uses the industry standard Web Services Description Language (WSDL) to model services.

For more information about resources, see "Working with Artix Resources" on page 37.

Deploying your solution

Deploying with Artix is a three-step process:

- Create the **Deployment Profile** to define machine-level information that you can use for one or more of your solutions
- Create the **Deployment Bundle** to define the characteristics of the collection you are deploying, including the type of deployment (client, server, or switch), configuration information, and environment scripts
- Deploy the solution - once your deployment profile and bundle are in place, actual deployment is performed using the **Run Deployer** dialog

For more information, see "Deploying Collections" on page 145.

For a detailed discussion of Artix configuration, see *Deploying and Managing Artix Solutions*.

For a detailed description of generating Artix stubs and skeletons, see *Developing Artix Applications with C++.*

Develop application code

Unless your services share identical interfaces, you will need to develop some application code. Artix can only map between services that share a common interface. Typically, you can make the required changes to only one side of the services you are integrating and you can write the application code using a familiar programming paradigm. For example, if you are a CORBA developer integrating a CORBA system with a Tuxedo application, Artix will generate the IDL representing the interface used in the service integration. You can then implement the interface using CORBA.

If you are developing new applications using Artix, you will have to write the application logic from scratch using the stubs and skeletons generated by Artix. For a detailed discussion of developing applications using Artix, see one of the following:

- *Developing Artix Applications in C++*
- *Developing Artix Applications in Java*

WSDL Basics

Web Services Description Language (WSDL) is an XML document format used to describe services offered over the Web. WSDL is standardized by the World Wide Web Consortium (W3C) and is currently at revision 1.1. You can find the standard on the W3C web site, www.w3.org.

Elements of a WSDL document

A WSDL document is made up of the following elements, which you will see represented throughout the Artix Designer. You can use the designer to create and edit these elements:

- `<types>` – the definition of complex data types based on in-line type descriptions and/or external definitions such as those in an XML Schema
- `<message>` – the abstract definition of the data being communicated
- `<operation>`– the abstract description of an action
- `<portType>` – the set of operations representing an absract endpoint
- `<binding>` – the concrete data format specification for a port type
- `<port>` – the endpoint defined by a binding and a physical address
- `<service>` – a set of ports

Example WSDL file

On the following pages is an example of a WSDL file. It is the HelloWorld WSDL used in many of the demos shipped with the Artix product

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="HelloWorld"
    targetNamespace="http://www.iona.com/hello_world_soap_http"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:http-conf="http://schemas.iona.com/transports/http/
     configuration"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:tns="http://www.iona.com/hello_world_soap_http"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

Note the types, messages, port types, and bindings defined in this section.

```
<types>
        <schema
   targetNamespace="http://www.iona.com/hello_world_soap_http"
            xmlns="http://www.w3.org/2001/XMLSchema"
            xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
            <element name="responseType" type="xsd:string"/>
            <element name="requestType" type="xsd:string"/>
        </schema>
    </types>
 <message name="sayHiRequest"/>
    <message name="sayHiResponse">
        <part element="tns:responseType" name="theResponse"/>
    </message>
    <message name="greetMeRequest">
        <part element="tns:requestType" name="me"/>
    </message>
    <message name="greetMeResponse">
        <part element="tns:responseType" name="theResponse"/>
    </message>
 <portType name="Greeter">
    <operation name="sayHi">
        <input message="tns:sayHiRequest" name="sayHiRequest"/>
        <output message="tns:sayHiResponse"
           name="sayHiResponse"/>
    </operation>
    <operation name="greetMe">
       <input message="tns:greetMeRequest"
           name="greetMeRequest"/>
       <output message="tns:greetMeResponse"
           name="greetMeResponse"/>
    </operation>
 </portType>
 <binding name="Greeter_SOAPBinding" type="tns:Greeter">
    <soap:binding style="document"
        transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="sayHi">
        <soap:operation soapAction="" style="document"/>
        <input name="sayHiRequest">
            <soap:body use="literal"/>
        </input>
        <output name="sayHiResponse">
            <soap:body use="literal"/>
        </output>
    </operation>
```

Note the service defined in this section.

```
      <operation name="greetMe">
          <soap:operation soapAction="" style="document"/>
          <input name="greetMeRequest">
              <soap:body use="literal"/>
          </input>
          <output name="greetMeResponse">
              <soap:body use="literal"/>
          </output>
      </operation>
  </binding>
  <service name="SOAPService">
      <port binding="tns:Greeter_SOAPBinding" name="SoapPort">
          <soap:address location="http://localhost:9000"/>
          <http-conf:client/>
          <http-conf:server/>
      </port>
  </service>
</definitions>
```

CHAPTER 2

Creating an Artix Workspace

The Artix Designer provides a canvas within which you can design Artix solutions. The packaging mechanism for these solutions is the workspace.

In this chapter

This chapter discusses the following topics:

What is a Workspace?

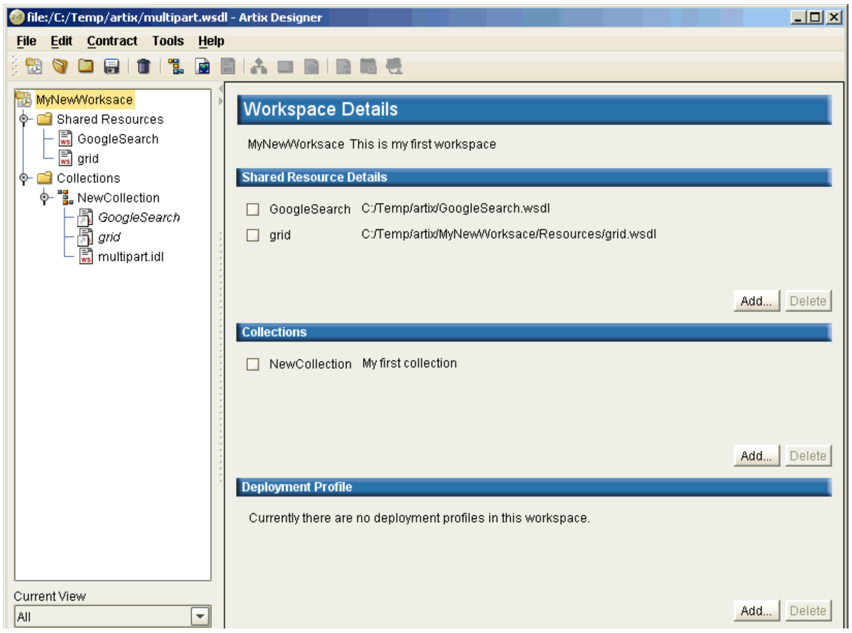
Overview

The Artix Workspace defines your Artix solution. It is the first thing you need to create, and all of the solution's components are included within it.

Workspaces contain *collections* and *resources*. While you can only have one workspace open at a time, you can have as many collections and resources within that workspace as you like.

A collection is a group of resources that can be deployed as one or more systems, for example a client, a server, or a switch. A resource is a WSDL file that, either by itself or with other resources, defines the Artix Contract.

Resources can be stored at the workspace level and applied to one or more collections (shared resources), or can be stored at the collection level and apply only to that collection. The Workspace Details panel shows you the contents of your workspace, as shown in Figure 3.



MyNewWorksace This is my first workspace

GoogleSearch	C:/Temp/artix/GoogleSearch.wsdl
grid	C:/Temp/artix/MyNewWorksace/Resources/grid.wsdl

NewCollection	My first collection

Currently there are no deployment profiles in this workspace.

Figure 3: *Workspace Details panel*

The Workspace Details panel, as shown in Figure 3 on page 14, provides another view of your workspace, besides the Designer Tree view. It lists the collections and resources contained in your workspace, and also provides Add and Delete functions.

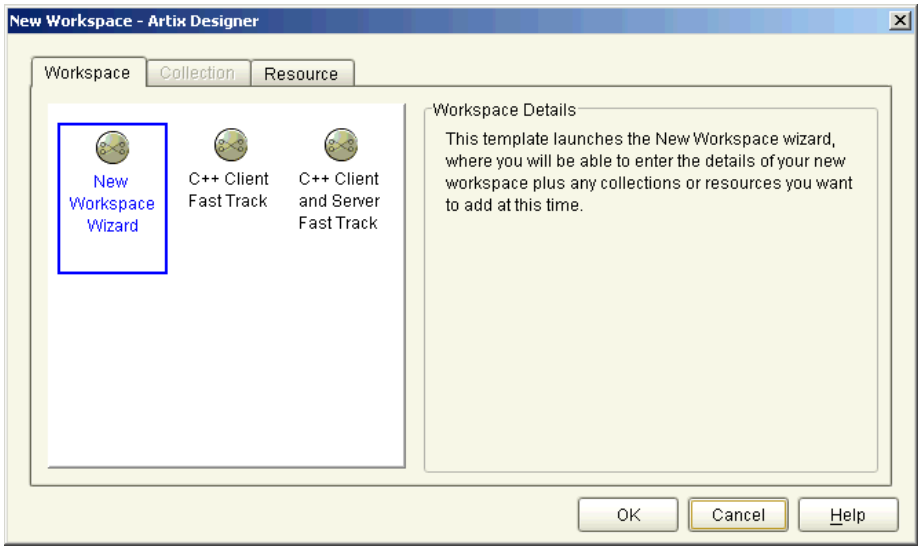
Deployment Profiles for the workspace are also listed on the Workspace Details panel. A profile is needed for each different operating system that will host your Artix deployment, such as Windows or Unix.

For more information about working with Artix Deployment, including Deployment Profiles, see "Deployment Explained" on page 146.

Creating a workspace

There are two ways to create a new workspace:

- Follow the New Workspace wizard (from the New Workspace dialog, as shown in Figure 4) - recommended for first time users of Artix. - see page 19 for more information.
- Select one of the Workspace Templates provided in the New Workspace dialog to have Artix assume all of the necessary defaults to get your workspace up and running quickly. See page 23 for details.



This template launches the New Workspace wizard, where you will be able to enter the details of your new workspace plus any collections or resources you want to add at this time.

Figure 4: *New Workspace dialog*

Shared resources

Shared resources are resources that are stored at the workspace level, and by default are automatically included in every collection you create within the workspace. All instances of this resource are linked, however - they are not individual copies of the resource. If you edit a shared resource, you are actually editing every instance of that resource.

When you create a new workspace, you are given the option of adding shared resources via the Shared Resources panel, as shown in Figure 5.

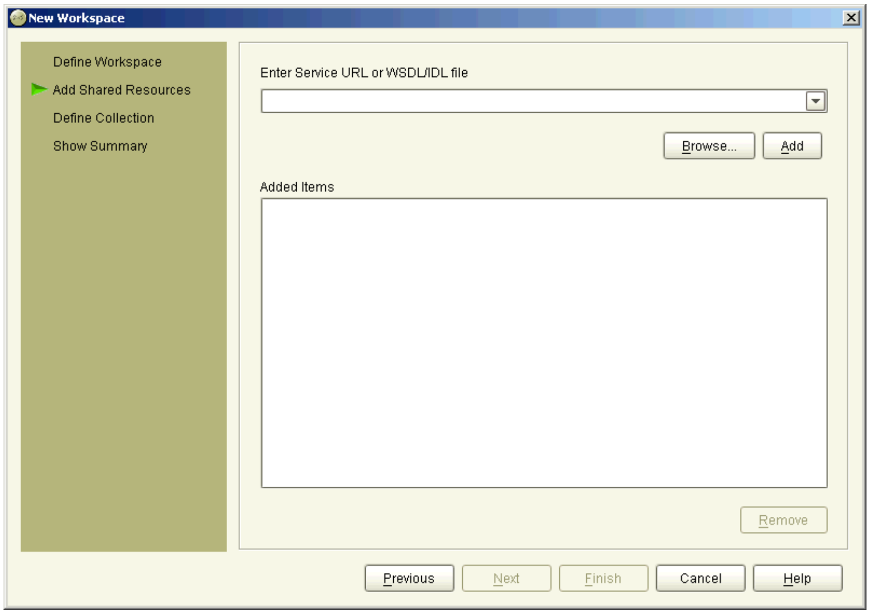


Figure 5: *New Workspace wizard - Shared Resources panel*

You can also add shared resources at other times from the Workspace Details panel by clicking the **Add** button under the Shared Resources Details list.

A shared resource is represented in two ways in the Designer Tree. The *original* version of the resource is listed under the Shared Resources folder, and the *reference* to the Shared Resource in each Collection is shown with the name of the resource italicized, and its icon having a dimmed shortcut arrow, as shown in Figure 6. An **X** icon next to the resource name indicates invalid WSDL.

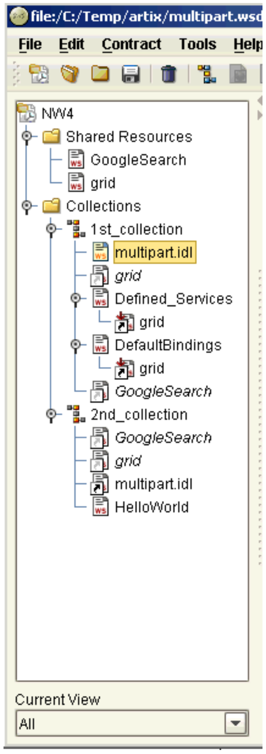


Figure 6: *Designer Tree Showing Collections and Shared Resources*

Collection-specific resources

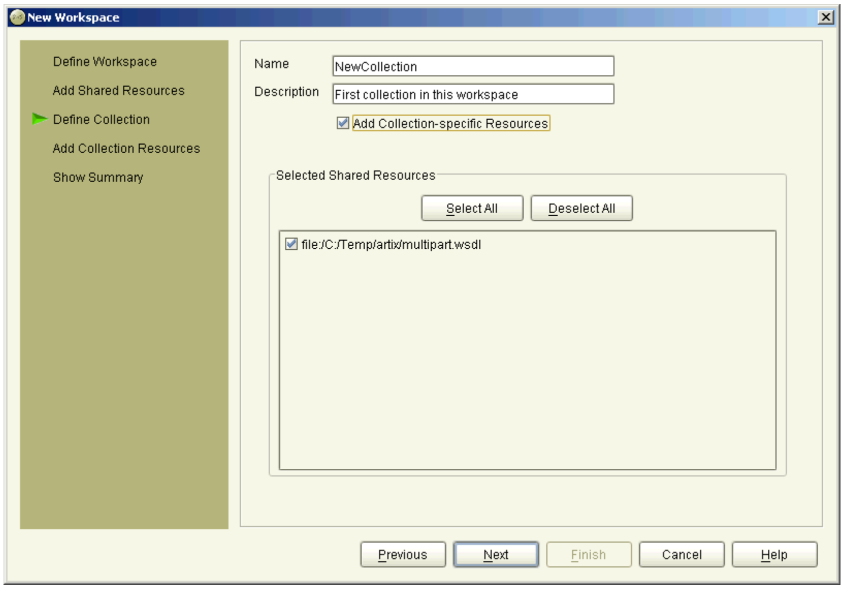
In contrast, you can also create resources that are collection-specific. That is, they apply only to the collection within which they are created and are not also included in other collections within the workspace. These are indicated in Figure 6 by the resource names within collections that are not italicized, and are created by selecting a the Collection name and selecting **File | New | Resource**. See "Working with Artix Resources" on page 37.

Collections

A collection is a group of related resources within your workspace. It can be deployed as one or more systems, such as a client, a server, a switch, or any combination of all three.

When you create a workspace, you are given the opportunity to create a collection via the Define Collection panel, as shown in Figure 7.

Otherwise you can add them to your workspace from the Workspace Details panel by clicking on the **Add** button under the Collections list.



Name: NewCollection
Description: First collection in this workspace
☑ Add Collection-specific Resources

☑ file:/C:/Temp/artix/multipart.wsdl

Figure 7: *New Workspace wizard - Collection panel*

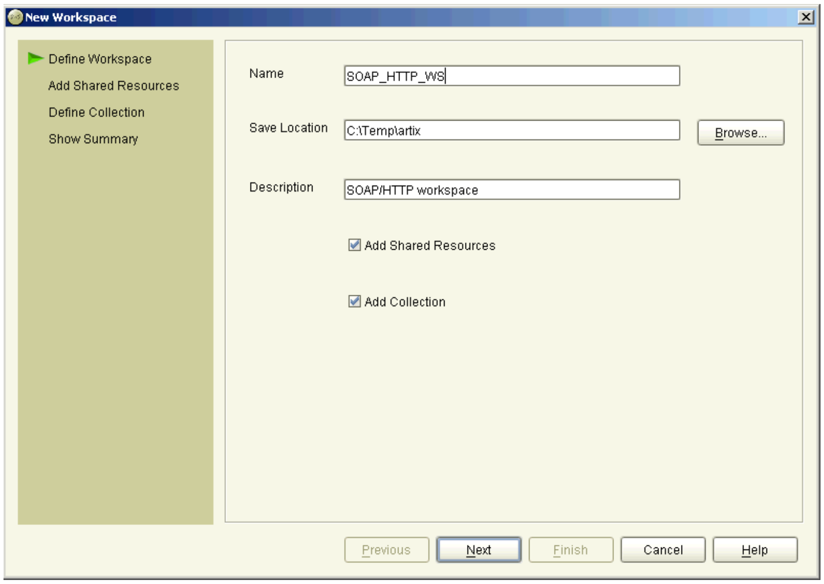
Collections contain resources that together define the Artix contract. These resources can be based on one or more items, including URLs, and WSDL or IDL files. If an IDL file is added to a collection, it is converted to WSDL and this WSDL is what is actually listed on the Designer Tree.

For more information see "Working with Artix Collections" on page 25.

Creating a Workspace using a Wizard

To add a workspace using the New Workspace wizard:

1. From the New Workspace dialog, select the **New Workspace wizard** icon to display the New Workspace wizard, as shown in Figure 8.



Name SOAP_HTTP_WS

Save Location C:\Temp\artix

Description SOAP/HTTP workspace

☑ Add Shared Resources

☑ Add Collection

Figure 8: *New Workspace Wizard*

2. Enter a name for the workspace, or accept the default provided.
3. Select the location where you would like to save your workspace, or accept the default provided.

   Tip: To define a new default save location for all future workspaces, go to the User Preferences dialog (under the Edit menu).
4. Add a description for this workspace in the field provided.
5. Select the **Add Shared Resources** check box if you want to add resources to this workspace that will be shared between all the collections in the workspace. This step is optional.

Selecting this option will add an extra panel to the wizard for you to enter the shared resource details.

6. Select the **Add Collection** check box if you want to add a collection to this workspace now. Note that this is optional - you can always add a collection later if you don't want to add one now.

   Selecting this option will add an extra panel to the wizard for you to enter the collection details.

7. Click **Next** to display one of the following panels, depending on which check boxes you selected on the first panel:

   ♦ If you checked the Add Shared Resources option, the Shared Resources panel is displayed, as shown in Figure 9. Continue with **step 8.**

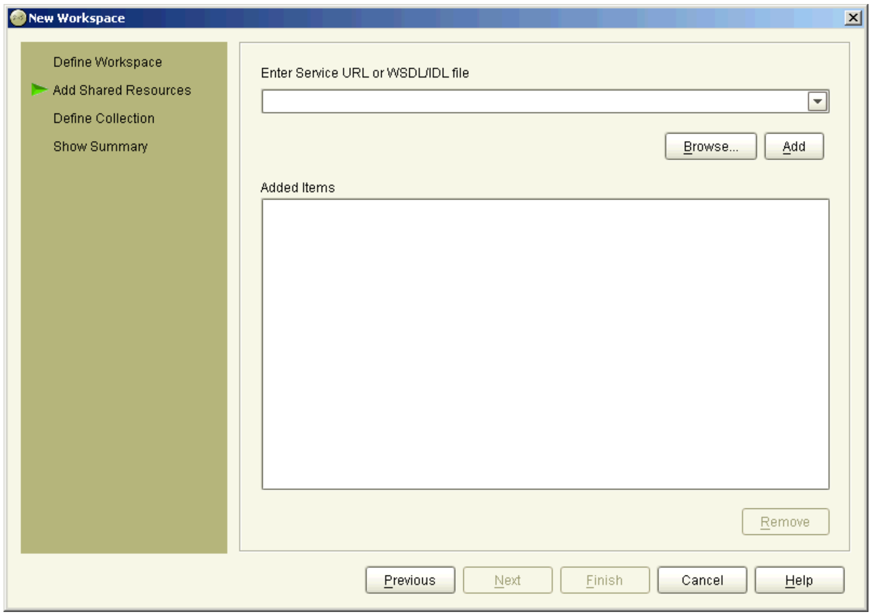


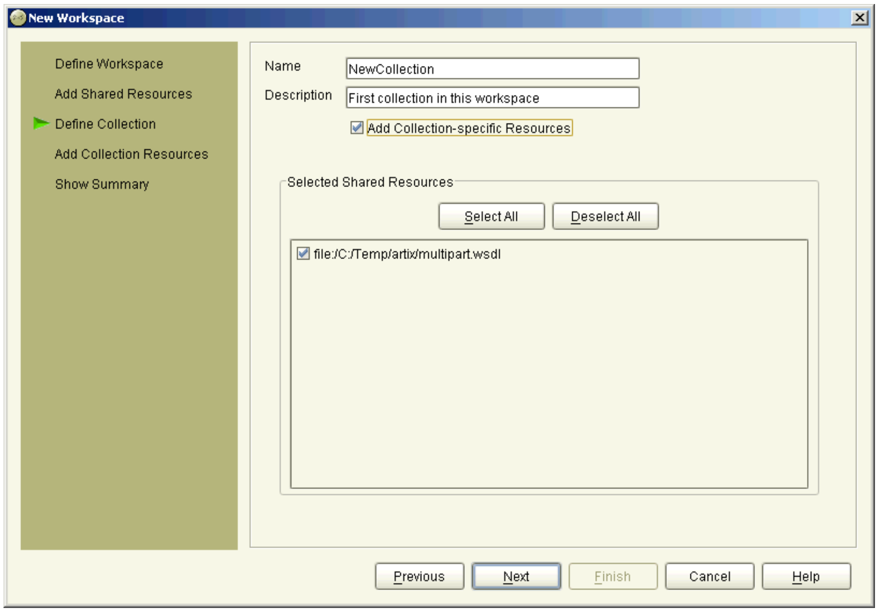
Figure 9: *New Workspace wizard - Shared Resources panel*

- If you did not check the Add Resources option but did check the Add Collection option, the Define Collection panel is displayed, as shown in Figure 10. Continue with **step 10**.
- If you did not check either of the options on the first panel, the Summary panel is displayed as shown in Figure 11 on page 22. Continue with **step 14**.

8. Type the location of either a WSDL file or an IDL file in the Enter Service URL or WSDL/IDL file field, or click **Browse** to navigate to the file you would like to use.

   When you have selected a file to use, click **Add** to list it in the Added Items list.

9. Repeat **step 8** as many times as you like to continue adding resources to the list, then click **Next** to display Define Collection panel as shown in Figure 10. If you did not choose to Add a Collection, go to **step 14**.

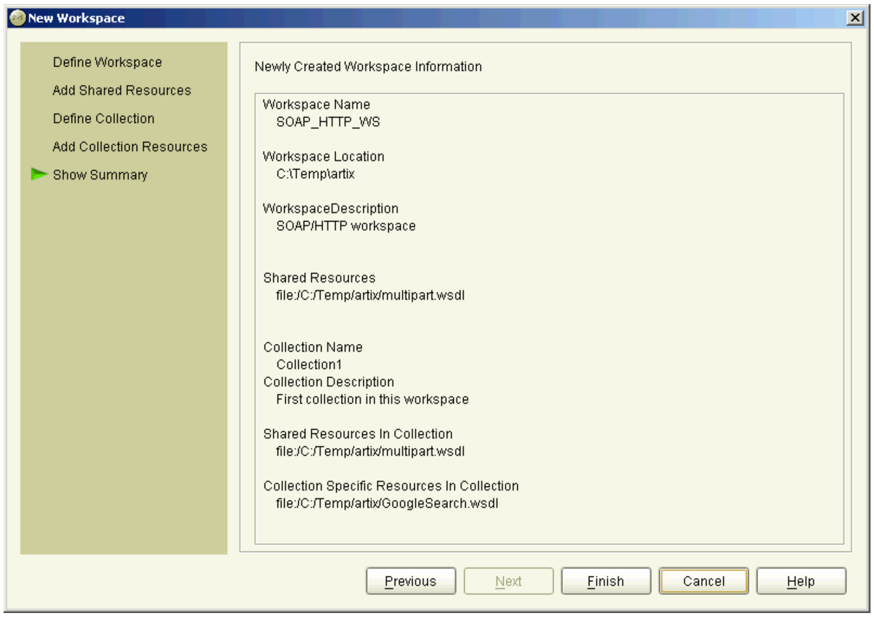


Name: NewCollection
Description: First collection in this workspace
☑ Add Collection-specific Resources

Selected Shared Resources
☑ file:/C:/Temp/artix/multipart.wsdl

Figure 10: *New Workspace wizard - Define Collection panel*

10. Enter a name for the new collection, or accept the default provided.
11. Enter a description for the new collection in the Description field.
12. By default, all shared resources you added to this workspace on the previous panel are selected to be added to this collection. If there are any resources you do not want added, click on their check box to deselect them.
13. Click **Next** to display the Summary panel, as shown in Figure 11. This panel lists everything you just specified in the wizard.



Newly Created Workspace Information

Workspace Name
SOAP_HTTP_WS

Workspace Location
C:\Temp\artix

WorkspaceDescription
SOAP/HTTP workspace

Shared Resources
file:/C:/Temp/artix/multipart.wsdl

Collection Name
Collection1
Collection Description
First collection in this workspace

Shared Resources In Collection
file:/C:/Temp/artix/multipart.wsdl

Collection Specific Resources In Collection
file:/C:/Temp/artix/GoogleSearch.wsdl

Figure 11: *New Workspace wizard - Summary panel.*

14. Click **Finish** to close the wizard and display the Artix Designer, where the Designer Tree displays your newly created workspace.

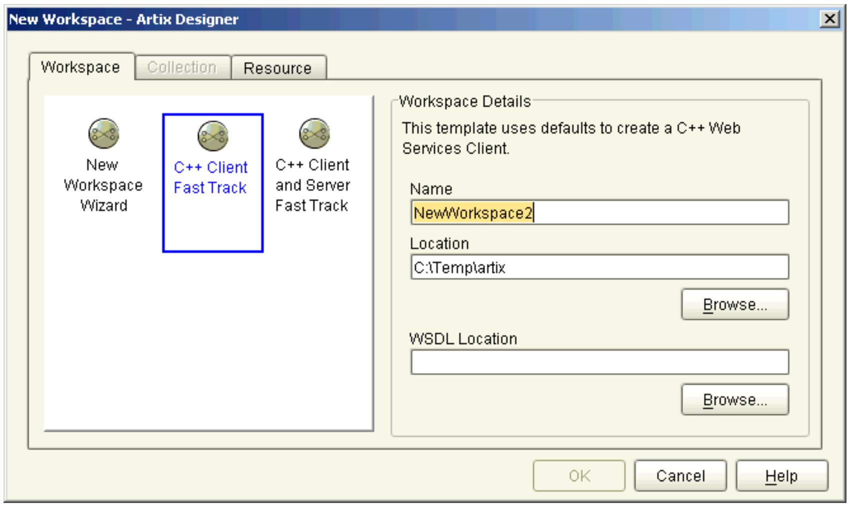
Creating a Workspace using a Template

To add a workspace using a Fast Track template:

1. From the New Workspace dialog, select one of the templates listed to create a workspace for that type of Artix deployment.

   As shown in Figure 12, the workspace templates you can choose from include:

   - A C++ Client
   - A C++ Server and Client



This template uses defaults to create a C++ Web Services Client.

Name: NewWorkspace2

Location: C:\Temp\artix

WSDL Location:

Figure 12: *New Workspace dialog showing Fast Track Template options*

2. Enter a name and save location for your workspace, or accept the defaults provided. Click **Browse** to navigate to a different save location if you wish.
3. Enter the file name or URL for your WSDL file in the field provided, or click **Browse** to navigate to a suitable file.

4. Click **OK** to display the Artix Designer with your new workspace loaded into the Designer Tree, as shown in Figure 13.

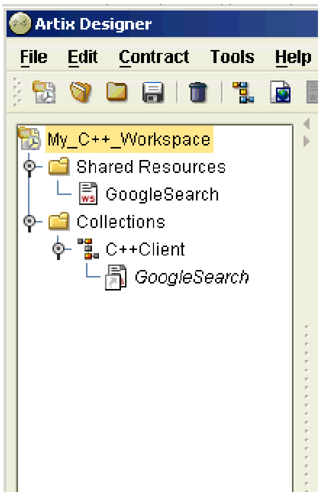


Figure 13: *Designer showing new C++ Client fast Track Workspace*

Behind the scenes

While creating your new *Fast Track* workspace, Artix has automatically performed the following tasks:

- Created a workspace directory and file in the save location you specified
- Imported your WSDL file and added it to the workspace file
- Depending on which system you decided to create, it has created a local deployment profile including configuration options
- Created a deployment bundle containing all required code and a makefile for your target service.

You could now deploy the client by selecting **Tools | Run Deployer**, which would generate all files required. **Note** than some hand editing of the implementation file will be required. For help with this, see *Developing Artix Applications in C++* or *Developing Artix Applications in Java*, depending on which type of code you're working with.

Working with Artix Collections

A Collection is a group of related WSDL contracts that can be deployed as one or more physical entities such as Java, C++, or CORBA based applications. It can also be deployed as a switch process.

In this chapter

This chapter discusses the following topics:

What is a Collection?

Overview

A collection is a group of related resources that create the Web Service definition. Resources are WSDL *contracts* that can be created by importing WSDL files or by importing IDL files which are automatically converted into WSDL by Artix. A collection may contain one or more WSDL contracts.

At deployment time, a collection can be generated into physical entities such as Java, C++, or CORBA based applications. Contracts can also be based on data sets, such a COBOL Copybooks.

Collections are listed in the Designer Tree, as are the resources belonging to that collection, as shown in Figure 14.

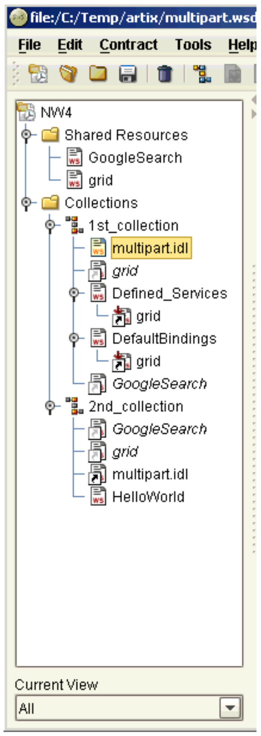
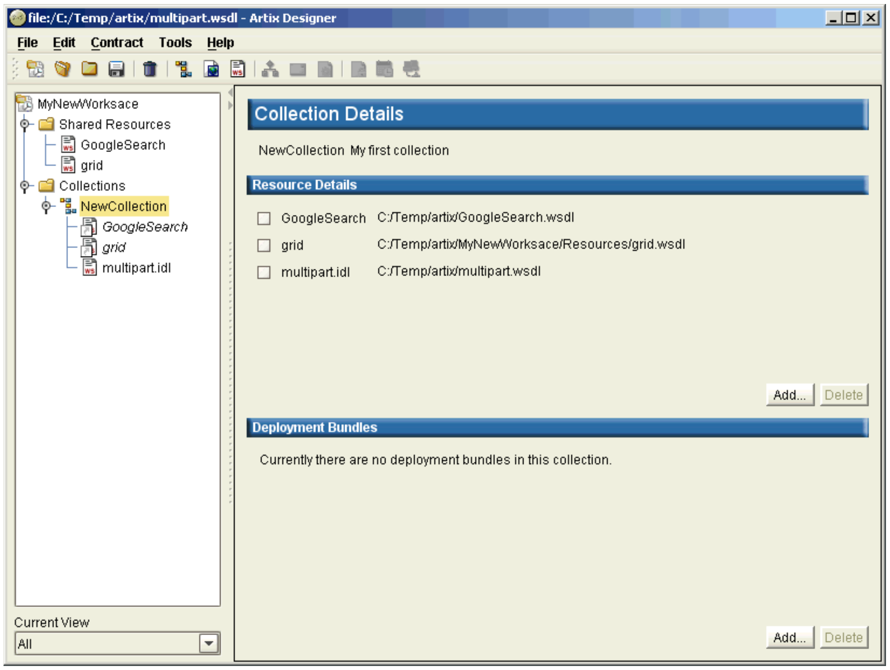


Figure 14: *Designer Tree showing Collections and Resources*

If you select a collection in the Designer Tree, the details for that collection are shown in the content panel on the right, as shown in Figure 15. In this panel you can view information about the resources contained in that collection, and about any Deployment Bundles that have been created for that collection.



NewCollection My first collection

☐	GoogleSearch	C:/Temp/artix/GoogleSearch.wsdl
☐	grid	C:/Temp/artix/MyNewWorksace/Resources/grid.wsdl
☐	multipart.idl	C:/Temp/artix/multipart.wsdl

Currently there are no deployment bundles in this collection.

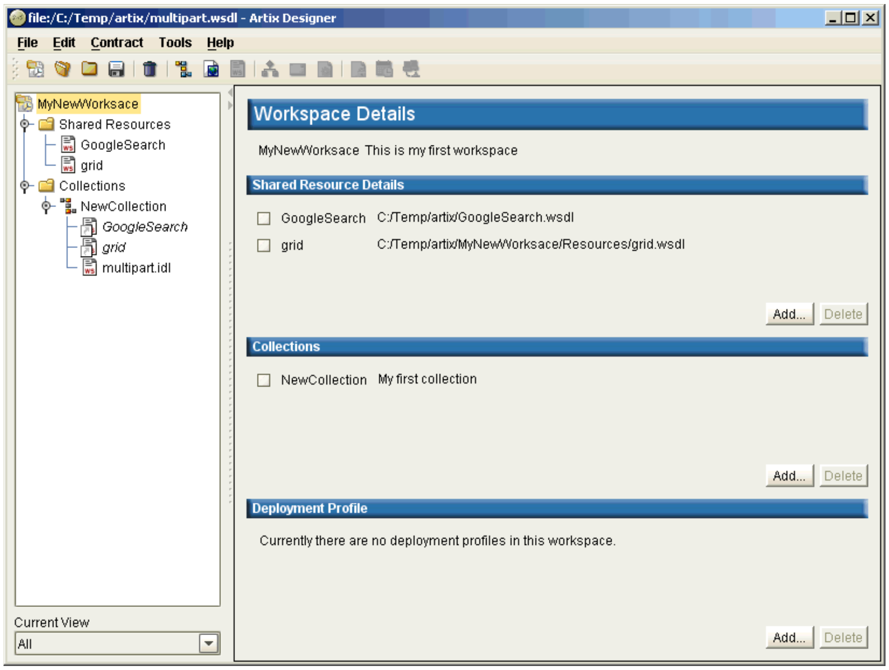
Figure 15: *Collection Details panel*

Creating a Collection

Overview

When you create a workspace using the New Workspace wizard (see page 19), you are given the option of creating a collection. Even though you are only given the option of creating one collection in this wizard, you can actually have as many collections in your workspace as you like.

Adding new collections is easy. You can click on the **Add** button under the Collections list on the Workspace Details panel (Figure 16), or you can select **File | New | Collection** from the menu bar.



Workspace Details

MyNewWorksace This is my first workspace

Shared Resource Details

☐ GoogleSearch C:/Temp/artix/GoogleSearch.wsdl

☐ grid C:/Temp/artix/MyNewWorksace/Resources/grid.wsdl

Collections

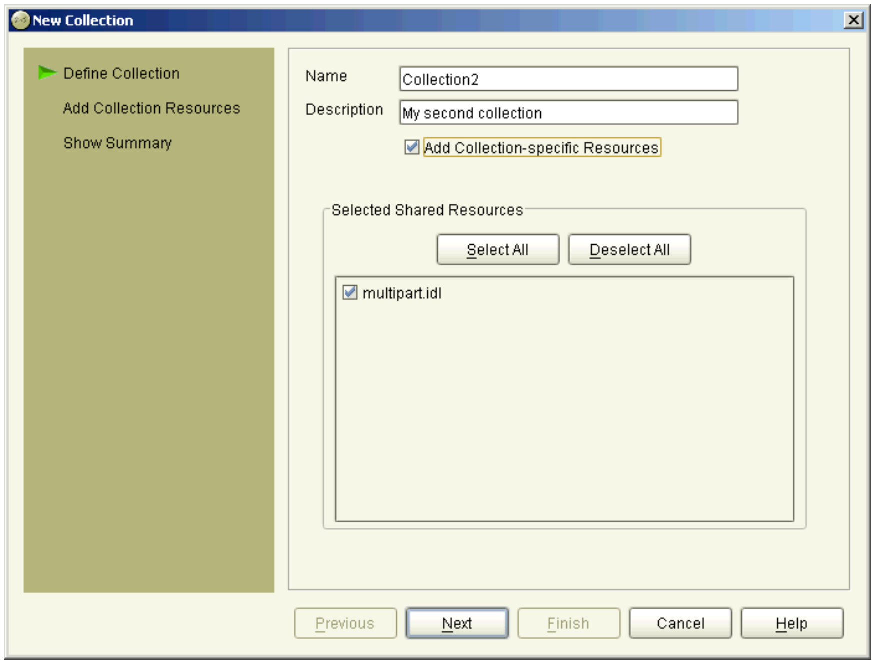
☐ NewCollection My first collection

Deployment Profile

Currently there are no deployment profiles in this workspace.

Figure 16: *Workspace Details panel*

Either way, you arrive at the New Collection wizard, as shown in Figure 17, and you can proceed through the procedure outlined below.



Name: Collection2
Description: My second collection
☑ Add Collection-specific Resources

Selected Shared Resources
☑ multipart.idl

Figure 17: *New Collection wizard*

Procedure

1. Enter a name for your collection, or accept the default provided.
2. Enter a description of your collection. This description should explain the purpose of the collection.
3. If you want to add resources to this collection that are collection-specific, check the box provided. This will cause an extra panel (Add Collection Resources) to be added to the wizard.
4. By default, the shared resources contained in your workspace will be added to this collection. They are listed in the Shared Resources table. If you do not want to add any or all of these resources to your collection, click the check boxes to deselect them.

5. Click **Next** to display the Add Collection Resources panel, as shown in Figure 18. If you did not choose to add resources to this collection, the Summary panel will be displayed - see **step 8** for details.

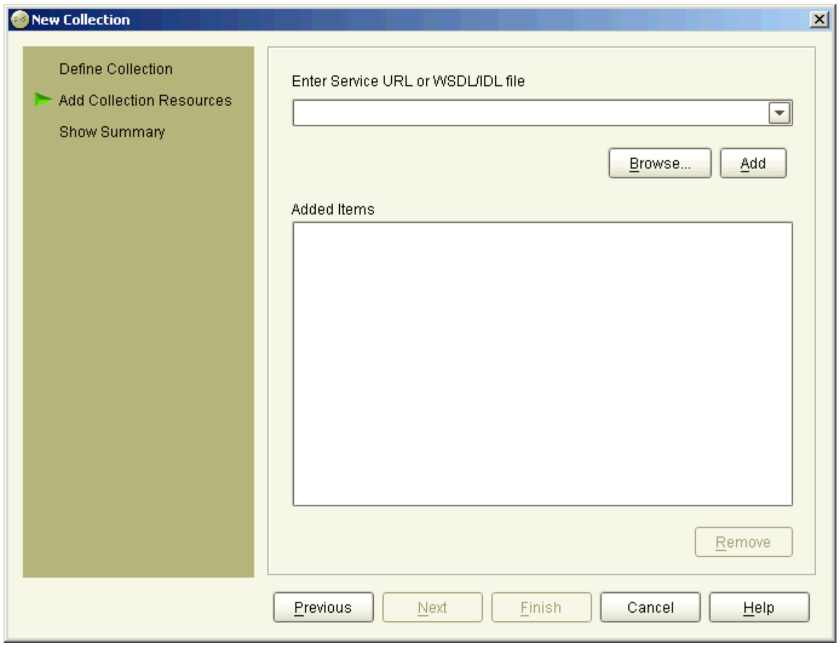
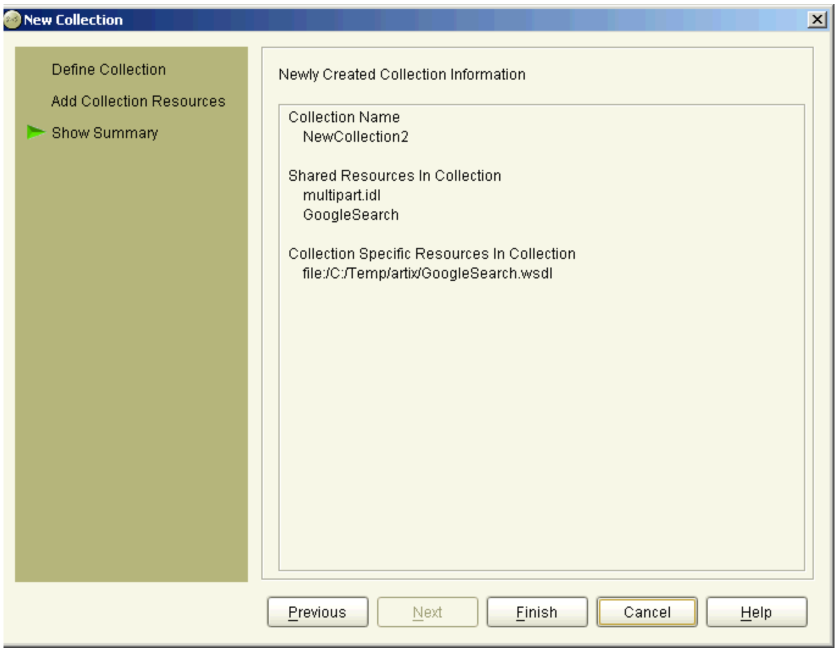


Figure 18: *New Collection wizard - Add Collection Resources panel*

6. Enter the URL or name of an existing file you would like to import into this collection as a resource. If you need to, click **Browse** to navigate to the file's location.
7. Click **Add** to add the file to the Added Items list and repeat as many times as necessary until you have added all the resources you want.

8. Click **Next** to display the Collection summary, as shown in Figure 19.



Newly Created Collection Information

Collection Name
  NewCollection2

Shared Resources In Collection
  multipart.idl
  GoogleSearch

Collection Specific Resources In Collection
  file:/C:/Temp/artix/GoogleSearch.wsdl

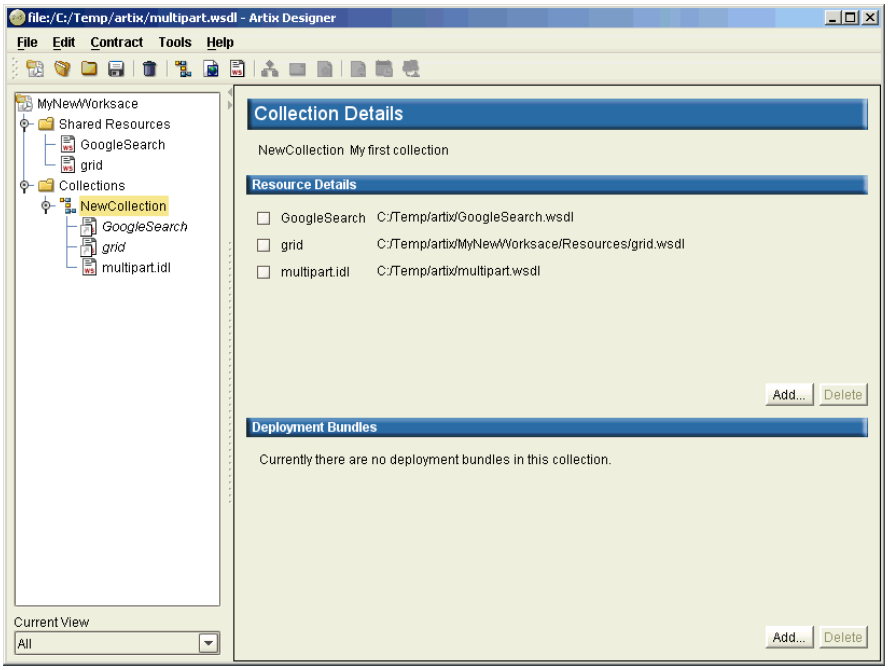
Figure 19: *New Collection wizard - Summary panel*

9. Click **Finish** to close this wizard and return to the Artix Designer, where you will see your new collection displayed in the Designer Tree.

Editing a Collection

Overview

You can make changes to your collection, or the resources within it, at any time using the Collection Details panel, as shown in Figure 20.



NewCollection My first collection

☐	GoogleSearch	C:/Temp/artix/GoogleSearch.wsdl
☐	grid	C:/Temp/artix/MyNewWorksace/Resources/grid.wsdl
☐	multipart.idl	C:/Temp/artix/multipart.wsdl

Currently there are no deployment bundles in this collection.

Figure 20: *Collection Details panel*

Adding and deleting resources

To add a resource, click **Add** to display the New Contract from File/URL dialog, as shown in Figure 21.

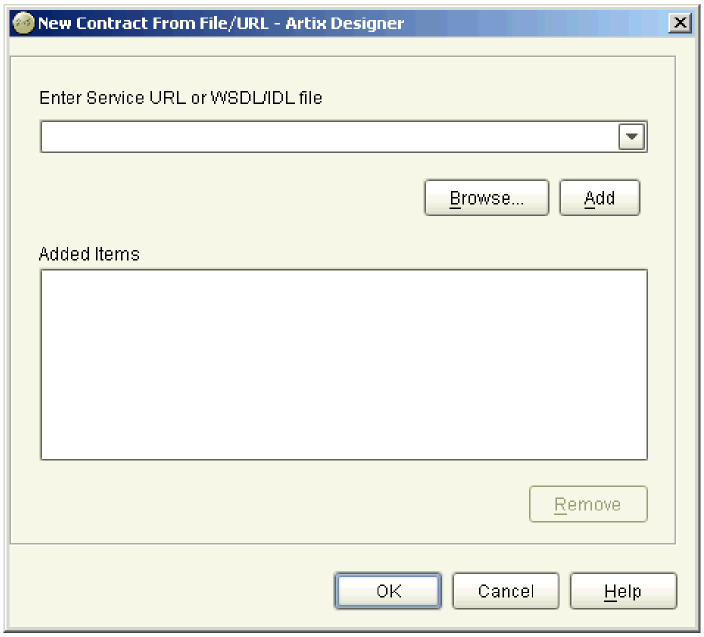

New Contract From File/URL - Artix Designer
Enter Service URL or WSDL/IDL file
Browse...
Add
Added Items
Remove
OK
Cancel
Help

Figure 21: *New Contract from File/URL dialog*

For help with this dialog, and other information about adding resources, see "Working with Artix Resources" on page 37.

To delete a resource you need to first select it on the Collection Details panel using the check box provided, and then click **Delete**.

Adding and deleting Deployment Bundles

To add a Deployment Bundle, click the **Add** button under the Deployment Bundles list on the Collection Details panel to display the Deployment Bundle wizard.

To delete a Deployment Bundle, you need to first select it using the check box provided, and then click **Delete**.

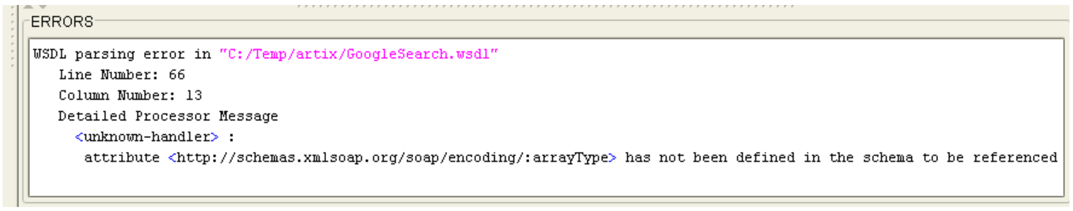
Editing deployed collections

If you make changes to any contract in a collection that has been deployed, you should be aware that these changes could make the deployment bundle(s) you have created for that collection invalid. It is recommended, therefore, that you redeploy the bundle using the Run Deployer dialog any time that you change a previously deployed collection.

For more information, see "Deploying the Bundle" on page 155 for more information.

Invalid WSDL

Also, be aware that any changes you make to a resource could leave its underlying WSDL document invalid - if this happens you will only be able to view the contract by selecting the WSDL tab of the Contract Navigator, as shown in Figure 23 on page 42. In this view, any problems with the WSDL are listed, as shown in Figure 22, so that you can fix them and return the WSDL to a valid state.


ERRORS

```
WSDL parsing error in "C:/Temp/artix/GoogleSearch.wsdl"
   Line Number: 66
   Column Number: 13
   Detailed Processor Message
     <unknown-handler> :
      attribute <http://schemas.xmlsoap.org/soap/encoding/:arrayType> has not been defined in the schema to be referenced
```

Figure 22: *Artix Designer Invalid WSDL Indicator*

Converting resources to shared

You can change a collection-specific resource within a collection to a shared resources, and thus have the opportunity to add it to all of your collections.

To do this, select the resource name in the Designer Tree and select **Contract | Convert to Shared**. This will invoke a dialog asking which other collections you would also like to include this resource - by default, all collections are selected.

Click **OK** to close this dialog and return to the Artix Designer. The resource is now in the Shared Resources list, plus all collections that you specified.

Deploying a Collection

As mentioned at the beginning of this chapter, collections can be generated into physical entities such as Java, C++, or CORBA based applications. Further, collections can be deployed as clients, servers, or switches, depending on your solution requirements.

The Artix deployment process has three steps:

1. Create a **Deployment Profile** - contains machine level information such as the Artix save location, the compiler location, and the operating system being used. A profile can be used multiple times as it is not specific to any particular collection defined within the workspace.

   To create a deployment profile, select **File | New | Deployment Profile** from the menu bar.

2. Create a **Deployment Bundle** - defines specific information about the deployment of a collection, such as the deployment type (client, server, or switch), configuration details, and code generation options.

   To create a deployment bundle, select a collection from the Designer Tree, then select **File | New | Deployment Bundle** from the menu bar.

3. Deploy the bundle - a very simple procedure once the profile and bundle are in place. Artix deploys the solution based on the information you provided in the bundle, and generates the code, environment scripts, and configuration files as specified in the locations you provided.

   To deploy a bundle, select a collection from the Designer Tree, then select **Tools | Run Deployer** from the menu bar.

After deploying the bundle, you need to hand edit the implementation code, and then you can run and compile the code.

For more information and detailed procedures for each of the deployment steps, see "Deploying Collections" on page 145.

CHAPTER 4

Working with Artix Resources

A resource is an XML document, sometimes also referred to as the Artix contract, that defines an interface to a collection.

In this chapter

This chapter discusses the following topics:

What are Resources?

Overview

An Artix Resource is an XML document that can be used to define the interface to a collection. In Artix 2.0 there are three supported resource types, that can define the Artix *contract*:

- WSDL documents
- WSDL created from IDL files
- WSDL created from data sets, such as COBOL Copybooks

When a resource is added to a workplace, it can take one of two roles:

- A "Shared" resource, which is automatically added to every collection in that workspace.
- A "collection-specific" resource, which only applies to the collection to which it is added.

How the Designer helps you create a contract

When building a WSDL contract, the Artix Designer guides you through the process by making only relevant options available to you depending on the current state of that contract.

For example, if you are building your contract from scratch you need to add components to it in a certain order - there are dependencies between the components. In short, the contents of the Designer's Contract menu, as shown below, reflect the order in which components need to be added to the contract.

- Types - the first item to be created. Not mandatory though, as some primitive types exist by default.
- Messages - cannot be created without a Type. The primitive types will suffice if you don't want to add new types.
- Port Types - cannot be created without a Message.
- Bindings - cannot be creating without a Port Type.
- Services - cannot be creating without a Binding.
- Routes - cannot be created without two compatible Services.

Contracts and their components are explained in more details on the following pages.

What is a Contract?

Overview

Artix contracts describe Artix resources and their integration. Each mapping of a port type to a binding and port defines an Artix collection. The contract also describes the routing between collections. It has two sections:

- Logical - describes the abstract operations, messages, and data types used by a collection.
- Physical - describes the concrete message formats and transports used by a collection. The routing information defining how messages are mapped between different collections is also specified here.

The Logical Section

The logical section of an Artix Contract defines the abstract operations that the collections offer. The logical view includes the `<types>`, `<message>`, and `<portType>` tags in a WSDL document. This portion of the contract also specifies the namespaces used in defining the contract.

Types

Applications typically use datatypes that are more complex than the primitive types, like `int`, defined by most programming languages. WSDL documents represent these complex datatypes using a combination of schema types defined in referenced external XML schema documents and complex types described in `<type>` elements.

For information about adding Types to your Artix contract, see "Adding Types" on page 48.

Messages

WSDL is designed to describe how data is passed over a network and because of this it describes data that is exchanged between two endpoints in terms of abstract messages described in `<message>` elements. Each abstract message consists of one or more parts, defined in `<part>` elements, that are the formal data elements of the abstract message. Each part is identified by a name and an attribute specifying its data type.

These abstract messages represent the parameters passed by the operations defined by the WSDL document and are mapped to concrete data formats in the WSDL document's `<binding>` elements.

For information about adding messages to your Artix contract, see "Adding Messages" on page 57.

Port Types

A portType can be thought of as an interface description and in many Web service implementations there is a direct mapping between port types and implementation objects. Port types are the abstract unit of a WSDL document that is mapped into a concrete binding to form the complete description of what is offered over a port.

Port types are described using the <portType> element in a WSDL document. Each port type in a WSDL document must have a unique name, specified using the name attribute, and is made up of a collection of operations, described in <operation> elements. A WSDL document can describe any number of port types.

For information about adding Port Types to your Artix contract, see "Adding Port Types" on page 61.

Operations

Operations, described in <operation> elements in a WSDL document are an abstract description of an interaction between two endpoints. For example, a request for a checking account balance and an order for a gross of widgets can both be defined as operations.

Each operation within a port type must have a unique name, specified using the name attribute. The name attribute is required to define an operation.

Each operation is made up of a set of elements. The elements represent the messages communicated between the endpoints to execute the operation.

For information about adding Operations to your Artix contract, see the Operations section in "Adding Port Types" on page 61.

The Physical Section

The physical section of an Artix contract defines the bindings and transports used by the collections. It includes the information specified in the `<binding>` and `<service>` tags of a WSDL document. It also includes the routing rules defining how the messages are routed between the endpoints defined in the document.

Bindings

To define an endpoint that corresponds to a running service, port types are mapped to bindings which describe how the abstract messages defined for the port type map to the data format used on the wire. The bindings are described in `<binding>` elements. A binding can map to only one port type, but a port type can be mapped to any number of bindings.

WSDL is intended to describe services offered over the Web and therefore most bindings are specified using SOAP as the message format. WSDL can bind data to other message formats however.

Artix provides bindings for several message formats including CORBA and FML. For specific information on using bindings see "Adding Bindings" on page 93.

Services

The final piece of information needed to describe how to connect a remote service is the network information needed to locate it. This information is defined inside a `<port>` element. Each port specifies the address and configuration information for connecting the application to a network.

Ports are grouped within `<service>` elements. A service can contain one or many ports. The convention is that the ports defined within a particular service are related in some way. For example all of the ports might be bound to the same port type, but use different network protocols, like HTTP and WebSphere MQ. For more information, see "Adding Services" on page 109.

Routing

To fully describe the integration of collections across an enterprise, Artix contracts include routing rules for directing data between the collections. Routing rules are described in "Routing Messages" on page 135.

For more information

For more detailed information about Artix contracts and their components, see either the *Artix Getting Started Guide*, or *Developing Artix Solutions from the Command Line*.

Navigating Through Resources - The Contract Navigator

Overview

The Artix Designer provides an interface, called the Contract Navigator, that gives you two views of a contract - graphical or WSDL. These views are accessed via tabs at the bottom of the Designer's content pane.

Graph view

Depending on how you want to work with your contract, you can use either of the available views. If you aren't very familiar with WSDL, you will probably prefer to work in the **Graph** view, as shown in Figure 23.

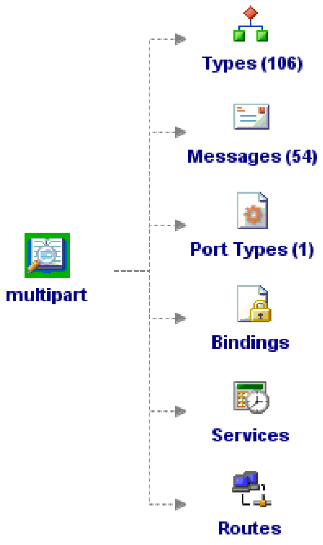


Figure 23: *Contract Navigator - Graph view*

The WSDL model

The graph view shows you the WSDL model. As seen in Figure 23, the multi-part contract has 106 types, 54 messages, and 1 port type. It currently contains no bindings, services, or routes.

If you right-click on one of the components, such as types, you are given the option to create a new type, or to edit or view the existing types. You can also expand and collapse the 106 existing types.

The model expanded

When you expand the existing types, the Contract Navigator changes to look more like Figure 24.

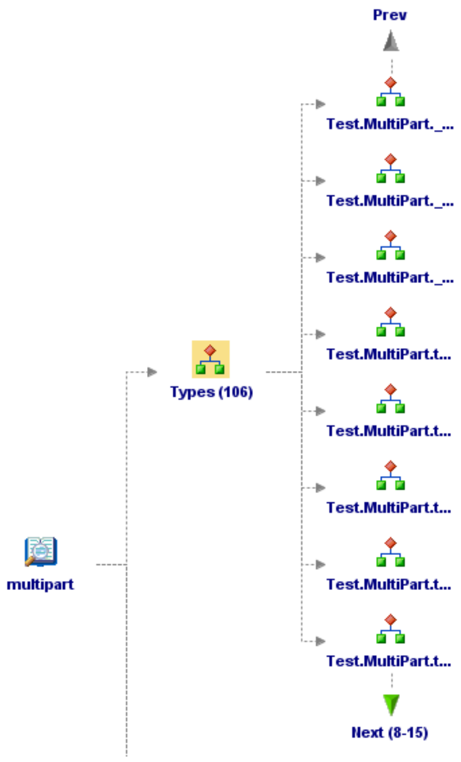


Figure 24: *Contract Navigator showing Types Expanded*

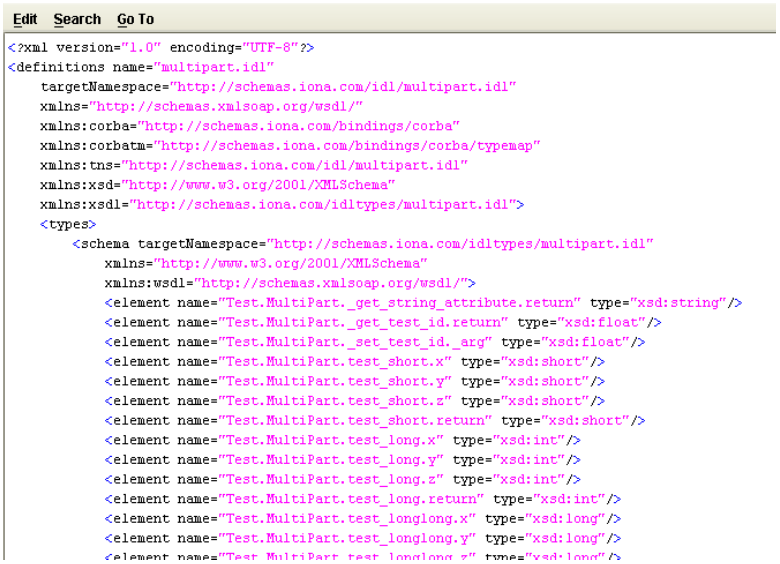
Navigating the expanded model

Now you can scroll through the individual types and right-click on any of them to edit or view the attributes for that type. The Contract Navigator lists eight "child" components at any one time. To scroll to view the next or the previous eight, you just need to click on the **Next** or **Previous** arrows.

If you want to see a list of all of the child components, right-click on either of the arrows and a scrollable list is displayed.

WSDL view

If you prefer, you can work directly in the WSDL by selecting the **WSDL** tab, as shown in Figure 25.



```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="multipart.idl"
    targetNamespace="http://schemas.iona.com/idl/multipart.idl"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:corba="http://schemas.iona.com/bindings/corba"
    xmlns:corbatm="http://schemas.iona.com/bindings/corba/typemap"
    xmlns:tns="http://schemas.iona.com/idl/multipart.idl"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsdl="http://schemas.iona.com/idltypes/multipart.idl">
    <types>
        <schema targetNamespace="http://schemas.iona.com/idltypes/multipart.idl"
            xmlns="http://www.w3.org/2001/XMLSchema"
            xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
            <element name="Test.MultiPart._get_string_attribute.return" type="xsd:string"/>
            <element name="Test.MultiPart._get_test_id.return" type="xsd:float"/>
            <element name="Test.MultiPart._set_test_id._arg" type="xsd:float"/>
            <element name="Test.MultiPart.test_short.x" type="xsd:short"/>
            <element name="Test.MultiPart.test_short.y" type="xsd:short"/>
            <element name="Test.MultiPart.test_short.z" type="xsd:short"/>
            <element name="Test.MultiPart.test_short.return" type="xsd:short"/>
            <element name="Test.MultiPart.test_long.x" type="xsd:int"/>
            <element name="Test.MultiPart.test_long.y" type="xsd:int"/>
            <element name="Test.MultiPart.test_long.z" type="xsd:int"/>
            <element name="Test.MultiPart.test_long.return" type="xsd:int"/>
            <element name="Test.MultiPart.test_longlong.x" type="xsd:long"/>
            <element name="Test.MultiPart.test_longlong.y" type="xsd:long"/>
            <element name="Test.MultiPart.test_longlong.z" type="xsd:long"/>
```

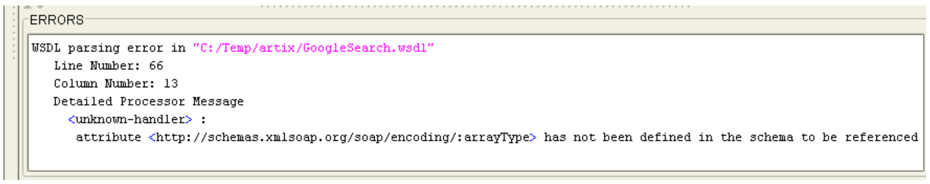
Figure 25: *Contract Navigator - WSDL view*

Editing tools

In this view you can hand edit the WSDL, and the Designer provides tools under the **Edit** menu in this view to make the task easier. You can also use the **Search** and **Go To** functions to locate segments or specific lines within the WSDL.

Validating your changes

When you make changes to the WSDL and click **Apply Edits**, the Designer checks that your changes have not compromised the WSDL. If your changes have made the WSDL invalid, the errors are listed in the Errors panel so that you can go to the relevant line and correct them. Figure 26 shows an example of the WSDL Error panel.


ERRORS

```
WSDL parsing error in "C:/Temp/artix/GoogleSearch.wsdl"
   Line Number: 66
   Column Number: 13
   Detailed Processor Message
     <unknown-handler> :
      attribute <http://schemas.xmlsoap.org/soap/encoding/:arrayType> has not been defined in the schema to be referenced
```

Figure 26: *WSDL Error panel*

Creating New Contracts

Overview

Creating Artix contracts from scratch takes a little time, but is still easy to do using the Artix Designer. Wizards guide you through the process.

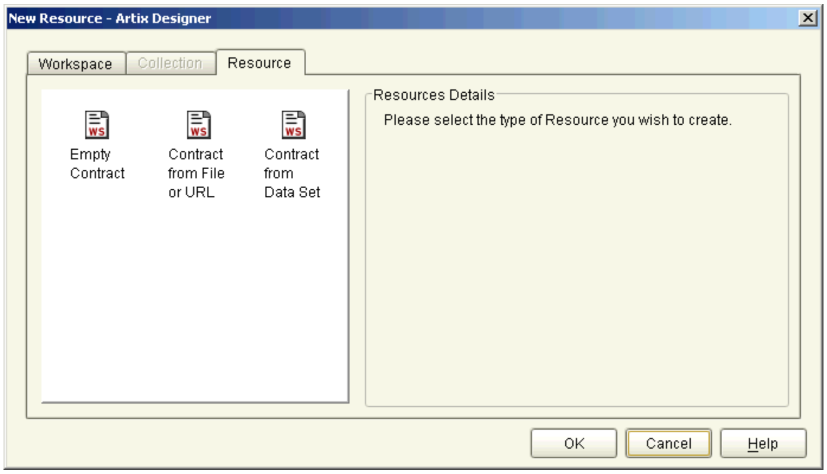
As explained in "Creating a Collection" on page 28, the WSDL contract is made up of a logical and a physical part. The logical part contains types, messages, port types and operations. The physical part contains services and bindings.

This section explains how to add each of the logical components of the Artix contract. For information on adding services to your contract, see "Adding Services" on page 109. For information on adding bindings to your contract, see "Adding Bindings" on page 93.

Creating the contract

The first thing you need to do is create the empty contract. To do this:

1. Select either the Shared Resources folder or a Collection from the Designer Tree.
2. Select **File | New | Resource** from the **File** menu to display the New Resource selection panel, as shown in Figure 27.



Please select the type of Resource you wish to create.

Figure 27: *New Resource dialog*

3. Select the **Empty** Contract icon and click **OK** to display the New Contract dialog, as shown in Figure 28.

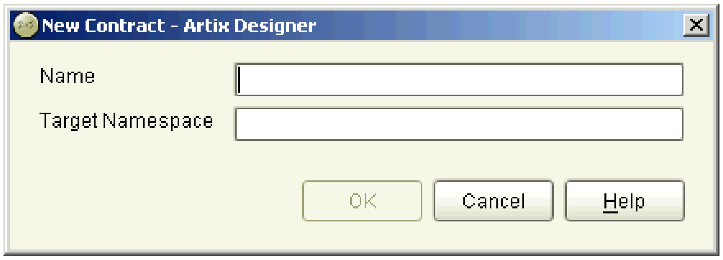


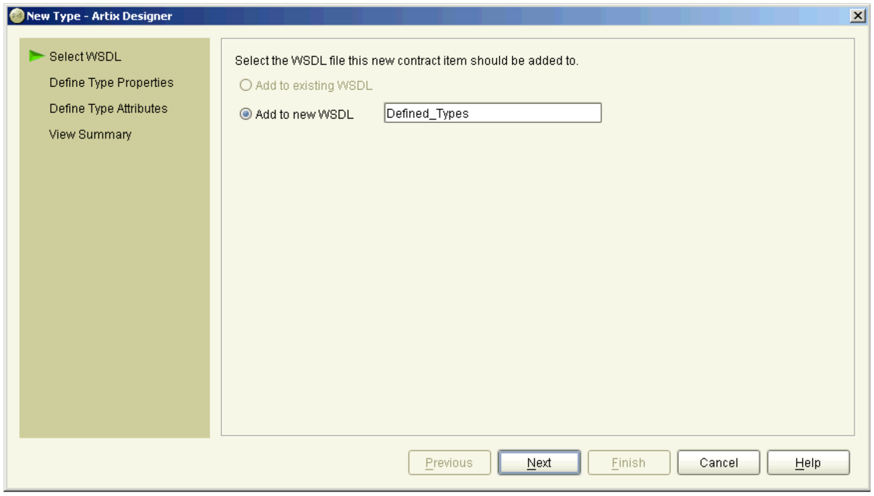
Figure 28: *New Contract dialog*

4. Enter a name in the **Name** field.
5. Enter a value in the **Target Namespace** field.
6. Click **OK** to close this dialog and return to the Artix Designer. Your new contract will be listed on the Tree and you are now ready to add elements to it using the procedures documented over the following pages.

Adding Types

To add a Type to your Artix contract:

1. Select **Contract | New | Type** from the menu bar to display the New Type wizard, as shown in Figure 29.



Select the WSDL file this new contract item should be added to.

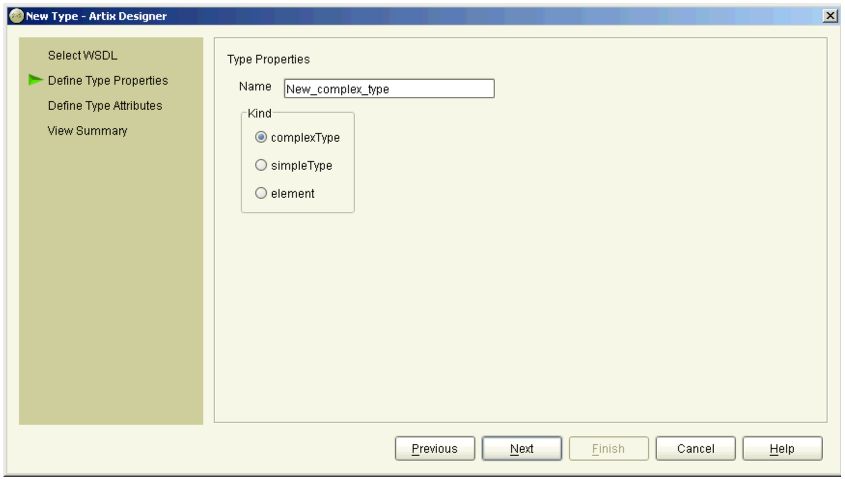
Add to existing WSDL [✓ Add to new WSDL] Defined_Types

Figure 29: *New Types wizard*

2. Select where to create the WSDL entry for the new type.
   - **Add to existing WSDL** adds the type information to the existing contract.
   - **Add to new WSDL** creates a new WSDL document that contains the type information.

   If, like in this example, you have an instance where the first option on this panel - Add to existing WSDL - is not able to be selected, it indicates that your WSDL file is read-only. Thus, you only have the option or creating a new WSDL file for the new type.

3. Click **Next** to display the Type Properties panel as displayed in Figure 30.



Type Properties

Name: New_complex_type

Kind: [✓ complexType] simpleType element

Figure 30: *New Types wizard - Type Properties panel*

4. Enter a name for the new type, or accept the default provided.
5. Select the Kind value for the type - **complex, simple, or element**.

6. Click **Next** to display the Type Attributes panel, as shown in Figure 31.

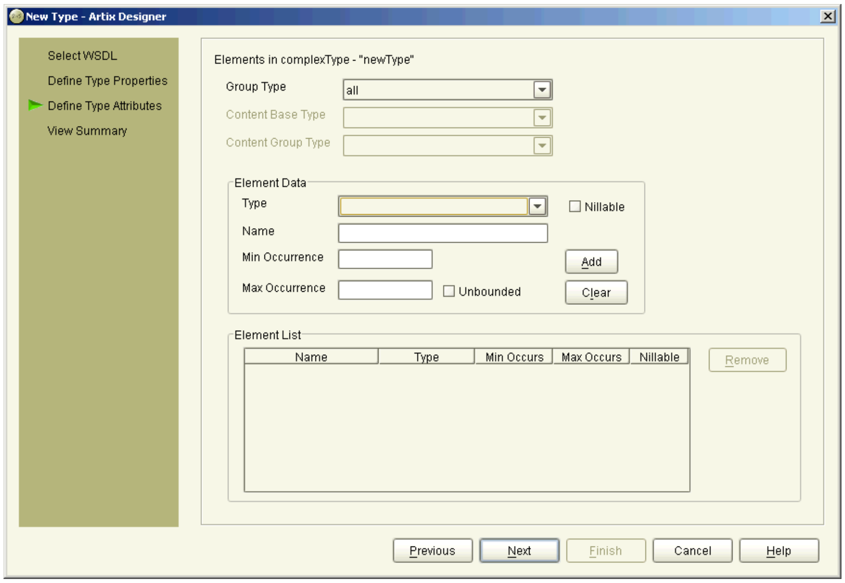


Figure 31: *New Types wizard - Type Attributes (complex) panel*

7. Depending on the Kind of type you selected, different options are displayed on the Type Attributes panel. The example shown in Figure 31 shows the options for a complex type. Continue with **step 8**, if this is the kind of type you are creating, otherwise go to one of the following steps:
   - **Step 16** for a simple type
   - **Step 22** for an element

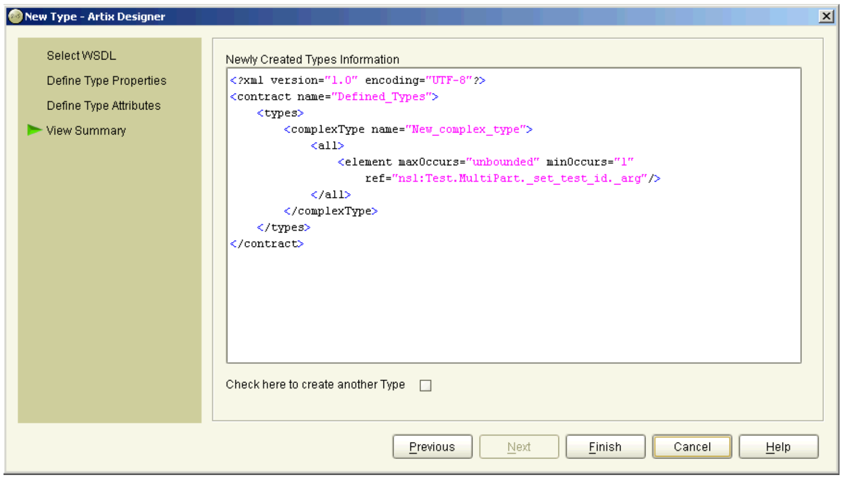
Complex type attributes

8. At the Type Attributes panel, as shown in Figure 31, select a Group Type value from the list provided. This defines how the complex type elements will be mapped to data structures.
9. Provide values for each of the Element Data fields:
    - Type - the base type for this schema
    - Name - a unique string identifier for element
    - Minimum Occurrence - the minimum times you want the element to be present
    - Maximum Occurrence - the maximum times you want the element to be present
10. Select the **Nillable** check box if you want to indicate that this element could potentially be omitted completely, or could pass an empty object across the wire.
11. Select the **Unbounded** check box if there is no maximum occurrence limit.
12. Click **Add** to move the details you have provided for this element into the Element List table.

    To edit an element in this table, select it and then make the changes in the fields above the table. Click **Update** to refresh the values in the table.

    You can delete an element from this table by selecting it and clicking **Remove**.
13. Repeat **steps 8 - 12** until you have added all of your elements.

14. Click **Next** to view the Summary panel, as shown in Figure 32.



```
<?xml version="1.0" encoding="UTF-8"?>
<contract name="Defined_Types">
    <types>
        <complexType name="New_complex_type">
            <all>
                <element maxOccurs="unbounded" minOccurs="1"
                    ref="ns1:Test.MultiPart._set_test_id._arg"/>
            </all>
        </complexType>
    </types>
</contract>
```

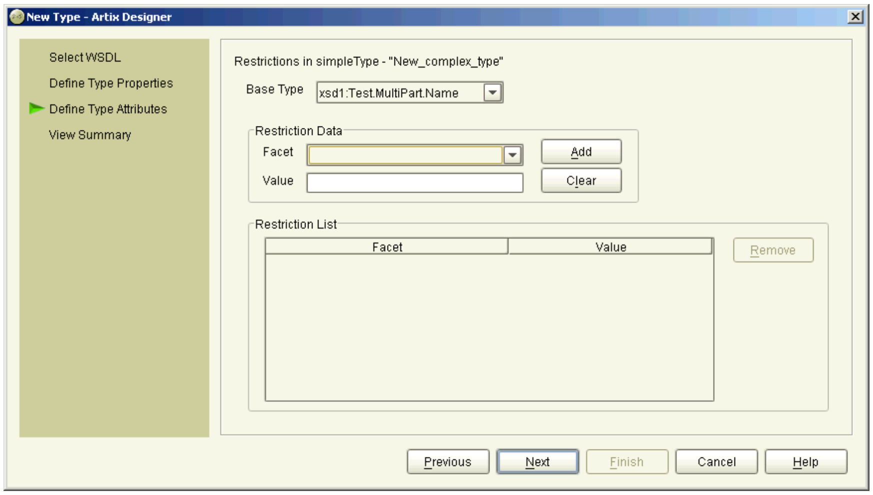
Figure 32: *New Types wizard - Summary panel for Complex Types*

15. If you would like to add another Type, click the check box provided and click **Next**. This will return you to the Type Properties panel, as displayed in Figure 30 on page 49.

    Alternatively, click **Finish** to close this wizard and return to the Artix Designer.

Simple type attributes

16. At the Type Attributes panel, as shown in Figure 33, select a Base Type from the list provided, for example, **string** or **boolean**.



Restrictions in simpleType - "New_complex_type"

Base Type: xsd1:Test.MultiPart.Name

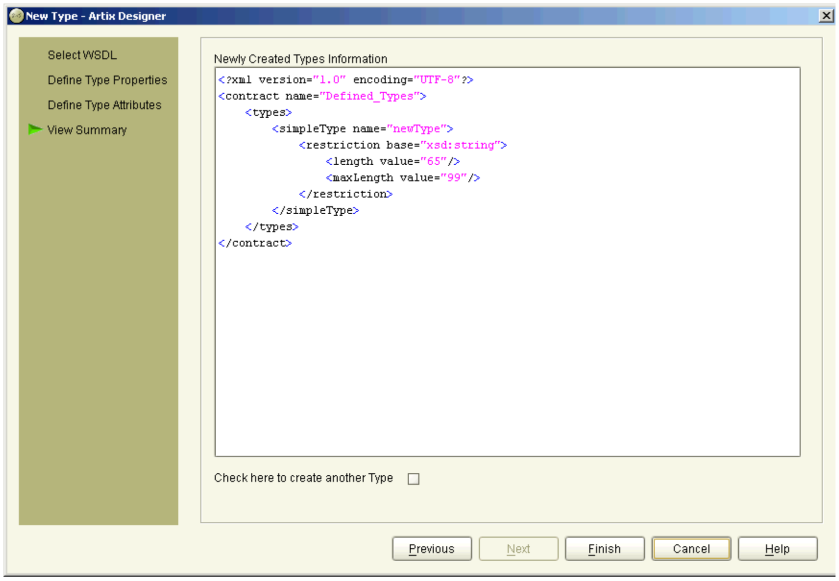
Figure 33: *New Types wizard - Type Attributes (simple) panel*

17. Provide values for each of the Restriction Data fields:
    - Facet - a characteristic of the base type, for example for a string, the available facets would be *enumeration*, *length*, or *maxLength.*
    - Value - the value for the facet, for example the value for *length* would be a non-negative integer.
18. Click **Add** to move the details you have provided for this restriction into the Restriction List table.

    To edit a restriction in this table, select it and then make the changes in the fields above the table. Click **Update** to refresh the values in the table.

    You can delete a restriction from this table by selecting it and clicking **Remove**.
19. Repeat **steps 16 - 18** until you have added all of your restrictions.

20. Click **Next** to view the Summary panel, as shown in Figure 34.



```
<?xml version="1.0" encoding="UTF-8"?>
<contract name="Defined_Types">
    <types>
        <simpleType name="newType">
            <restriction base="xsd:string">
                <length value="65"/>
                <maxLength value="99"/>
            </restriction>
        </simpleType>
    </types>
</contract>
```

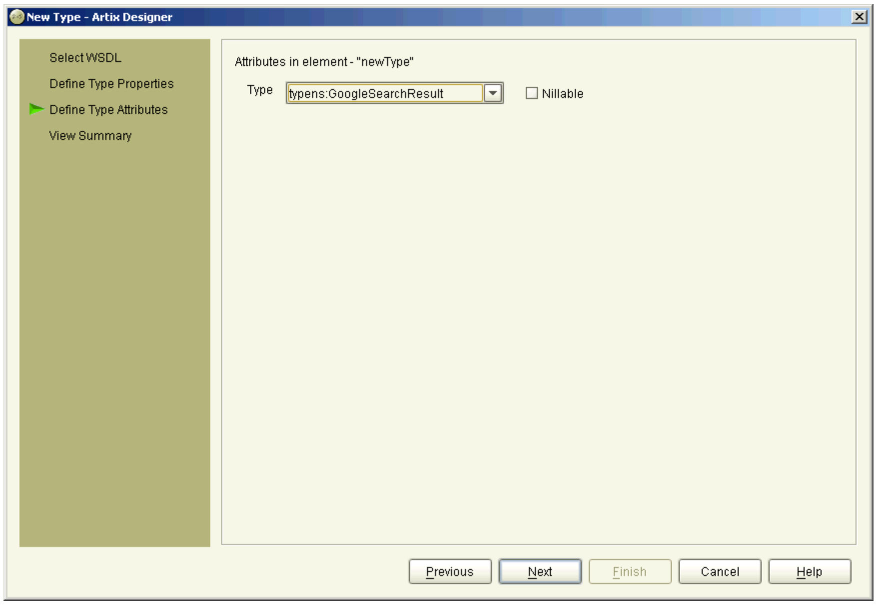
Figure 34: *New Type wizard - Summary panel for Simple Types*

21. If you would like to add another Type, click the check box provided and click **Next**. This will return you to the Type Properties panel, as displayed in Figure 30 on page 49.

    Alternatively, click **Finish** to close this wizard and return to the Artix Designer.

Element attributes

22. At the Type Attributes panel, as shown in Figure 35, select an Attribute Type from the list provided, for example, **string** or **boolean**.



Attributes in element - "newType"

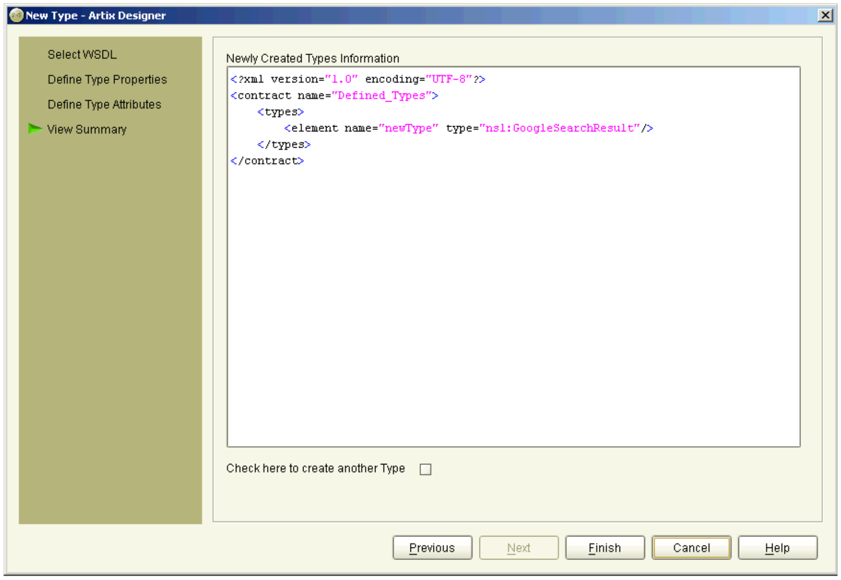
Type typens:GoogleSearchResult

Nillable

Figure 35: *New Types wizard - Type Attributes (element) panel*

23. Select the Nillable check box if you want to indicate that this element could potentially be omitted completely, or could pass an empty object across the wire.

24. Click **Next** to view the Summary panel, as shown in Figure 36.



```
<?xml version="1.0" encoding="UTF-8"?>
<contract name="Defined_Types">
    <types>
        <element name="newType" type="ns1:GoogleSearchResult"/>
    </types>
</contract>
```

Check here to create another Type ☐

Figure 36: *New Type wizard - Summary panel for Element*

25. If you would like to add another Type, click the check box provided and click **Next**. This will return you to the Type Properties panel, as displayed in Figure 30 on page 49.

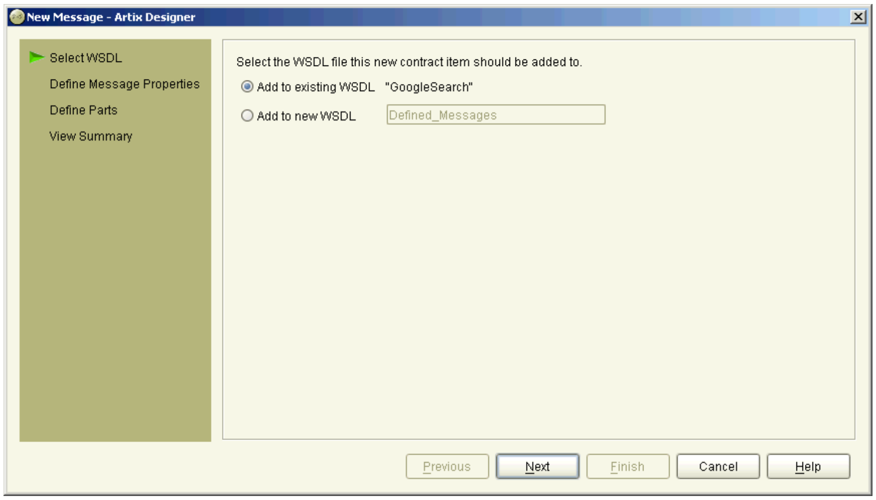
    Alternatively, click **Finish** to close this wizard and return to the Artix Designer.

Adding Messages

Procedure

To add a Message to your Artix contract:

1. Select **Contract | New | Message** from the menu bar to display the New Message wizard, as shown in Figure 37.



Select the WSDL file this new contract item should be added to.

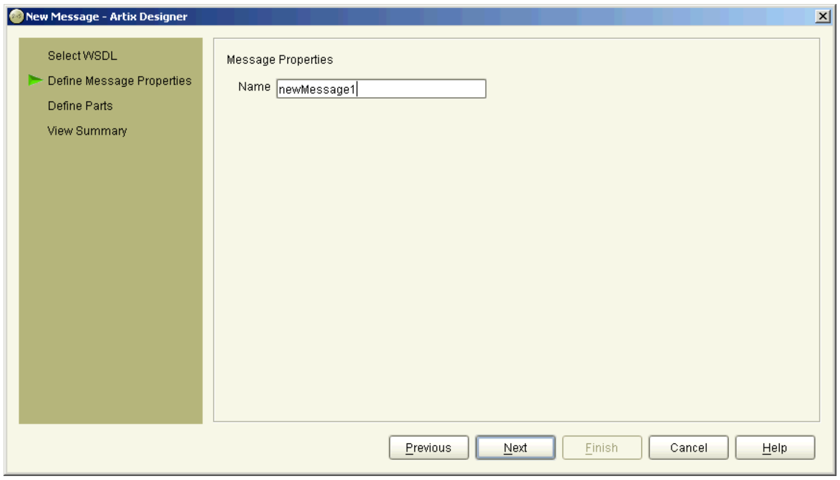
(●) Add to existing WSDL "GoogleSearch"

( ) Add to new WSDL Defined_Messages

Figure 37: *New Message wizard*

2. Select where to create the WSDL entry for the new message.
   - **Add to existing WSDL** adds the message information to an existing contract.
   - **Add to new WSDL** creates a new WSDL document that contains the message information.

3. Click **Next** to display the Message Properties panel, as shown in Figure 38.



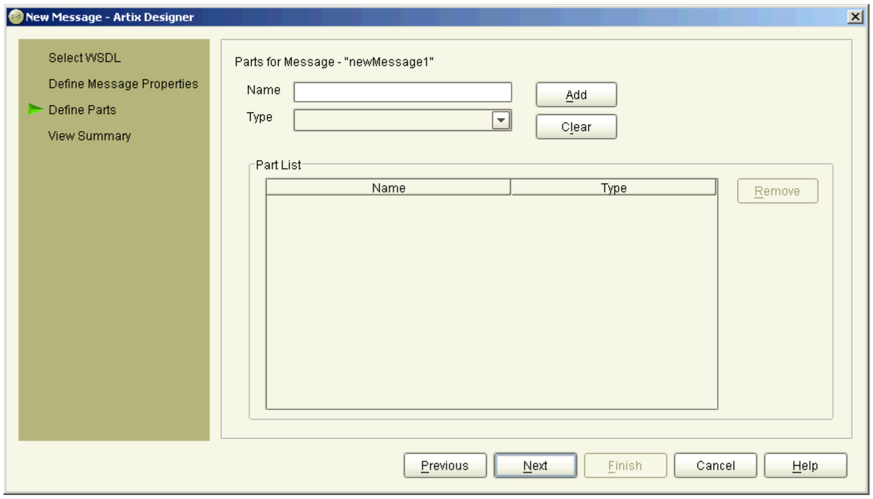
Message Properties

Name: newMessage1

Figure 38: *New Message wizard - Message Properties panel*

4. Enter a name for the message, or accept the default provided.

5. Click **Next** to display the Message Parts panel, as shown in Figure 39.



Parts for Message - "newMessage1"

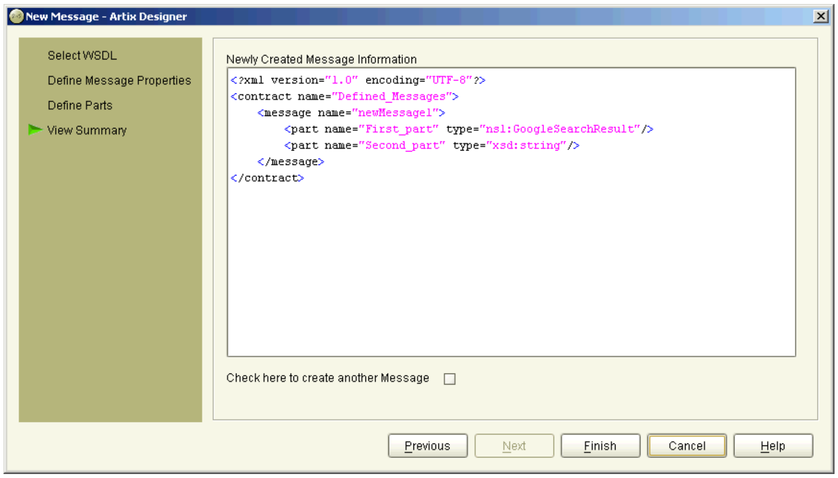
Figure 39: *New Message wizard - Message Parts panel*

6. Enter a name for the message part, and select a type from the list provided.
7. Click **Add** to move the details you have provided for this part into the Part List table.

   To edit a part in this table, select it and then make the changes in the fields above the table. Click **Update** to refresh the values in the table.

   You can delete a part from this table by selecting it and clicking **Remove**.
8. Repeat **steps 6 and 7** until you have added all of your parts.

9. Click **Next** to view the Summary panel, as shown in Figure 40.


New Message - Artix Designer
Select WSDL
Define Message Properties
Define Parts
View Summary
Newly Created Message Information
Check here to create another Message
Previous
Next
Finish
Cancel
Help

```
<?xml version="1.0" encoding="UTF-8"?>
<contract name="Defined_Messages">
    <message name="newMessage1">
        <part name="First_part" type="ns1:GoogleSearchResult"/>
        <part name="Second_part" type="xsd:string"/>
    </message>
</contract>
```

Figure 40: *New Messages wizard - Summary panel*

10. If you would like to add another Message, click the check box provided and click **Next**. This will return you to the Message Properties panel, as displayed in Figure 38 on page 58.

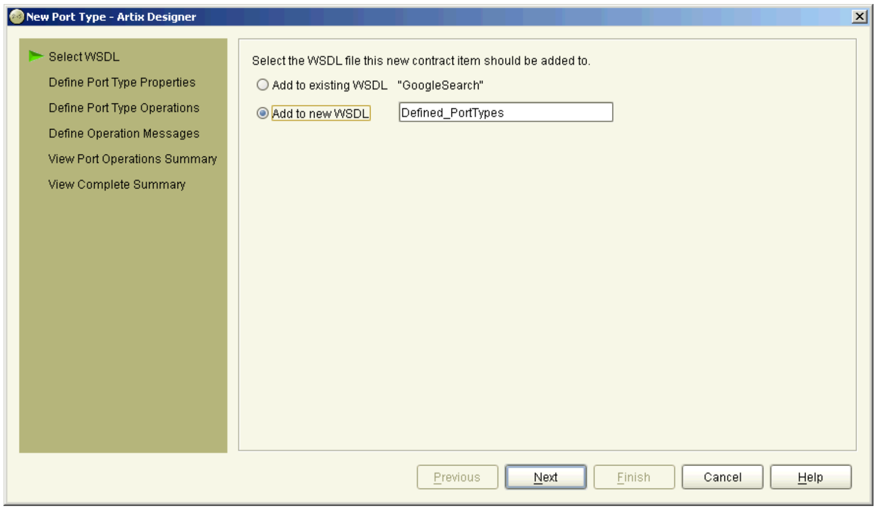
    Alternatively, click **Finish** to close this wizard and return to the Artix Designer.

Adding Port Types

Procedure

To add a Port Type to your Artix contract:

1. Select **Contract** | **New** | **Port Type** from the menu bar to display the New Port Type wizard, as shown in Figure 41.


New Port Type - Artix Designer
Select WSDL
Define Port Type Properties
Define Port Type Operations
Define Operation Messages
View Port Operations Summary
View Complete Summary
Previous
Next
Finish
Cancel
Help

Select the WSDL file this new contract item should be added to.

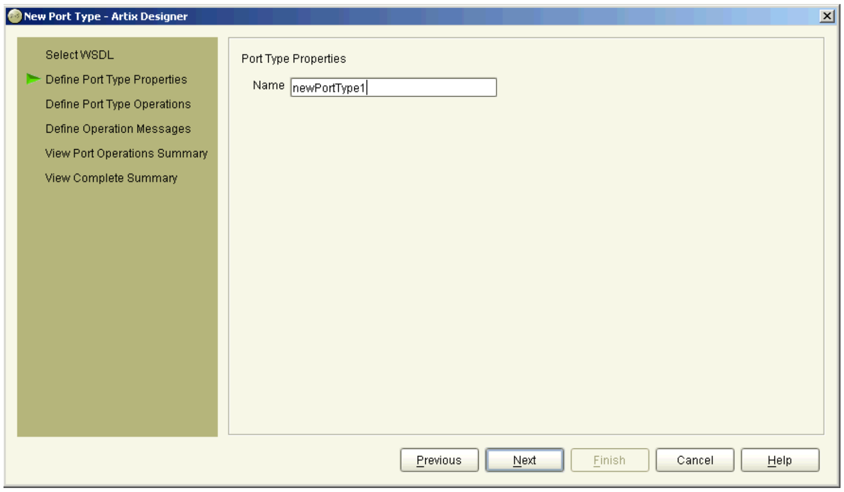
( ) Add to existing WSDL "GoogleSearch"

(●) Add to new WSDL Defined_PortTypes

Figure 41: *New Port Type wizard*

2. Select where to create the WSDL entry for the new port type.
   - **Add to existing WSDL** adds the port type information to the existing contract.
   - **Add to new WSDL** creates a new WSDL document that contains the port type information.

3. Click **Next** to display the Port Type Properties panel, as shown in Figure 42.


New Port Type - Artix Designer
Select WSDL
Define Port Type Properties
Define Port Type Operations
Define Operation Messages
View Port Operations Summary
View Complete Summary
Previous
Next
Finish
Cancel
Help

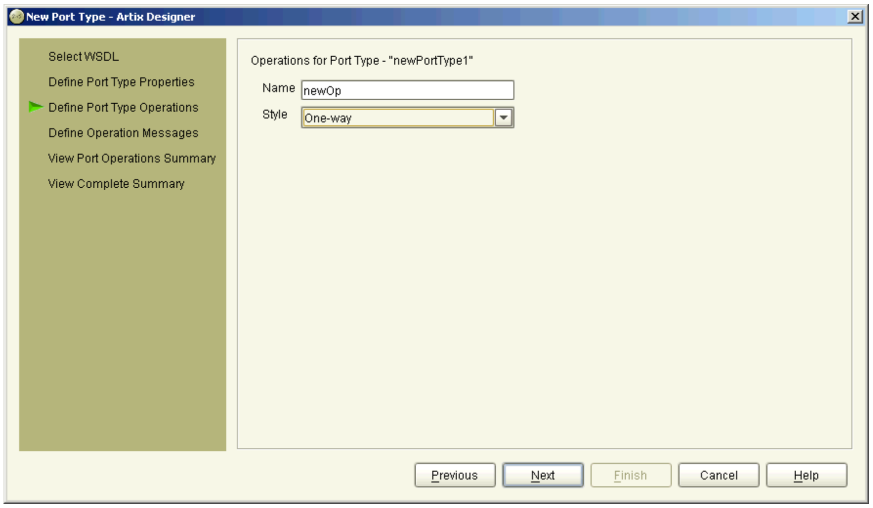
Port Type Properties

Name newPortType1

Figure 42: *New Port Type wizard - Port Type Properties panel*

4. Enter a name for the port type, or accept the default provided.

5. Click **Next** to display the Port Type Operations panel, as shown in Figure 43.


New Port Type - Artix Designer
Select WSDL
Define Port Type Properties
Define Port Type Operations
Define Operation Messages
View Port Operations Summary
View Complete Summary
Previous
Next
Finish
Cancel
Help

Operations for Port Type - "newPortType1"

Name newOp

Style One-way

Figure 43: *New Port Type wizard - Port Type Operations panel*

6. Enter a name for the new operation and select a style from the list provided. Valid options are:
   - one-way
   - request-response

7. Click **Next** to display the Operation Messages panel, as shown in Figure 44.

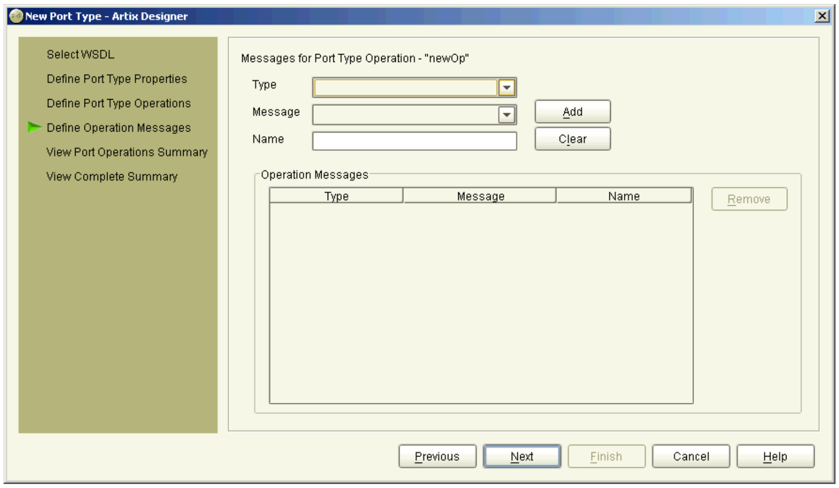

New Port Type - Artix Designer
Select WSDL
Define Port Type Properties
Define Port Type Operations
Define Operation Messages
View Port Operations Summary
View Complete Summary
Messages for Port Type Operation - "newOp"
Type
Message
Name
Add
Clear
Operation Messages
Type
Message
Name
Remove
Previous
Next
Finish
Cancel
Help

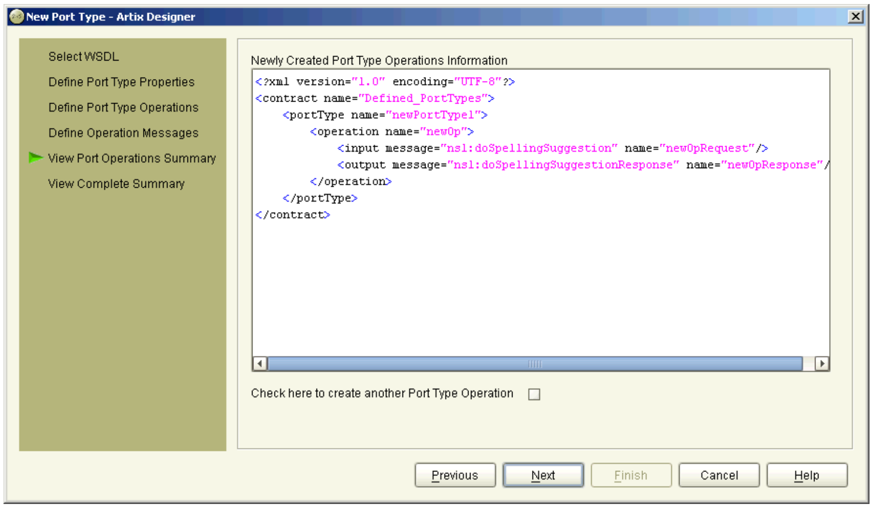
Figure 44: *New Port Type wizard - Operation Messages panel*

8. Select a Message Type from the list provided.
9. Select a Message from the list provided.
10. Enter a name for the message, or accept the one provided.
11. Click **Add** to move the details you have provided for this operation message into the Message List table.

    To edit a message in this table, select it and then make the changes in the fields above the table. Click **Update** to refresh the values in the table.

    You can delete a message from this table by selecting it and clicking **Remove**.

12. Click **Next** to display the Port Operations Summary panel, as shown in Figure 45.


New Port Type - Artix Designer
Select WSDL
Define Port Type Properties
Define Port Type Operations
Define Operation Messages
View Port Operations Summary
View Complete Summary
Previous
Next
Finish
Cancel
Help

Newly Created Port Type Operations Information

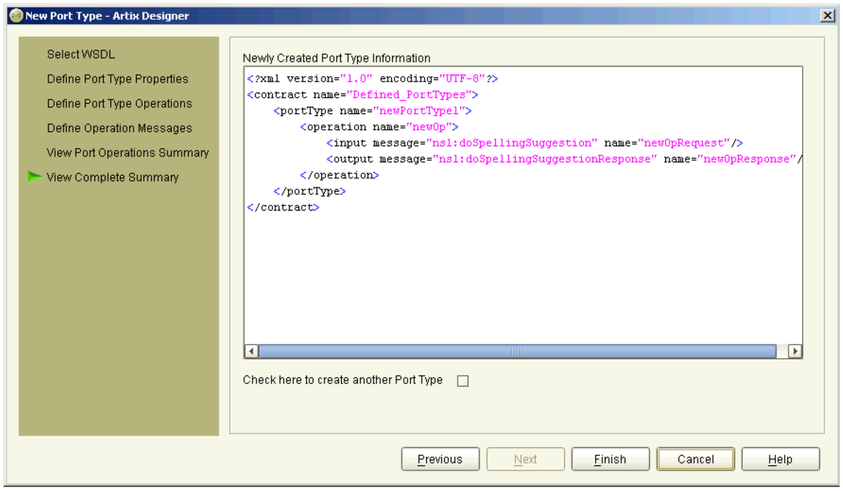
```
<?xml version="1.0" encoding="UTF-8"?>
<contract name="Defined_PortTypes">
    <portType name="newPortType1">
        <operation name="newOp">
            <input message="ns1:doSpellingSuggestion" name="newOpRequest"/>
            <output message="ns1:doSpellingSuggestionResponse" name="newOpResponse"/
        </operation>
    </portType>
</contract>
```

Check here to create another Port Type Operation ☐

Figure 45: *New Port Type wizard - Port Operations Summary panel*

13. If you would like to add another Port Type Operation, click the check box provided and click **Next**. This will return you to the Port Type Operation panel, as displayed in Figure 43 on page 63.

Alternatively, click Next to display the Port Type Summary panel, as shown in Figure 46.


New Port Type - Artix Designer
Select WSDL
Define Port Type Properties
Define Port Type Operations
Define Operation Messages
View Port Operations Summary
View Complete Summary
Newly Created Port Type Information
Check here to create another Port Type
Previous
Next
Finish
Cancel
Help

```
<?xml version="1.0" encoding="UTF-8"?>
<contract name="Defined_PortTypes">
    <portType name="newPortType1">
        <operation name="newOp">
            <input message="ns1:doSpellingSuggestion" name="newOpRequest"/>
            <output message="ns1:doSpellingSuggestionResponse" name="newOpResponse"/
        </operation>
    </portType>
</contract>
```

Figure 46: *New Port Type wizard - Port Type Summary panel*

14. If you would like to add another Port Type, click the check box provided and click **Next**. This will return you to the Port Type Properties panel, as displayed in Figure 42 on page 62.

    Alternatively, click **Finish** to close this wizard and return to the Artix Designer.

Creating Contracts from a File/URL

Overview

If you don't want to create your contract from scratch, you might be able to base it on an existing URL or File. You have three options:

- URL - you can use WSDL located at a URL address. For more information, see "Using a File or a URL to create a Contract" on page 68.
- WSDL - if you have some existing WSDL, you can import this into Artix and use it as is, or edit it to change its components. For more information, see "Using a File or a URL to create a Contract" on page 68.
- IDL - If you are starting from a CORBA server or client, Artix can generate the logical portion of the WSDL contract from IDL, automatically adding the required CORBA-specific entries and namespaces. For more information, see "Using IDL to create a Contract" on page 70.

  The IDL compiler also generates the binding information required to format the operations specified in the IDL. However, since port information is specific to the deployment environment, the port information is left blank, and you need to separately define a port using the Services wizard - "Adding Services" on page 109 for help with this task.

Using a File or a URL to create a Contract

Procedure

To use existing WSDL as the basis for your contract:

1. Select either the Shared Resources folder or a Collection from the Designer Tree.
2. Select **File | New | Resource** from the menu bar to display the New Resource selection panel, as shown in Figure 49.

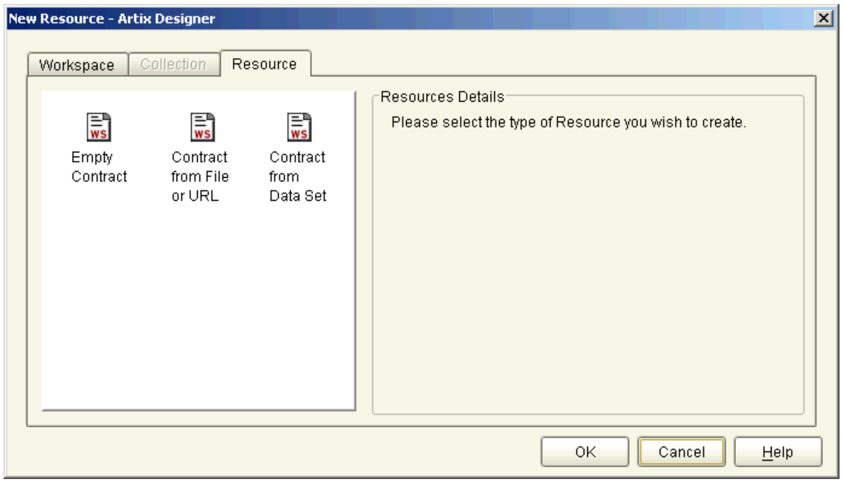

New Resource - Artix Designer
Workspace
Collection
Resource
Empty Contract
Contract from File or URL
Contract from Data Set
Resources Details
Please select the type of Resource you wish to create.
OK
Cancel
Help

Figure 47: *New Resource Selection panel*

3. Select the **Contract from File/URL** icon and click **OK** to display the New Contract from File/URL dialog, as shown in Figure 48.

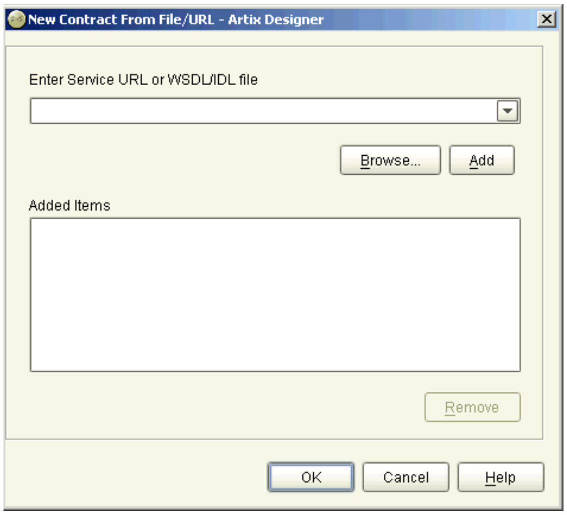

New Contract From File/URL - Artix Designer
Enter Service URL or WSDL/IDL file
Browse...
Add
Added Items
Remove
OK
Cancel
Help

Figure 48: *New Contract from File/URL dialog*

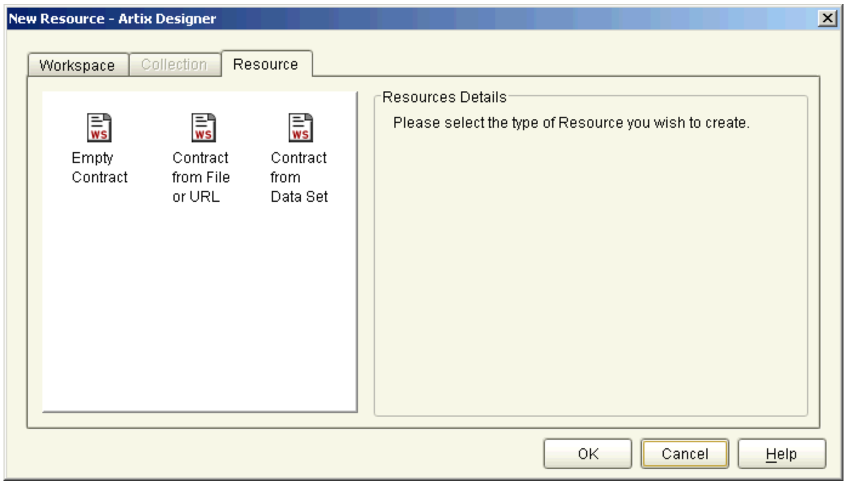
4. Either type the URL address, or click **Browse** to locate the WSDL file.
5. Click **Add** to move this resource to the **Added Items** list. Repeat **steps 3 - 5** to add as many more WSDL resources as you like.
6. Click **OK** to close this dialog and return to the Artix Designer. One contract will be listed under the selected collection for each WSDL file added or referenced.

Using IDL to create a Contract

Procedure

To use an IDL file as the basis for your contract:

1. Select either the Shared Resources folder or a Collection from the Designer Tree.
2. Select **New Resource** from the **File** menu to display the New Resource selection panel, as shown in Figure 49.



Please select the type of Resource you wish to create.

Figure 49: *New Resource dialog*

3. Select the **Contract from File/URL** icon and click **OK** to display the New Contract from File/URL dialog, as shown in Figure 50.

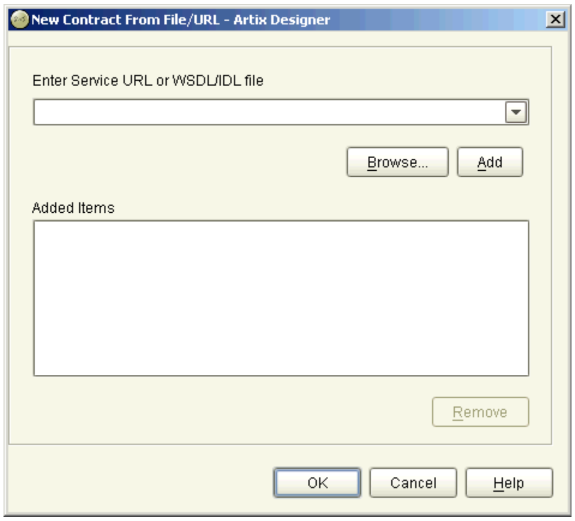
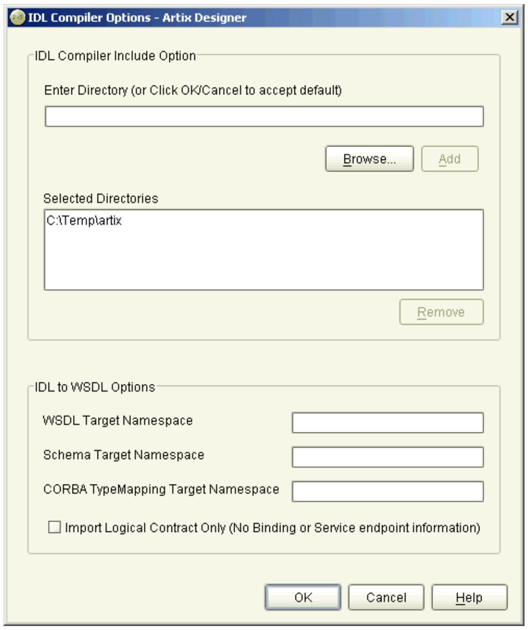

New Contract From File/URL - Artix Designer
Enter Service URL or WSDL/IDL file
Browse...
Add
Added Items
Remove
OK
Cancel
Help

Figure 50: *New Contract from File/URL dialog*

4. Click **Browse** to locate the IDL file you want to use as the resource for your Artix contract.

5. Click **Add** to move this file to the Added Items list and display the IDL Compiler Options dialog, as shown in Figure 51.


IDL Compiler Options - Artix Designer

IDL Compiler Include Option

Enter Directory (or Click OK/Cancel to accept default)

Selected Directories

C:\Temp\artix

IDL to WSDL Options

WSDL Target Namespace

Schema Target Namespace

CORBA TypeMapping Target Namespace

☐ Import Logical Contract Only (No Binding or Service endpoint information)

Figure 51: *IDL Compiler Options dialog*

6. Enter the names of the directories to search for included IDL files.
7. Click **Add** to add a directory to the list. Selecting a directory and clicking **Remove** will delete it from the list.
8. Add values to each of the namespace fields:
   - WSDL Target Namespace - the name the IDL Compiler will set for the *targetNamespace* value in the WSDL
   - Schema Target Namespace - the name the IDL compiler will set for the *targetNamespace* value in the Schema
   - CORBA TypeMapping Target Namespace - the name the IDL compiler will set for the CORBA *targetNamespace*

   If you do not set values for these fields, defaults will be assumed.

9. If you only wish to generate the logical portion of the contract select the **Logical Contract Only** check box.

   Note: If this option is selected the generated contracts will not contain any binding, CORBA typemap, or transport information.

10. Click **OK** to close this dialog and return to the New Contract from File/URL dialog
11. Repeat **steps 4 - 10** until you have added all of the IDL resources to import.
12. Click **OK** to close this dialog and return to the Artix Designer. One contract will be listed under the selected collection for each IDL file imported. The contracts will include a CORBA binding (unless you specified in **Step 9** to create only the Logical Contract), a CORBA type map, and a CORBA port description.

Deploying a service with the CORBA port

You need to add location information to the CORBA port before you can deploy a service using the CORBA port. For more information, see "Adding a CORBA Port" on page 113.

For information about deploying Artix solutions, see "Deploying Collections" on page 145.

Creating Contracts from Data Sets

Overview

The third way you can create new contracts is by basing them on a data set. Examples of this include:

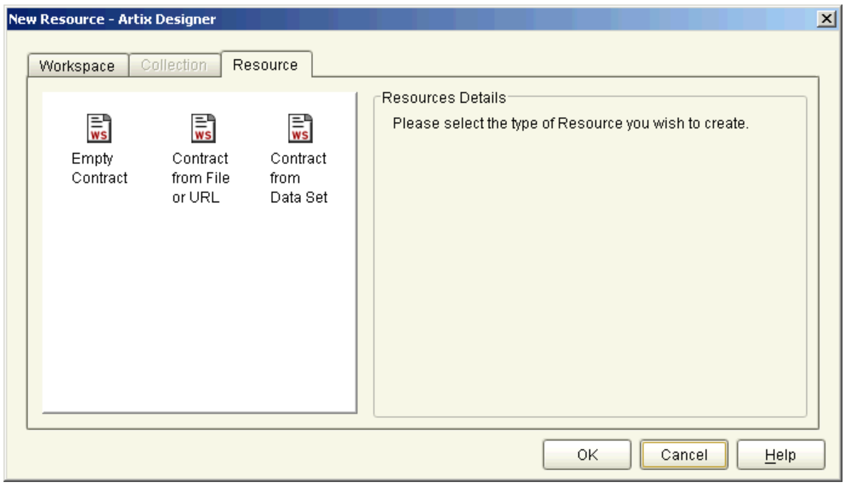
- Defining fixed data
- Using an existing COBOL Copybook to define the fixed data
- Defining tagged data

When you create a contract in this way, you're actually also creating the associated binding at the same time. When creating contracts using the other methods described in this chapter, the binding definition is a separate step.

Procedure

To create a contract from a data set:

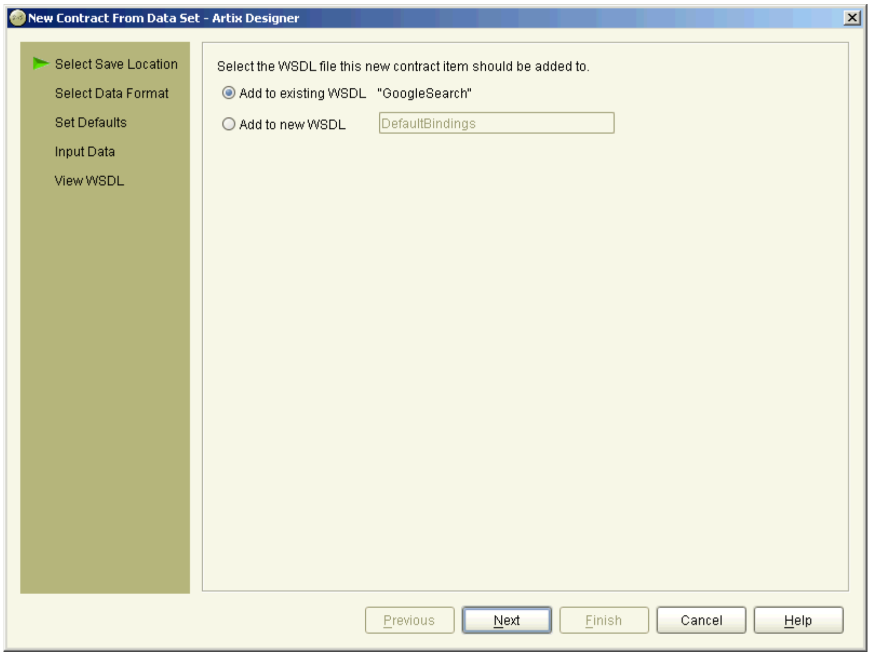
1. Select **New Resource** from the **File** menu to display the New Resource dialog, as shown in Figure 52.



Please select the type of Resource you wish to create.

Figure 52: *New Resource dialog*

2. Select the **New Contract from Data Set** icon, and click **OK** to display the New Contract from Data Set wizard, as shown in Figure 53.



Select the WSDL file this new contract item should be added to.

(●) Add to existing WSDL "GoogleSearch"

( ) Add to new WSDL DefaultBindings

Figure 53: *New Contract from Data Set wizard*

3. Enter a name for the WSDL that will contain the new binding, or accept the default provided.
4. Click **Next** to display the Data Format panel.

Now you need to turn to the relevant page, depending on what type of contract and binding you are creating. You can create:

- A contract containing a fixed binding - see page 76
- A contract containing a fixed binding from a CCB - see page 79
- A contract containing a tagged binding - see page 82

Creating a Contract Containing a Fixed Binding

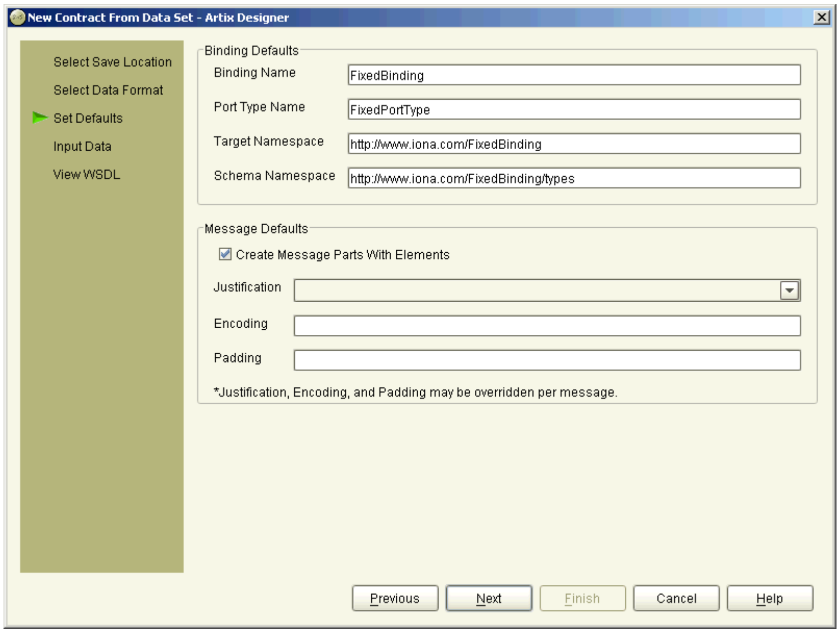
Overview

Many applications send data in fixed length records. For example, COBOL applications often send fixed record data over WebSphere MQ. Artix provides a binding that maps logical messages to concrete fixed record length messages. The fixed binding allows you to specify attributes such as encoding style, justification, and padding characters.

Procedure

To add a contract containing a Fixed binding:

1. At the Data Format panel, select **Fixed**.
2. Click **Next** to display the Set Defaults panel, as shown in Figure 54.


New Contract From Data Set - Artix Designer

Binding Defaults
Binding Name: FixedBinding
Port Type Name: FixedPortType
Target Namespace: http://www.iona.com/FixedBinding
Schema Namespace: http://www.iona.com/FixedBinding/types

Message Defaults
[✓] Create Message Parts With Elements
Justification:
Encoding:
Padding:
*Justification, Encoding, and Padding may be overridden per message.

Figure 54: *New Contract from Data Set wizard - Set Fixed Defaults panel*

3. Under the **Binding Defaults**, enter a name for the binding being created in this new contract, or accept the default provided.

4. Enter a name for the new port type, or accept the default provided.
5. The **Target Namespace** and **Schema Namespace** values default to whatever is specified by the platform. Unless absolutely necessary, it is recommended that you do not change these.
6. Under the **Message Defaults,** check the box provided if you want to create your message parts as elements rather than types.
7. Select a justification value from the drop-down list. Options are **Left** and **Right**.
8. Enter an encoding value**.** Valid options are **UTF-8** and **UTF-16**.
9. Enter a value in the **Padding** field, if required. This is a character string to be used to fill unused space in the message field. You can use any character, or combination of characters, that you like.
10. Click **Next** to display the Input Data panel, as shown in Figure 55.

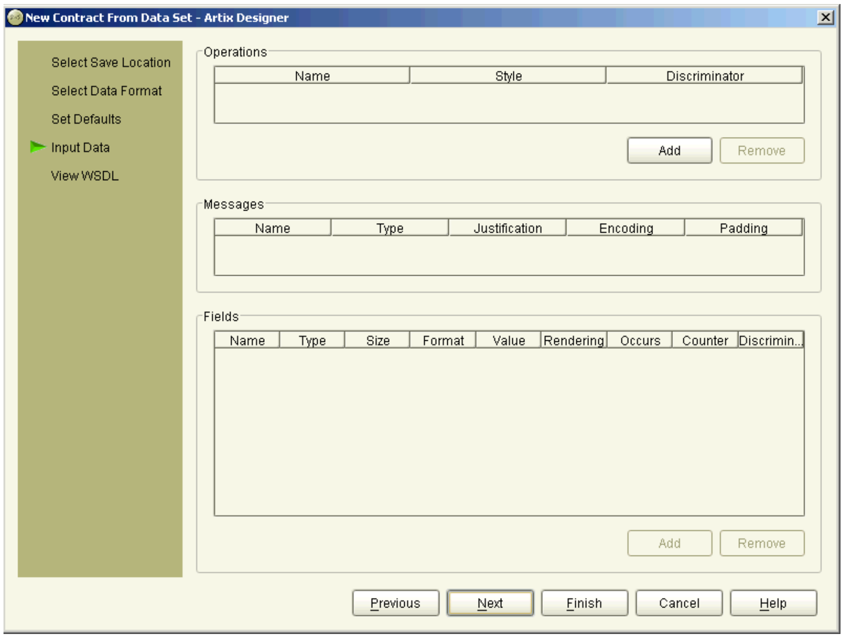


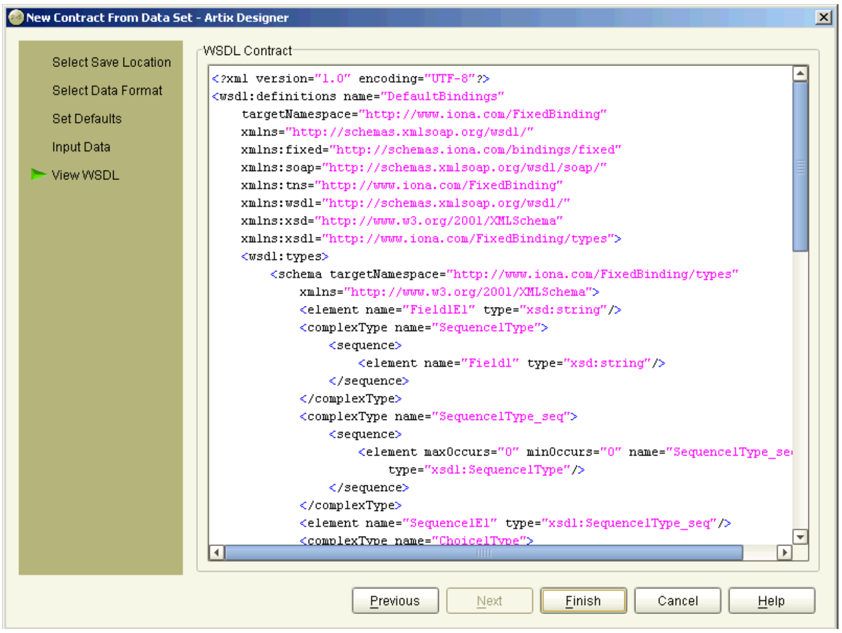
Figure 55: *New Contract from Data Set wizard - Input Data panel (Fixed)*

11. Click **Add** to create a new Operation.

12. Enter a name for the Operation, or accept the one provided.
13. Change the Operation **style** by clicking on the default Style value.
14. You can add a **discriminator** to filter the operations by adding one to the Discriminator cell for the new Operation.
15. Under **Messages** enter values for the attributes for the messages that have been created for the Operation.
16. Click **Add** to add fields to your messages and select each of the available cells to enter attributes for the fields as required.

    Click on the Type cell to change the field type. You can then add subsequent fields to the each of the field types.

    Message parts can be fields, enumerations, sequences, or choices.
17. When you have finished adding objects click **Finish** to create the contract with the fixed record binding, as shown in Figure 56.



```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="DefaultBindings"
    targetNamespace="http://www.iona.com/FixedBinding"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:fixed="http://schemas.iona.com/bindings/fixed"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:tns="http://www.iona.com/FixedBinding"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsd1="http://www.iona.com/FixedBinding/types">
    <wsdl:types>
        <schema targetNamespace="http://www.iona.com/FixedBinding/types"
            xmlns="http://www.w3.org/2001/XMLSchema">
            <element name="Field1El" type="xsd:string"/>
            <complexType name="Sequence1Type">
                <sequence>
                    <element name="Field1" type="xsd:string"/>
                </sequence>
            </complexType>
            <complexType name="Sequence1Type_seq">
                <sequence>
                    <element maxOccurs="0" minOccurs="0" name="Sequence1Type_se
                        type="xsd1:Sequence1Type"/>
                </sequence>
            </complexType>
            <element name="Sequence1El" type="xsd1:Sequence1Type_seq"/>
            <complexType name="Choice1Type">
```

Figure 56: *New Contract from Data Set - Summary panel*

Creating a Contract Containing a Fixed Binding from a COBOL Copybook

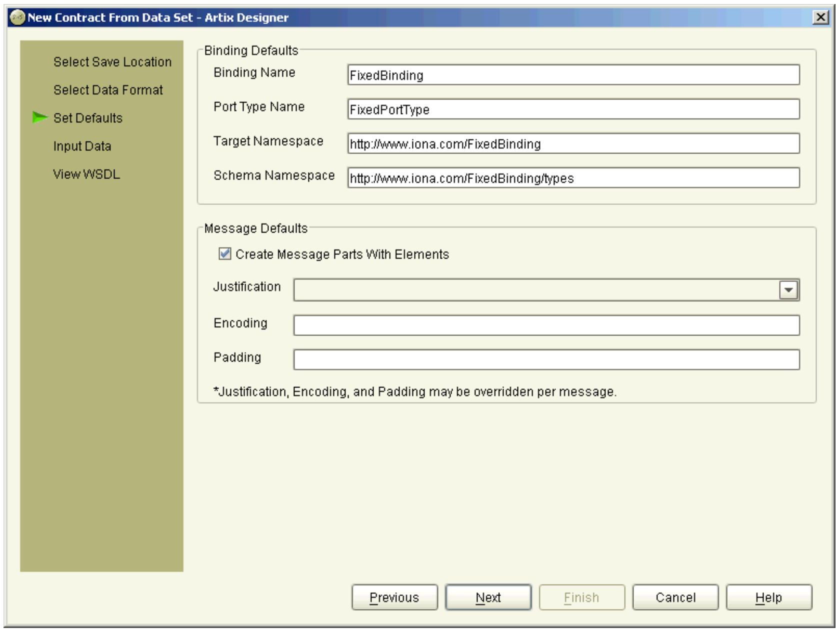
Overview

The other way to create a contract containing a fixed binding is to base the messages in that binding on an existing COBOL Copybook. Your CCB can contain one or more messages - at the time that you associate each fixed message with a message from the CCB, you'll be asked to specify the message to use.

Procedure

To add a contract containing a Fixed binding from a COBOL Copybook:

1. At the Data Format panel, select **Fixed**.
2. Click **Next** to display the Set Defaults panel, as shown in Figure 57.



Binding Defaults

Binding Name: FixedBinding
Port Type Name: FixedPortType
Target Namespace: http://www.iona.com/FixedBinding
Schema Namespace: http://www.iona.com/FixedBinding/types

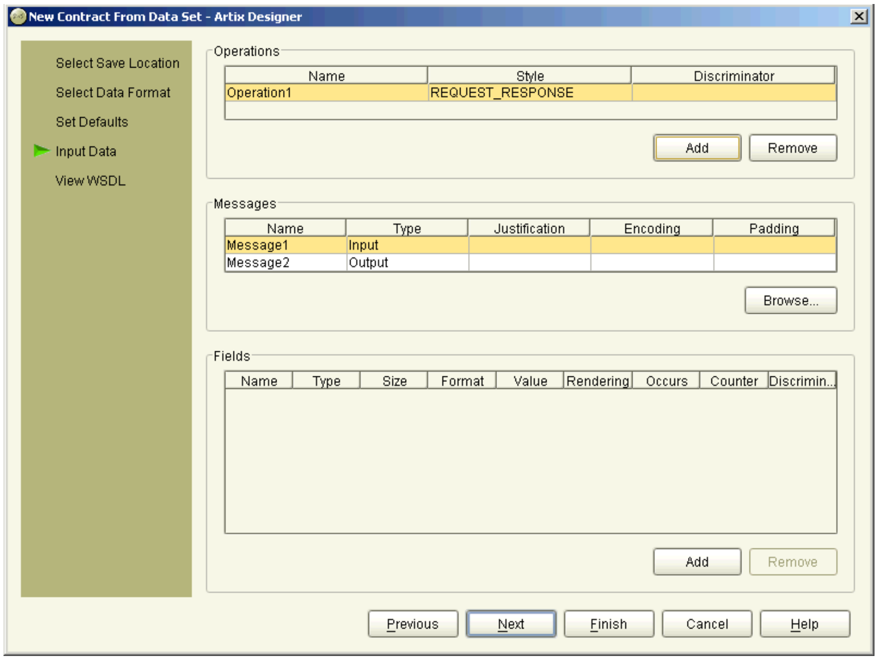
Message Defaults

☑ Create Message Parts With Elements
Justification:
Encoding:
Padding:

*Justification, Encoding, and Padding may be overridden per message.

Figure 57: *New Contract from Data Set wizard - Set Fixed Defaults (CCB)*

3. Under the **Binding Defaults**, enter a name for the binding being created in this new contract, or accept the default provided.
4. Enter a name for the new port type, or accept the default provided.
5. The **Target Namespace** and **Schema Namespace** values default to whatever is specified by the platform. Unless absolutely necessary, it is recommended that you do not change these.
6. Under the **Message Defaults,** check the box provided if you want to create your message parts as elements rather than types.
7. Select a justification value from the drop-down list. Values are **Left** and **Right**.
8. Enter an encoding value**.** Valid options are **UTF-8** and **UTF-16**.
9. Enter a value in the **Padding** field, if required. This is any character string to be used to fill unused space in the message field.
10. Click Next to display the Input Data panel, as shown in Figure 55.



Name	Style	Discriminator
Operation1	REQUEST_RESPONSE	

Name	Type	Justification	Encoding	Padding
Message1	Input			
Message2	Output			

Name	Type	Size	Format	Value	Rendering	Occurs	Counter	Discrimin...

Figure 58: *New Contract from Data Set wizard - Input Data panel (CCB)*

11. Click **Add** to create a new Operation. (In the example shown, this step has already been performed so that the Browse button described in step 15 could be enabled.)
12. Enter a name for the Operation, or accept the one provided.
13. Change the Operation **style** by clicking on the default Style value.
14. You can add a **discriminator** to filter the operations by adding one of the Discriminator cell for the new Operation.
15. Under **Messages** enter values for the attributes for the messages that have been created for the Operation. To use the details from your COBOL Copybook, click the **Browse** button.

    This will invoke a file chooser, from where you can navigate to your COBOL Copybook. When you select one, and click OK, Artix will do one of two things:

    - If the COBOL Copybook contains only one message, the associated fields on the Input Data panel will be populated.
    - If the COBOL Copybook contains more that one message, you will see an intermediary dialog from where you can select which message to associate with the fixed message in the Input Data panel.
16. You can edit any of the fields that are populated for this message from the COBOL Copybook by clicking on the relevant cell.
17. Click **Add** to add extra fields to your messages as required.

    Click on the Type cell to change the field type. You can then add subsequent fields to the each of the field types.

    Each message part can be either a field, an enumeration, a sequence, or a choice.
18. When you have finished adding objects click **Finish** to create the contract with the fixed record binding, as shown in Figure 56.

Creating a Contract Containing a Tagged Binding

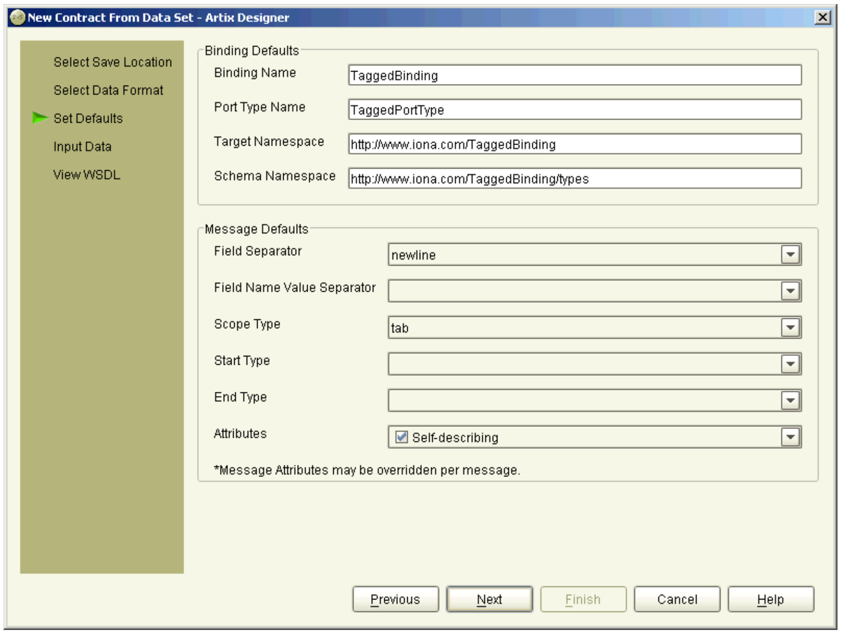
Overview

The tagged data format supports applications that use self-describing, or delimited, messages to communicate. Artix can read tagged data and write it out in any supported data format. Similarly, Artix is capable of converting a message from any of its supported data formats into a self-describing or tagged data message.

Procedure

To add a contract containing a Tagged binding:

1. At the Data Format panel, select **Tagged**.
2. Click **Next** to display the Set Defaults panel, as shown in Figure 59.



Binding Defaults

Binding Name	TaggedBinding
Port Type Name	TaggedPortType
Target Namespace	http://www.iona.com/TaggedBinding
Schema Namespace	http://www.iona.com/TaggedBinding/types

Message Defaults

Field Separator	newline
Field Name Value Separator	
Scope Type	tab
Start Type	
End Type	
Attributes	✓ Self-describing

*Message Attributes may be overridden per message.

Figure 59: *New Contract from Data Set - Set Tagged Defaults panel*

3. Under the **Binding Defaults**, enter a name for the binding being created in this new contract, or accept the default provided.

4. Enter a name for the new port type, or accept the default provided.
5. The **Target Namespace** and **Schema Namespace** values default to whatever is specified by the platform. Unless absolutely necessary, it is recommended that you do not change these.
6. Under the **Message Defaults**, select a value for the **Field Separator**, or accept the default provided.
7. Select a value for the **Field Name Value Separator**.
8. Select a value for the **Scope Type**, or accept the default provided.
9. Select a value for the **Start Type**.
10. Select a value for the **End Type**.
11. Select the **Attributes** you want to apply as defaults to your messages.
12. Click **Next** to display the Input Data panel, as shown in Figure 60.

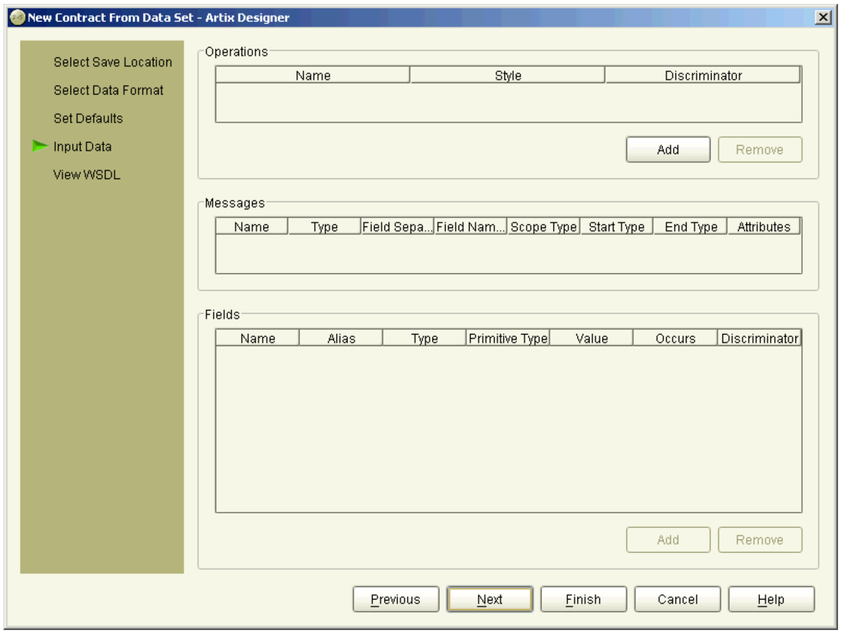
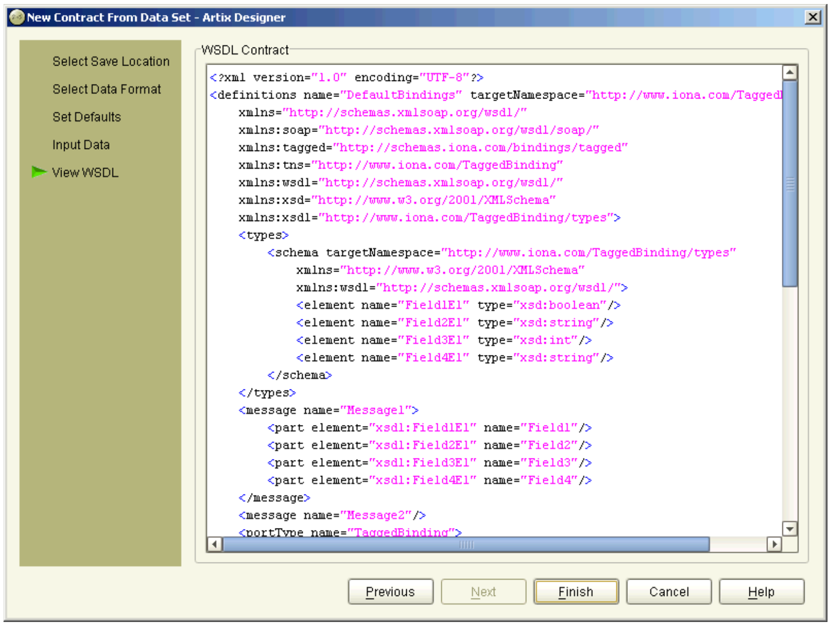


Figure 60: *New Contract from Data Set wizard - Input Data panel (Tagged)*

13. Click **Add** to create a new Operation.

14. Enter a name for the Operation, or accept the one provided.
15. Change the Operation **style** by clicking on the default Style value.
16. You can add a **discriminator** to filter the operations by adding one to the Discriminator cell for the new Operation.
17. Under **Messages** enter values for the attributes for the messages that have been created for the Operation. The values you specified on the Defaults panel are displayed here, but can be over-written at the individual message level if required.
18. Click **Add** to add fields to your messages and select each of the available cells to enter attributes for the fields as required.

    Click on the Type cell to change the field type. You can then add subsequent fields to the each of the field types. Messages can be a field, an enumeration, a sequence, or a choice.
19. When you have finished adding objects click **Finish** to create the contract with the tagged record binding, as shown in Figure 61.



```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="DefaultBindings" targetNamespace="http://www.iona.com/TaggedI
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:tagged="http://schemas.iona.com/bindings/tagged"
    xmlns:tns="http://www.iona.com/TaggedBinding"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsdl="http://www.iona.com/TaggedBinding/types">
    <types>
        <schema targetNamespace="http://www.iona.com/TaggedBinding/types"
            xmlns="http://www.w3.org/2001/XMLSchema"
            xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
            <element name="Field1El" type="xsd:boolean"/>
            <element name="Field2El" type="xsd:string"/>
            <element name="Field3El" type="xsd:int"/>
            <element name="Field4El" type="xsd:string"/>
        </schema>
    </types>
    <message name="Message1">
        <part element="xsdl:Field1El" name="Field1"/>
        <part element="xsdl:Field2El" name="Field2"/>
        <part element="xsdl:Field3El" name="Field3"/>
        <part element="xsdl:Field4El" name="Field4"/>
    </message>
    <message name="Message2"/>
    <portType name="TaggedBinding">
```

Figure 61: *New Contract from Data Set - Summary panel (Tagged)*

Editing Contracts

Overview

The Artix Designer provides edit dialogs for all of the contract components. From these dialogs you can edit most of the properties for your contract. This section walks you through that process.

You can access the edit dialog for a contract component either through the menu bar or through the contract navigator (graph view).

To access the edit dialog for one of the components:

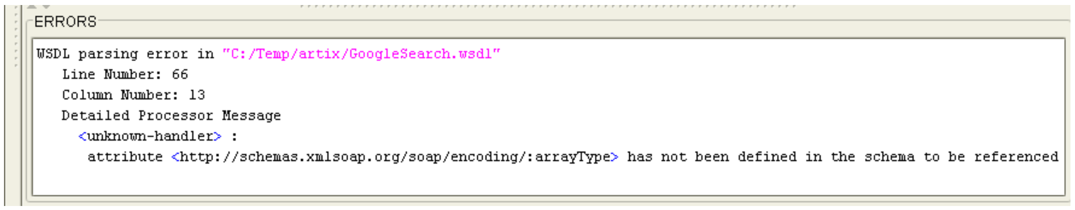
1. While in the graph view of the **Contract Navigator**, select **Contract | Edit | <component>**, where <component> is the name of the contract element you want to work with.

   The Edit dialog for that component is displayed.
2. Alternatively, you can right-click on the component name and select **Edit**, which will also display the Edit dialog.

Editing in the WSDL view

If you prefer, you can use the WSDL view of the contract to hand-edit the WSDL. Be aware however, that any changes you make to the WSDL could invalidate the contract. If this happens, you will only be able to view the contract in the WSDL view - the graph view will be disabled as the model cannot be generated with invalid WSDL.

The Artix Designer provides you with tools to try to help you avoid invalidating your WSDL, or to identify and rectify WSDL errors. Every time you make a change to the WSDL and click Apply Edits, the Designer displays any WSDL errors in the error bar at the bottom of the WSDL view, as shown in Figure 62.

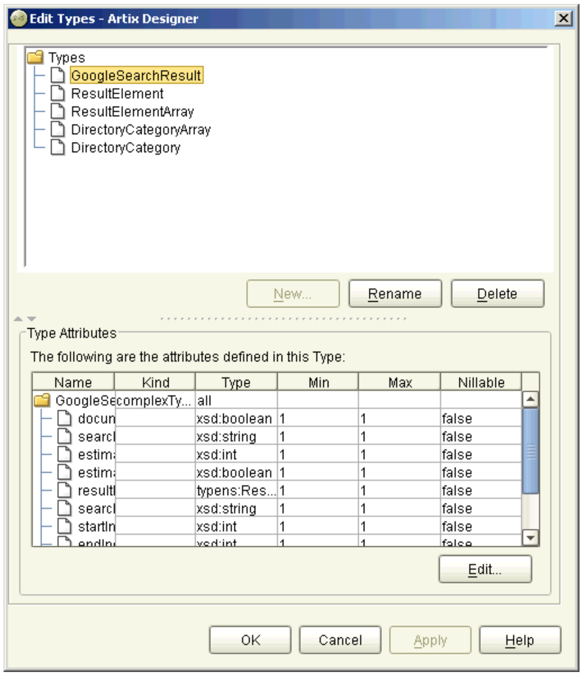


```
WSDL parsing error in "C:/Temp/artix/GoogleSearch.wsdl"
   Line Number: 66
   Column Number: 13
   Detailed Processor Message
     <unknown-handler> :
      attribute <http://schemas.xmlsoap.org/soap/encoding/:arrayType> has not been defined in the schema to be referenced
```

Figure 62: *WSDL Error Indicator*

Editing Types

You can edit a type by selecting **Contract | Edit | Types**, to display the Edit Types dialog as shown in Figure 63.



Types
- GoogleSearchResult
- ResultElement
- ResultElementArray
- DirectoryCategoryArray
- DirectoryCategory

The following are the attributes defined in this Type:

Name	Kind	Type	Min	Max	Nillable
GoogleSe	complexTy...	all			
docun		xsd:boolean	1	1	false
searcl		xsd:string	1	1	false
estim:		xsd:int	1	1	false
estim:		xsd:boolean	1	1	false
resultl		typens:Res...	1	1	false
searcl		xsd:string	1	1	false
startIn		xsd:int	1	1	false
endIn		xsd:int	1	1	false

Figure 63: *Edit Types dialog*

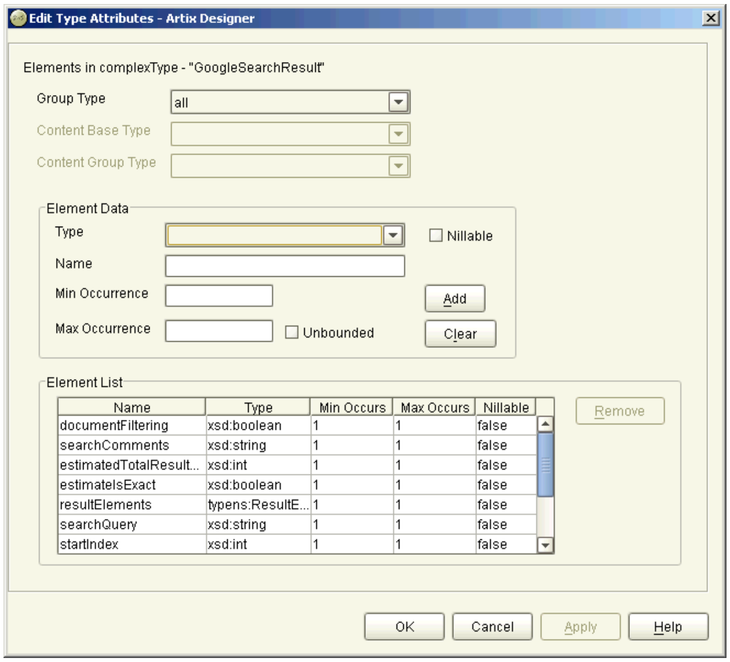
At the Edit Types dialog, all of your types and their associated attributes are listed in the top half of the dialog. From here you can:

- Rename a type or an attribute by selecting it and clicking **Rename**
- Delete a type or an attribute by selecting it and clicking **Delete**.
- Add a new type by clicking **Add** to display the New Type wizard, as described in "Adding Types" on page 48.

Editing attribute properties

When you select a type in the top of this dialog, the type attributes are displayed in the panel at the bottom of the dialog.

To edit any of the Type Attributes, click the **Edit** button to display the Edit Type Attributes dialog, as shown in Figure 64.



Elements in complexType - "GoogleSearchResult"

Name	Type	Min Occurs	Max Occurs	Nillable
documentFiltering	xsd:boolean	1	1	false
searchComments	xsd:string	1	1	false
estimatedTotalResult...	xsd:int	1	1	false
estimateIsExact	xsd:boolean	1	1	false
resultElements	typens:ResultE...	1	1	false
searchQuery	xsd:string	1	1	false
startIndex	xsd:int	1	1	false

Figure 64: *Edit Type Attributes dialog*

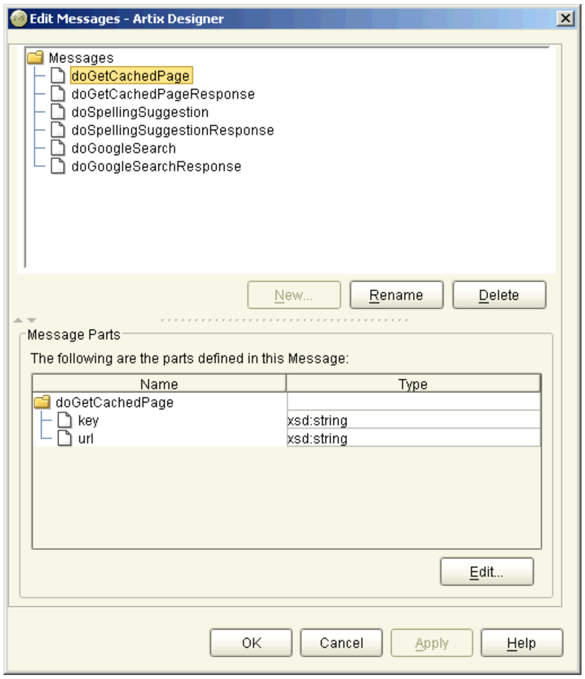
To change values of attributes in this dialog, click on the item you want to change in the Element List - its details will be populated into the Element Data fields. Make your changes and click **Update**.

When you have finished making your changes, click **Apply** to update the attribute, and **OK** to close the wizard and return to the Edit Types dialog, where your changes are displayed in the Type Attributes panel.

Click **OK** to close this dialog and return to the Artix Designer.

Editing Messages

You can edit a type by selecting **Contract | Edit | Messages**, to display the Edit Messages dialog as shown in Figure 65.



Name	Type
doGetCachedPage	
key	xsd:string
url	xsd:string

Figure 65: *Edit Messages dialog*

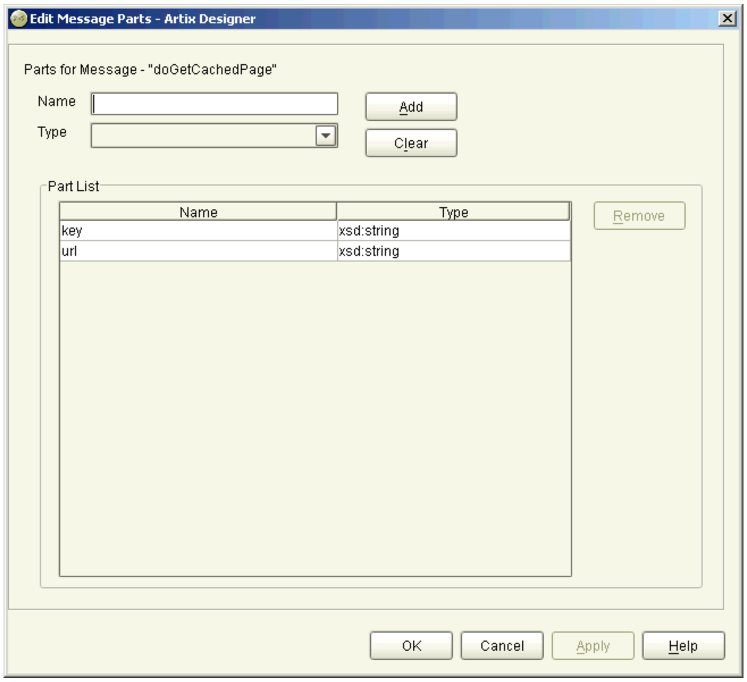
At the Edit Messages dialog, all of your messages and their associated parts are listed in the top half of the dialog. From here you can:

- Rename a message or a part by selecting it and clicking **Rename**
- Delete a message or a part by selecting it and clicking **Delete**.
- Add a new message by clicking **Add** to display the New Message wizard, as described in “Adding Messages” on page 57.

Editing message parts

When you select a message in the top of this dialog, the message parts are displayed in the panel at the bottom of the dialog.

To edit any of the message parts, click the **Edit** button to display the Edit Message Parts dialog, as shown in Figure 64.



Parts for Message - "doGetCachedPage"

Name	Type
key	xsd:string
url	xsd:string

Figure 66: *Edit Message Parts dialog*

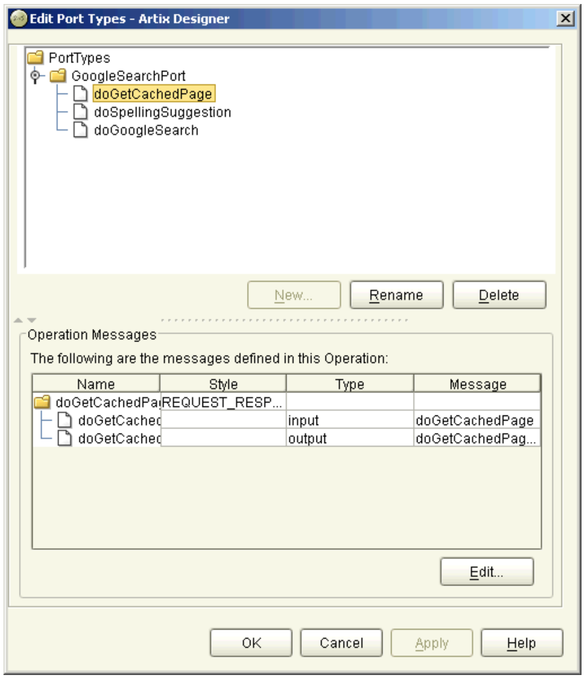
To change values of parts in this dialog, click on the item you want to change in the Part List - its details will be populated into the Parts fields. Make your changes and click **Update**.

When you have finished making your changes, click **Apply** to update the part, and **OK** to close the wizard and return to the Edit Messages dialog, where your changes are displayed in the Message Parts panel.

Click **OK** to close this dialog and return to the Artix Designer.

Editing Port Types

You can edit a port type by selecting **Contract | Edit | Port Types**, to display the Edit Port Types dialog as shown in Figure 67.



The following are the messages defined in this Operation:

Name	Style	Type	Message
doGetCachedPa	REQUEST_RESP...		
doGetCached		input	doGetCachedPage
doGetCached		output	doGetCachedPag...

Figure 67: *Edit Port Types dialog*

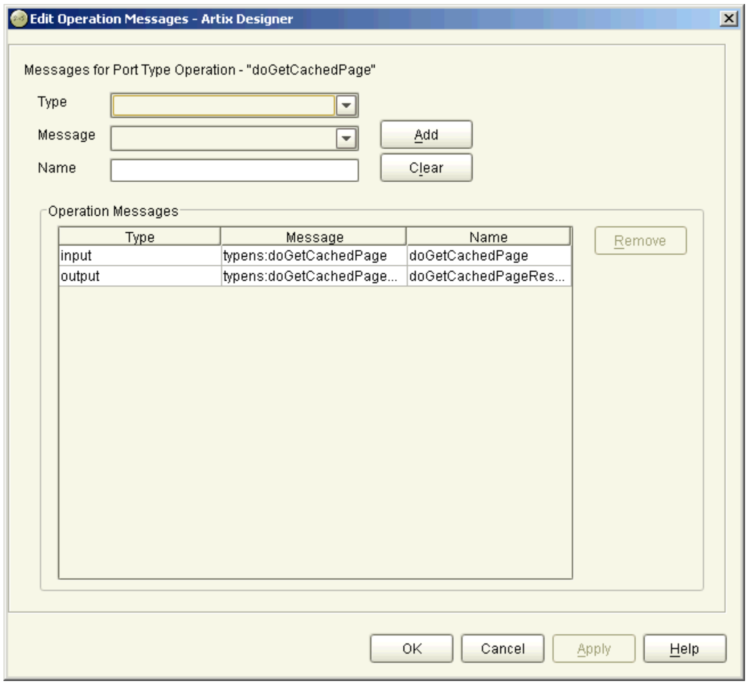
At the Edit Port Types dialog, all of your port types and their associated operation messages are listed in the top half of the dialog. From here you can:

- Rename a port type or an operation message by selecting it and clicking **Rename**
- Delete a port type or an operation message by selecting it and clicking **Delete**.
- Add a new port type by clicking **Add** to display the New Port Type wizard, as described in "Adding Port Types" on page 61.

Editing operation messages

When you select a port type in the top of this dialog, the operation messages are displayed in the panel at the bottom of the dialog.

To edit any of the operation messages, click the **Edit** button to display the Edit Operation Messages dialog, as shown in Figure 68.



Messages for Port Type Operation - "doGetCachedPage"

Type	Message	Name
input	typens:doGetCachedPage	doGetCachedPage
output	typens:doGetCachedPage...	doGetCachedPageRes...

Figure 68: *Edit Type Attributes dialog*

To change values of operation messages in this dialog, click on the item you want to change in the Operation Messages list - its details will be populated into the Messages fields. Make your changes and click **Update**.

When you have finished making your changes, click **Apply** to update the operation message, and **OK** to close the wizard and return to the Edit Port Types dialog, where your changes are displayed in the Operation Messages panel.

Click **OK** to close this dialog and return to the Artix Designer.

CHAPTER 5

Adding Bindings

Bindings contain information used by Artix at runtime to reformat data between endpoints, enabling it to be understood by the target service.

In this chapter

This chapter discusses the following topics:

What is a Binding?

Overview

If you are exposing an existing service using a new transport or payload format, you need to add the mapping of the service's data and operations to the new payload format and transport. To do this, you add one or more bindings to your services. The information you include in the binding is used by Artix at runtime to reformat the data on the wire and thus make it understandable by the target service.

The New Binding wizard walks you through the generation of a binding based on your existing contract. It then adds the binding to the contract.

Artix binding types

Artix provides support for several binding types. They are accessed via two methods:

- From the New Binding wizard, which enables you to create the following binding types:
    - CORBA
    - SOAP
    - XML
- From the Contract From Data Set wizard, which enables you to create a new contract that also includes a binding of one of the following types:
    - Fixed
    - Fixed, using data from an existing COBOL Copybook
    - Tagged

  These last three bindings can be created with or without an existing contract, and will create the binding and logical elements from input data.

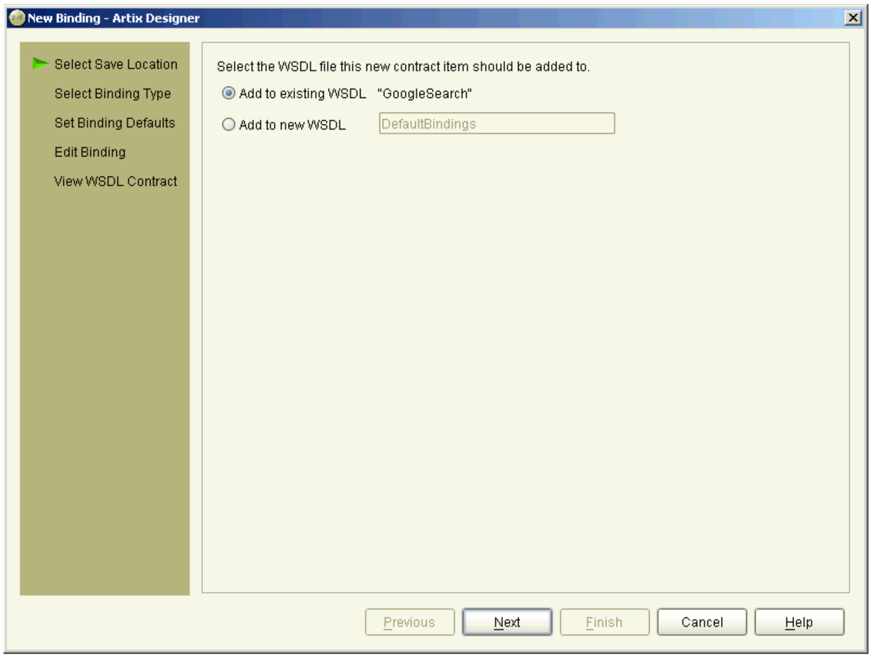
  For more information about adding contracts with these bindings included, "Creating Contracts from Data Sets" on page 74.

Adding bindings via New Binding wizard

To add a binding to an Artix contract using the New Binding wizard:

1. From the Designer Tree, select the contract to which you want to add the binding.
2. Select **Contract | New | Binding** from the menu bar to display the New Binding wizard, as shown in Figure 69.

   Note that your WSDL needs to contain at least one message and a port type before you can add a binding.



Select the WSDL file this new contract item should be added to.

(●) Add to existing WSDL "GoogleSearch"

( ) Add to new WSDL DefaultBindings

Figure 69: *New Binding wizard*

3. Select where to create the WSDL entry for the new binding.
   - **Add to existing WSDL** adds the binding information to the existing contract.

- **Add to new WSDL** creates a new WSDL document that contains the binding information plus an import statement in the logical contract in which the binding is being created.

4. Click **Next** to display the Binding Type panel.

Now you need to turn to the appropriate page for the type of binding you are creating:

- For a CORBA binding, see page 97
- For a SOAP binding, see page 100
- For an XML binding, see page 104

Adding a CORBA Binding

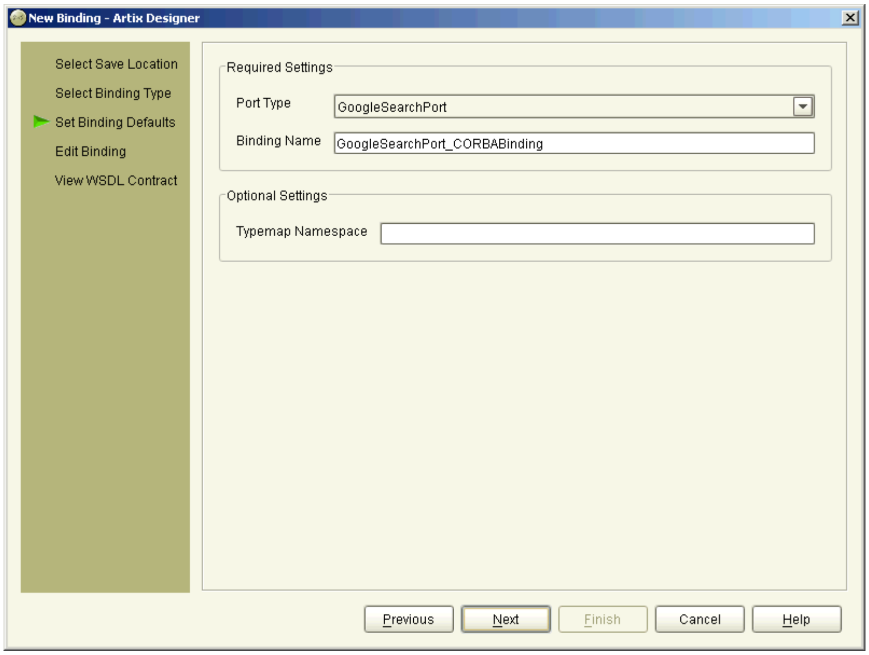
Overview

To ensure that messages are converted into a format that a CORBA application can understand, Artix contracts need to describe how data is mapped to CORBA data types.

Procedure

To add a CORBA binding to an Artix contract from the Binding Type panel:

1. Select **CORBA**, and then click **Next** to display the Binding Defaults panel, as shown in Figure 70.

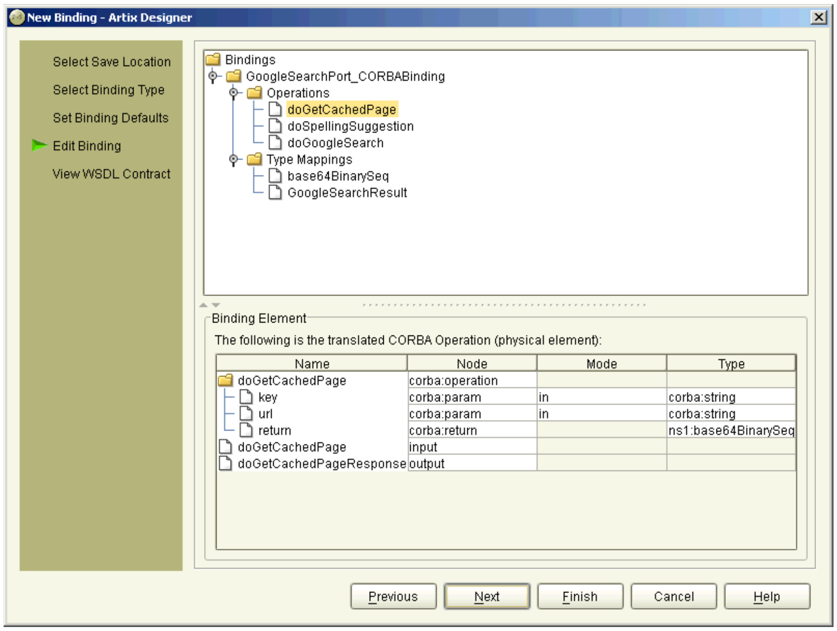


Port Type: GoogleSearchPort
Binding Name: GoogleSearchPort_CORBABinding
Typemap Namespace:

Figure 70: *New Binding wizard - CORBA Binding Defaults panel*

2. From the **Port Type** drop down list select the port type you want to map to the CORBA binding.

3. Enter a name for the new binding, or accept the default provided.
4. Enter a value for the Typemap Namespace if required (optional).
5. Click **Next** to display the Edit Binding panel, as shown in Figure 71, which displays the generated operations and CORBA types.



```
Bindings
  GoogleSearchPort_CORBABinding
    Operations
      doGetCachedPage
      doSpellingSuggestion
      doGoogleSearch
    Type Mappings
      base64BinarySeq
      GoogleSearchResult
```

Binding Element

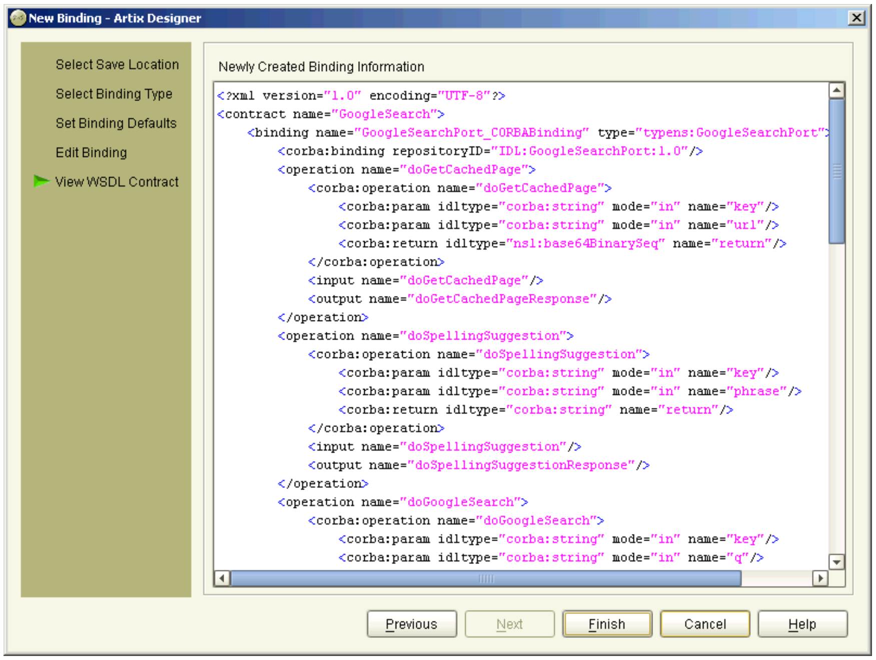
The following is the translated CORBA Operation (physical element):

Name	Node	Mode	Type
doGetCachedPage	corba:operation		
key	corba:param	in	corba:string
url	corba:param	in	corba:string
return	corba:return		ns1:base64BinarySeq
doGetCachedPage	input		
doGetCachedPageResponse	output		

Figure 71: *New Binding wizard - Edit CORBA Binding panel*

6. Examine the different elements of the binding by selecting them from the tree at the top of the dialog.
7. If you like, you can change the name of the Binding. The attribute fields are read-only.

8. Click **Next** to display the Summary panel, as shown in Figure 72.



```
<?xml version="1.0" encoding="UTF-8"?>
<contract name="GoogleSearch">
    <binding name="GoogleSearchPort_CORBABinding" type="typens:GoogleSearchPort">
        <corba:binding repositoryID="IDL:GoogleSearchPort:1.0"/>
        <operation name="doGetCachedPage">
            <corba:operation name="doGetCachedPage">
                <corba:param idltype="corba:string" mode="in" name="key"/>
                <corba:param idltype="corba:string" mode="in" name="url"/>
                <corba:return idltype="ns1:base64BinarySeq" name="return"/>
            </corba:operation>
            <input name="doGetCachedPage"/>
            <output name="doGetCachedPageResponse"/>
        </operation>
        <operation name="doSpellingSuggestion">
            <corba:operation name="doSpellingSuggestion">
                <corba:param idltype="corba:string" mode="in" name="key"/>
                <corba:param idltype="corba:string" mode="in" name="phrase"/>
                <corba:return idltype="corba:string" name="return"/>
            </corba:operation>
            <input name="doSpellingSuggestion"/>
            <output name="doSpellingSuggestionResponse"/>
        </operation>
        <operation name="doGoogleSearch">
            <corba:operation name="doGoogleSearch">
                <corba:param idltype="corba:string" mode="in" name="key"/>
                <corba:param idltype="corba:string" mode="in" name="q"/>
```

Figure 72: *New Binding wizard - CORBA Binding Summary panel*

9. Click **Finish** to close this wizard and return to the Artix Designer.

When you have created the new CORBA binding, the contract describing the binding and the CORBA type map are added to the Designer Tree under the selected service. Note however, that this new contract **will not** contain a CORBA port description.

For details on adding a CORBA port description see "Adding a CORBA Port" on page 113.

Adding a SOAP Binding

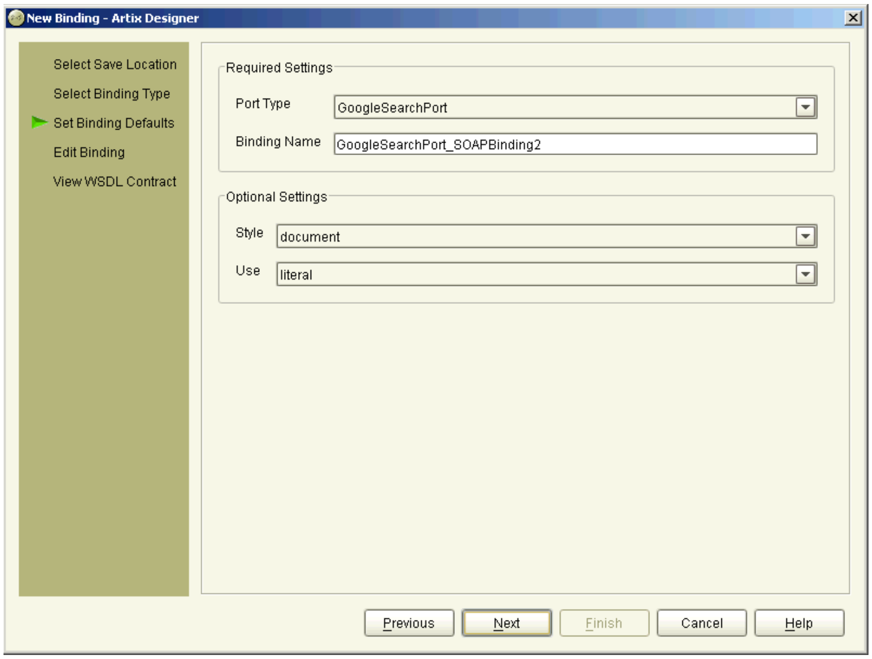
Overview

SOAP is termed a *messaging* protocol. It is a framework for transporting client request and server response messages in the form of XML documents over (usually) the HTTP transport.

Procedure

To add a SOAP binding to an Artix contract:

1. At the Binding Type panel, select **SOAP**.
2. Click **Next** to display the Binding Defaults panel, as shown in Figure 73.



Required Settings

Port Type: GoogleSearchPort

Binding Name: GoogleSearchPort_SOAPBinding2

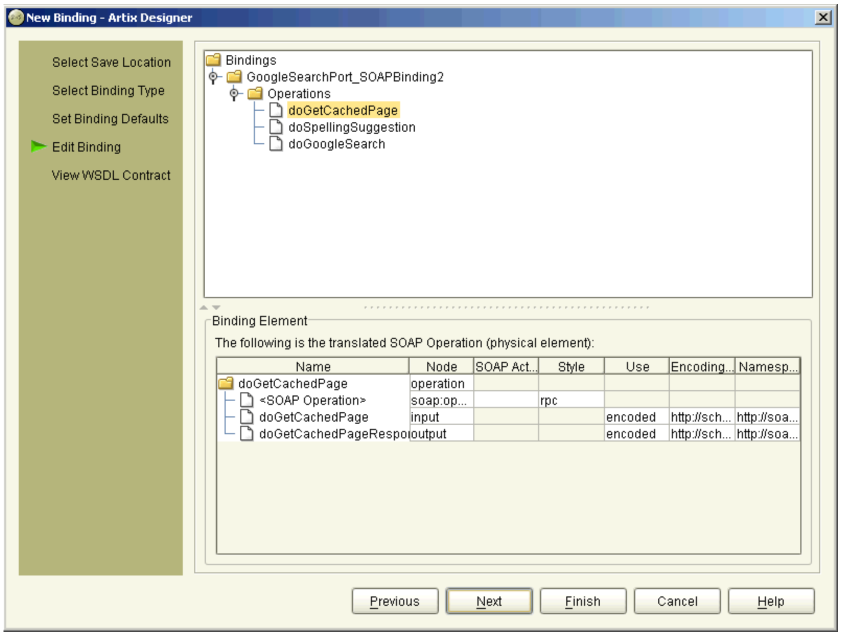
Optional Settings

Style: document

Use: literal

Figure 73: *New Binding wizard - SOAP Binding Defaults panel*

3. From the **Port Type** drop down list, select the port type that the binding relates to.
4. Enter a name for the new binding, or accept the default provided.
5. From the **Style** drop down list, select either **rpc** or **document**, to indicate whether message parts pertaining to each operation are to consist of RPC-based parameters and return values or document-based body entries by default. The value you choose is subsequently populated in the `soap:binding style` attribute in your WSDL contract.
6. From the **Use** drop down list, select either **encoded** or **literal**, to indicate whether message parts are to consist of abstract type definitions or concrete schema definitions. The value you choose is subsequently populated in the `soap:body use` attribute in your WSDL contract.
7. Click **Next** to display the Edit Binding panel, as shown in Figure 74.



Bindings
- GoogleSearchPort_SOAPBinding2
  - Operations
    - doGetCachedPage
    - doSpellingSuggestion
    - doGoogleSearch

Binding Element

The following is the translated SOAP Operation (physical element):

Name	Node	SOAP Act...	Style	Use	Encoding...	Namesp...
doGetCachedPage	operation					
<SOAP Operation>	soap:op...		rpc			
doGetCachedPage	input			encoded	http://sch...	http://soa...
doGetCachedPageRespo	output			encoded	http://sch...	http://soa...

Figure 74: *New Binding wizard - Edit SOAP Binding panel*

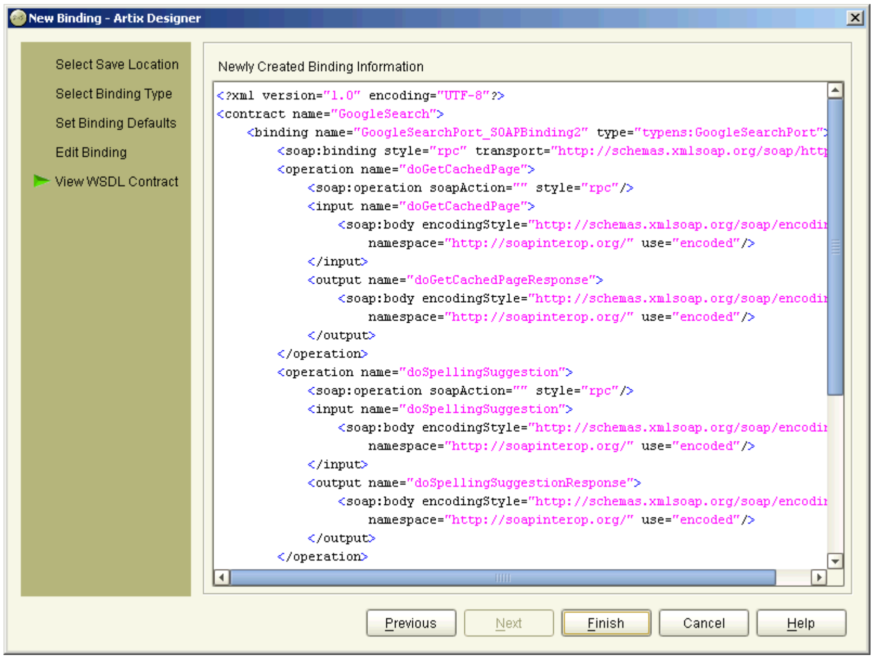
8. Click on the name of an operation within your binding.
9. If you want to include a SOAPAction field in the HTTP header of a SOAP message, use the **SOAP Action** cell in the Binding Elements table to specify the URL that represents the resource being requested by the operation.

   Note: This step only relates to the use of SOAP over HTTP, but it is not mandatory for the purposes of Artix. It is available in case some third-party SOAP servers that do use a SOAPAction field in their HTTP headers are to be contacted.

10. If you want to override the default setting for **Style** that you set in **step 5**, click on the **Style** cell and select another value.
11. If you want to override the default setting for **Use** that you set in **step 6**, click on the **Use** cell and select another value.
12. If you want to use other customized encoding styles, add the URL(s) relating to each style to the relevant field(s) in the Encoding Style column. (**Note**: Only possible where Use=Encoded).

    Note: If you want this field to contain more than one URL, ensure that they are separated by spaces, and ordered according to the most restrictive set of rules first and least restrictive set of rules last.

13. Click **Next** to display the Summary panel, as shown in Figure 75.



```
<?xml version="1.0" encoding="UTF-8"?>
<contract name="GoogleSearch">
    <binding name="GoogleSearchPort_SOAPBinding2" type="typens:GoogleSearchPort">
        <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http
        <operation name="doGetCachedPage">
            <soap:operation soapAction="" style="rpc"/>
            <input name="doGetCachedPage">
                <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encodin
                    namespace="http://soapinterop.org/" use="encoded"/>
            </input>
            <output name="doGetCachedPageResponse">
                <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encodin
                    namespace="http://soapinterop.org/" use="encoded"/>
            </output>
        </operation>
        <operation name="doSpellingSuggestion">
            <soap:operation soapAction="" style="rpc"/>
            <input name="doSpellingSuggestion">
                <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encodin
                    namespace="http://soapinterop.org/" use="encoded"/>
            </input>
            <output name="doSpellingSuggestionResponse">
                <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encodin
                    namespace="http://soapinterop.org/" use="encoded"/>
            </output>
        </operation>
```

Figure 75: *New Binding wizard - SOAP Binding Summary panel*

14. Click **Finish** to close this wizard and return to the Artix Designer.

Adding an XML Binding

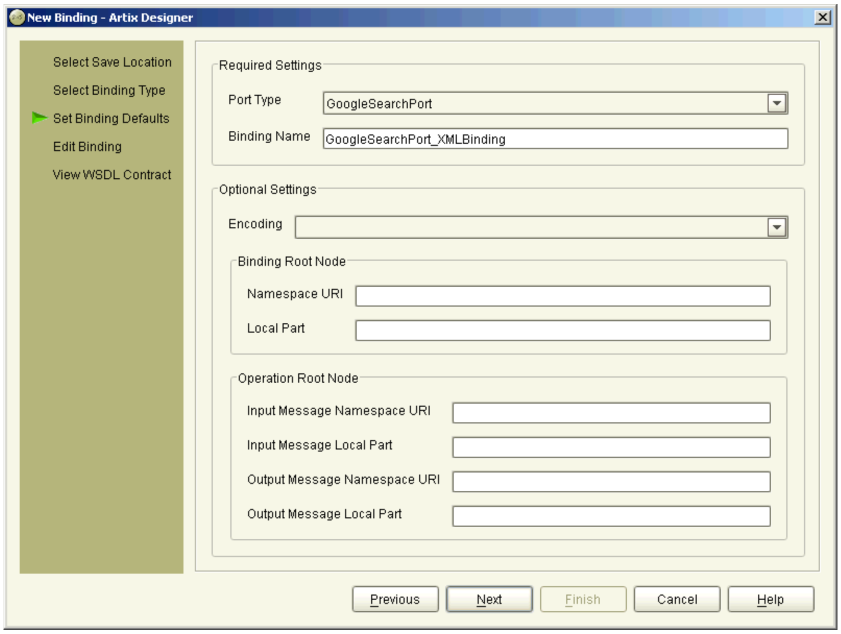
Overview

The pure XML payload format provides an alternative to the SOAP binding by allowing services to exchange data using straight XML documents without needing the overhead of the SOAP envelope.

Procedure

To add an XML binding to an Artix contract:

1. At the Binding Type panel, select **XML** and click **Next** to display the Binding Defaults panel, as shown in Figure 76.



Port Type: GoogleSearchPort

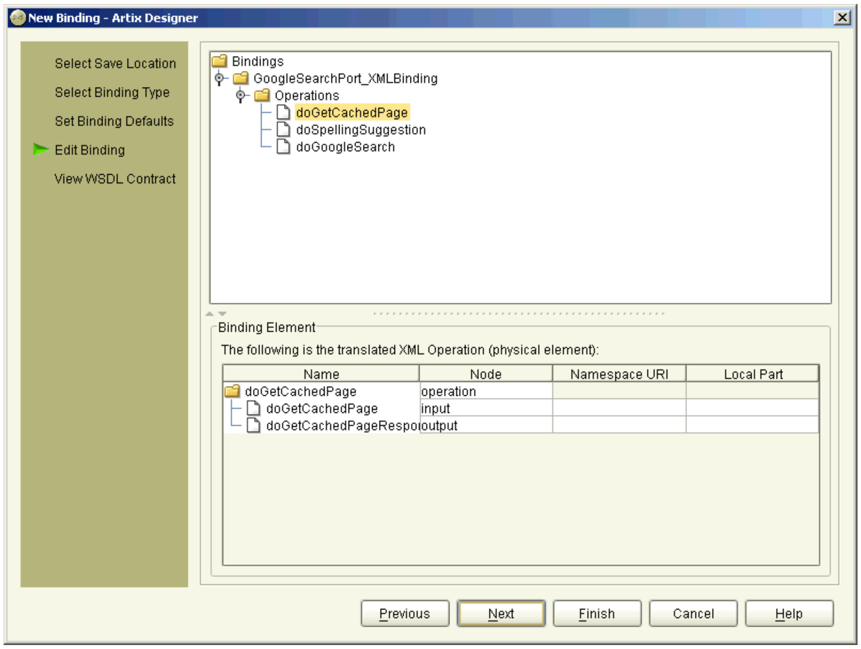
Binding Name: GoogleSearchPort_XMLBinding

Figure 76: *New Binding wizard - XML Binding Defaults panel*

2. From the **Port Type** drop down list select the Port Type you want to map to the XML binding.
3. Enter a name for the new binding, or accept the default provided.

4. Under the **Optional Settings**, select an **Encoding** value.
5. Enter values in the Binding Route Node section. This is the *Qname* for the binding. This is a unique identifier made up of two parts:
   - Namespace URI - the location of the binding element
   - Local Part - any name you wish to append to the binding element
6. Enter values in the Operation Root Node section. This is the *Qname* at the operation level. This is a unique identifier, again made up of two parts but this time there will be two parts for each message, i.e. input and output, or just input for one-way messages:
   - Namespace URI - the location of the binding element
   - Local Part - any name you wish to append to the binding element

   If you do not specify these values at the Operation level, the Binding Route Node is used by default.
7. Click **Next** to display the Edit Binding panel, as shown in Figure 77.



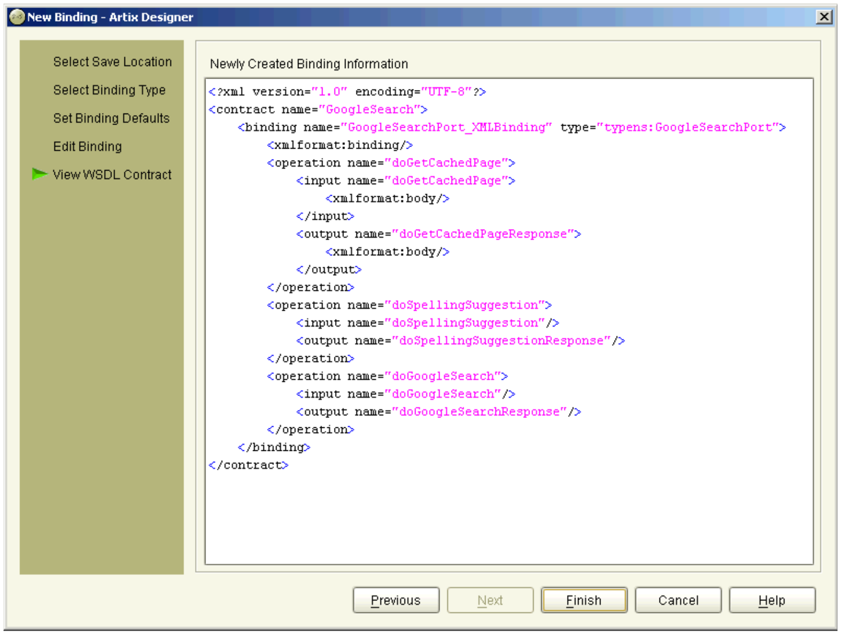
Binding Element

The following is the translated XML Operation (physical element):

Name	Node	Namespace URI	Local Part
doGetCachedPage	operation		
doGetCachedPage	input		
doGetCachedPageRespo	output		

Figure 77: *New Binding wizard - Edit XML Binding panel*

8. Examine the different operations of the binding by selecting them from the tree at the top of the dialog.
9. Edit the **Namespace URI** and **Local Part** values shown in the Binding Element table, or accept the defaults provided.
10. Click **Next** to display the WSDL Contract panel, as shown in Figure 78.



Newly Created Binding Information

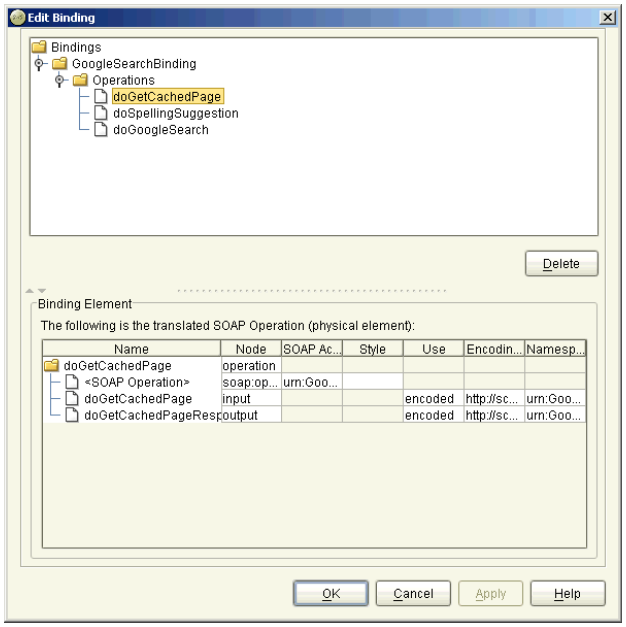
```
<?xml version="1.0" encoding="UTF-8"?>
<contract name="GoogleSearch">
    <binding name="GoogleSearchPort_XMLBinding" type="typens:GoogleSearchPort">
        <xmlformat:binding/>
        <operation name="doGetCachedPage">
            <input name="doGetCachedPage">
                <xmlformat:body/>
            </input>
            <output name="doGetCachedPageResponse">
                <xmlformat:body/>
            </output>
        </operation>
        <operation name="doSpellingSuggestion">
            <input name="doSpellingSuggestion"/>
            <output name="doSpellingSuggestionResponse"/>
        </operation>
        <operation name="doGoogleSearch">
            <input name="doGoogleSearch"/>
            <output name="doGoogleSearchResponse"/>
        </operation>
    </binding>
</contract>
```

Figure 78: *Binding wizard - XML Binding WSDL Contract panel*

11. Click **Finish** to close this wizard and return to the Artix Designer.

Editing Bindings

You can edit a binding by selecting it in the Contract Navigator (Graph view) and selecting Contract | Edit | Binding, to display the Edit Binding panel as shown in Figure 79.



```
Bindings
  GoogleSearchBinding
    Operations
      doGetCachedPage
      doSpellingSuggestion
      doGoogleSearch
```

The following is the translated SOAP Operation (physical element):

Name	Node	SOAP Ac...	Style	Use	Encodin...	Namesp...
doGetCachedPage	operation					
<SOAP Operation>	soap:op...	urn:Goo...				
doGetCachedPage	input			encoded	http://sc...	urn:Goo...
doGetCachedPageResp	output			encoded	http://sc...	urn:Goo...

Figure 79: *Edit Binding panel*

At the Edit Binding panel, you can delete operations, by selecting them and clicking the **Delete** button.

You can also change some of the Binding Element attributes by either double-clicking the cell and typing a new value, or by clicking the cell and selecting a new value from the drop-down list provided.

When you have finished making your changes, click **Apply** to update the binding and **OK** to close the wizard and return to the Artix Designer.

CHAPTER 6

Adding Services

A service defines the ports supported by the Web Service.

In this chapter

This chapter discusses the following topics:

Introduction

The final piece of information needed to describe how to connect a remote service is the network information needed to locate it. This information is defined inside a `<port>` element. Each port specifies the address and configuration information for connecting the application to a network.

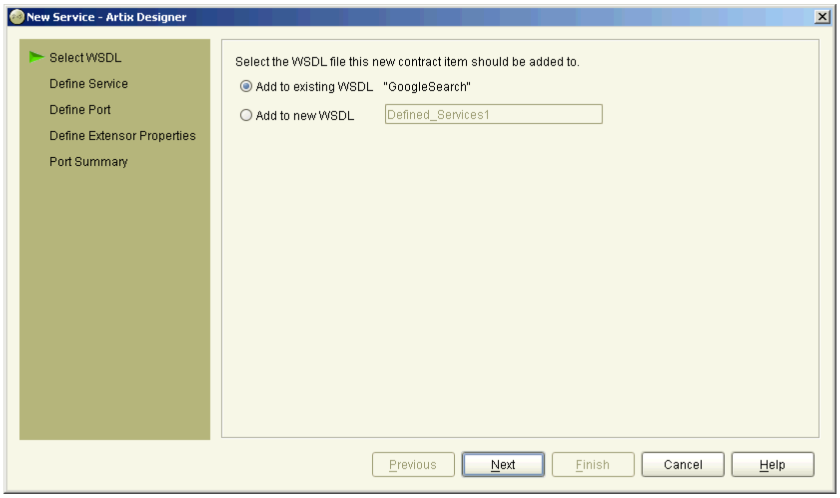
For each of the supported protocols, there is one `<port>` element. The `<service>` element is a collection of these ports. A service can contain one or many ports.

Typically, ports defined within a particular service are related in some way. For example all of the ports might be bound to the same port type, but use different network protocols, like HTTP and WebSphere MQ.

Procedure

To add a Service to your Artix contract:

1. Select **Contract | New | Service** from the menu bar to display the New Service wizard, as shown in Figure 80.



Select the WSDL file this new contract item should be added to.

(●) Add to existing WSDL "GoogleSearch"

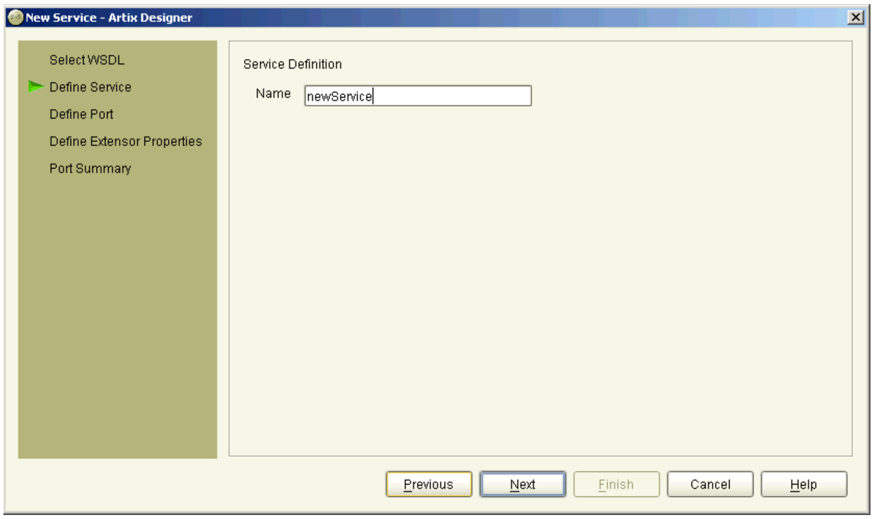
( ) Add to new WSDL Defined_Services1

Figure 80: *New Service wizard*

2. Select where to create the WSDL entry for the new service.

- **Add to existing WSDL** adds the service information to the existing contract.
- **Add to new WSDL** creates a new WSDL document that contains the service information.

3. Click **Next** to display the Service Definition panel, as shown in Figure 81.



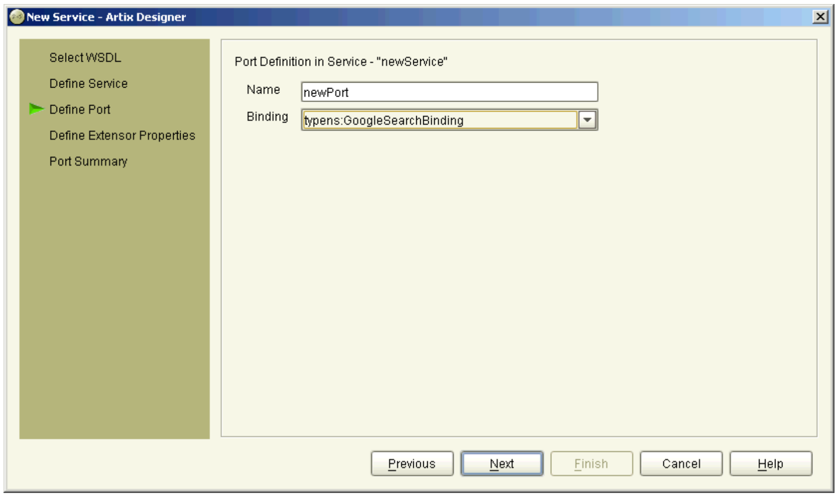
Service Definition

Name newService

Figure 81: *New Service wizard - Service Definition panel*

4. Enter a name for the new service, or accept the default provided.

5. Click **Next** to display the Port Definition panel, as shown in Figure 82.



Port Definition in Service - "newService"

Name: newPort

Binding: typens:GoogleSearchBinding

Figure 82: *New Service wizard - Port Definition panel*

6. Enter a name for the new port that is being created as part of this service, or accept the default provided.
7. From the **Binding** drop down list, select the binding that the port is going to expose.
8. Click **Next** to display the Extensor Properties panel.

Now you need to turn to the page that is relevant for the type of service you are creating:

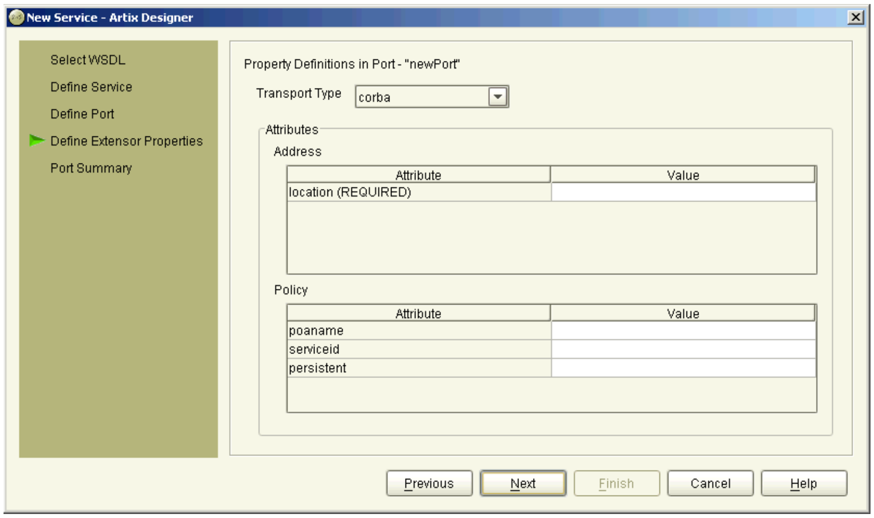
- For a CORBA service, see page 113
- For a non-secure HTTP service, see page 115
- For a secure HTTP service, see page 116
- For a WebSphere MQ service, see page 118
- For a Tuxedo service, see page 120
- For a Java Message Service (JMS), see page 123
- For an IIOP Tunnel service, see page 125
- For a non-secure SOAP over HTTP service, see page 128
- For a secure SOAP over HTTP service, see page 131

Adding a CORBA Port

CORBA ports are described using the IONA-specific WSDL elements `<corba:address>` and `<corba:policy>` within the WSDL `<port>` element, to specify how a CORBA object is exposed.

Procedure

1. At the Extensor Properties panel, as shown in Figure 83, select **CORBA** from the **Transport Type** drop down list.



Property Definitions in Port - "newPort"

Transport Type: corba

Attributes

Address

Attribute	Value
location (REQUIRED)	

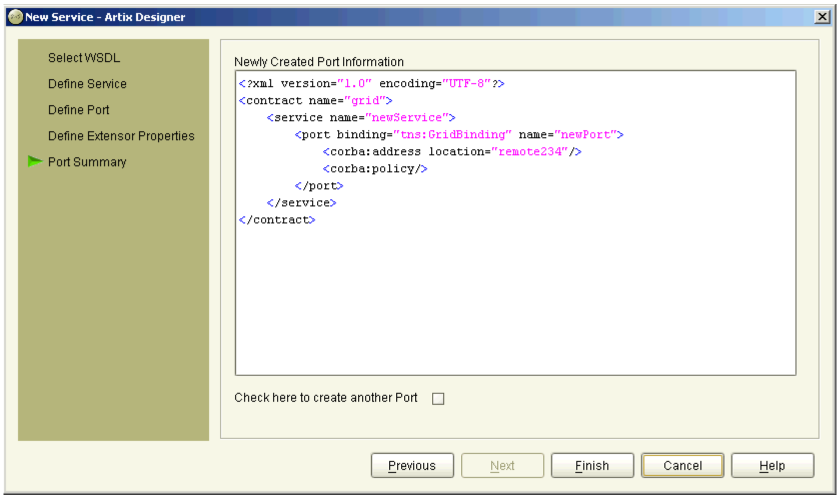
Policy

Attribute	Value
poaname	
serviceid	
persistent	

Figure 83: *New Service wizard - Define CORBA Extensor Properties*

2. In the **Address** table, enter the CORBA address in the **Location** field.
3. If you want to set any of the supported Policy Attributes, enter a valid value in the **Policy** table for any or all of the attributes listed.

4. Click **Next** to display the Summary panel, as shown in Figure 84.



Newly Created Port Information

```
<?xml version="1.0" encoding="UTF-8"?>
<contract name="grid">
    <service name="newService">
        <port binding="tns:GridBinding" name="newPort">
            <corba:address location="remote234"/>
            <corba:policy/>
        </port>
    </service>
</contract>
```

Check here to create another Port ☐

Figure 84: *New Service wizard - Summary panel (CORBA)*

5. To add another port to this service, check the box provided under the summary panel and click **Next**. This will take you back to the Define Port panel (as shown in Figure 82 on page 112), where you can enter details for the new port.
6. Click **Finish** to close this wizard and return to the Designer.

Artix expects the IOR for the CORBA object to be located in a file called `objref.ior`, and creates a persistent POA with an object id of `personalInfo` to connect the CORBA application.

Adding an HTTP Port

When adding an HTTP port, you have the option of making it either secure or non-secure. A secure port means that the connections with that port, and information moving in and out of it, are secure.

Non-Secure Connections

This section describes how to add an HTTP port that does not enable secure connections.

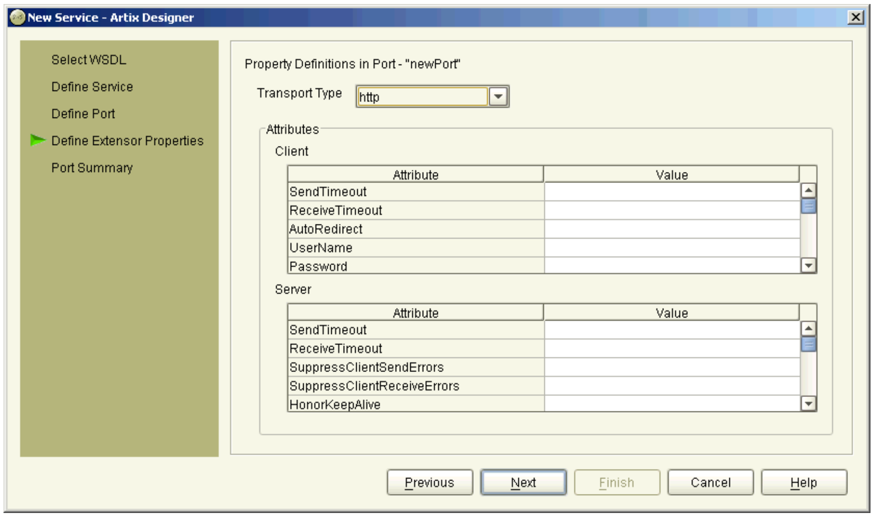
Before you begin

To add a port, you must have already created a binding within the `<binding>` component of the contract. See "Adding Bindings" on page 93 for more information.

Procedure

To add an HTTP port to your service contract:

1. At the Extensor Properties panel, as shown in Figure 85, select **http** from the **Transport Type** drop-down list.



Property Definitions in Port - "newPort"

Transport Type: http

Attributes

Client

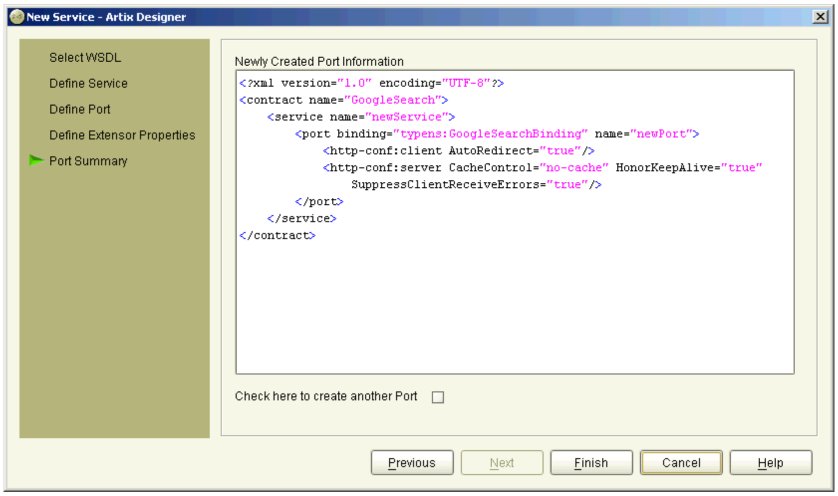
Attribute	Value
SendTimeout	
ReceiveTimeout	
AutoRedirect	
UserName	
Password	

Server

Attribute	Value
SendTimeout	
ReceiveTimeout	
SuppressClientSendErrors	
SuppressClientReceiveErrors	
HonorKeepAlive	

Figure 85: *New Service wizard - Define HTTP Extensor Properties*

2. To specify a value for a one of the client or server attribute, type (or in the case of certain true or false attributes select) the value you want.
3. Click **Next** to display the Summary panel, as shown in Figure 86.


New Service - Artix Designer
Select WSDL
Define Service
Define Port
Define Extensor Properties
Port Summary
Newly Created Port Information
Check here to create another Port
Previous
Next
Finish
Cancel
Help

```
<?xml version="1.0" encoding="UTF-8"?>
<contract name="GoogleSearch">
    <service name="newService">
        <port binding="typens:GoogleSearchBinding" name="newPort">
            <http-conf:client AutoRedirect="true"/>
            <http-conf:server CacheControl="no-cache" HonorKeepAlive="true"
                SuppressClientReceiveErrors="true"/>
        </port>
    </service>
</contract>
```

Figure 86: *New Service wizard - Summary panel (HTTP)*

4. To add another port to this service, check the box provided under the summary panel and click **Next**. This will take you back to the Define Port panel (as shown in Figure 82 on page 112), where you can enter details for the new port.
5. Click **Finish** to close this wizard and return to the Artix Designer.

Secure Connections

This section describes how to add an HTTP port that enables secure connections.

Before you begin

To add a port, you must have already created a payload format binding within the `<binding>` component of the contract. See “Adding Bindings” on page 93 for more information.

SSL-related attributes

The SSL-related attributes that can be configured to be included in the `<http-conf:client>` and `<http-conf:server>` elements of an HTTP port binding are as follows:

Client SSL Attributes	Server SSL Attributes
`UseSecureSockets`	`UseSecureSockets`
`ClientCertificate`	`ServerCertificate`
`ClientCertificateChain`	`ServerCertificateChain`
`ClientPrivateKey`	`ServerPrivateKey`
`ClientPrivateKeyPassword`	`ServerPrivateKeyPassword`
`TrustedRootCertificate`	`TrustedRootCertificate`

Procedure

Follow the steps described in "Procedure" on page 115, with the following minor changes:

- Specify `https://` rather than `http://` as the prefix for the value of the **URL** attribute in the **Client** configuration table.
- Enter values for the various SSL-related attributes in the **Client** and **Server** configuration tables. See "SSL-related attributes" above for a listing of these attributes.

Note: When you specify `https://` as the prefix for the value of the **URL** attribute in the **Client** configuration table, a secure HTTP connection is automatically enabled, even if **UseSecureSockets** is not set to `true`.

Adding a WebSphere MQ Port

The description for an Artix WebSphere MQ port is entered in a `<port>` element of the Artix contract containing the interface to be exposed over WebSphere MQ. Artix defines two elements to describe WebSphere MQ ports and their attributes:

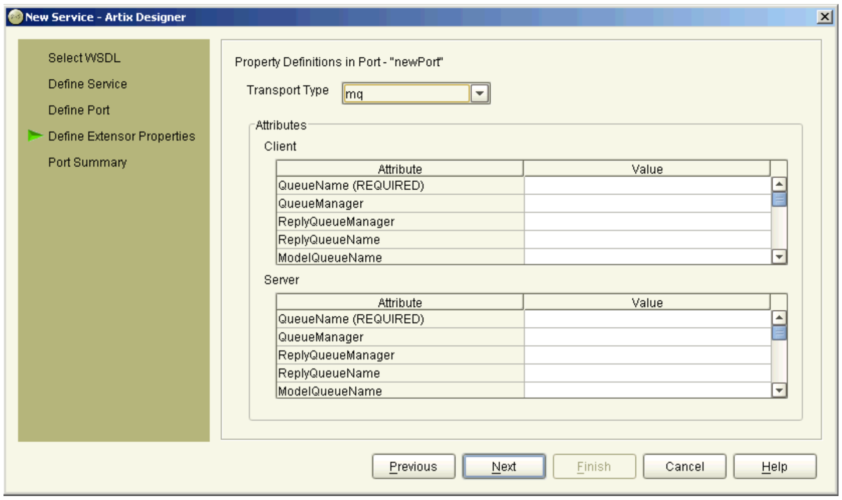
- <mq:client> describes the port Artix client applications use to connect to an WebSphere MQ server application.
- <mq:server> describes the port WebSphere MQ client applications use to connect to Artix.

You can use one or both of the WebSphere MQ elements to describe the Artix WebSphere MQ port. Each can have different configurations depending on the attributes you choose to set.

Procedure

To add a WebSphere MQ port to an Artix contract:

1. At the Extensor Properties panel, as shown in Figure 87, select **mq** from the **Transport Type** drop-down list.



Property Definitions in Port - "newPort"

Transport Type: mq

Attributes

Client

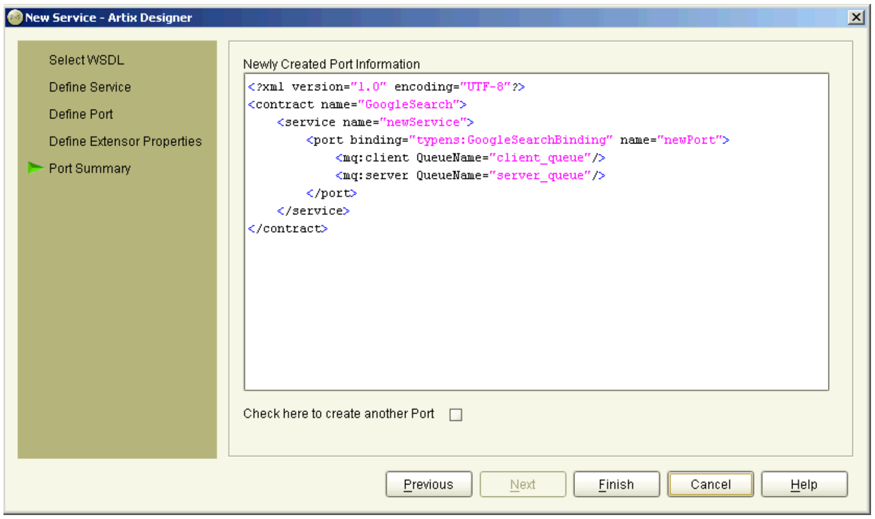
Attribute	Value
QueueName (REQUIRED)	
QueueManager	
ReplyQueueManager	
ReplyQueueName	
ModelQueueName	

Server

Attribute	Value
QueueName (REQUIRED)	
QueueManager	
ReplyQueueManager	
ReplyQueueName	
ModelQueueName	

Figure 87: *New Service Wizard - Define WebSphere MQ Port Properties*

2. Enter values for the desired attributes. You must supply `QueueName` values at a minimum.
3. Click **Next** to display the Port Summary panel as shown in Figure 88.



Newly Created Port Information

```
<?xml version="1.0" encoding="UTF-8"?>
<contract name="GoogleSearch">
    <service name="newService">
        <port binding="typens:GoogleSearchBinding" name="newPort">
            <mq:client QueueName="client_queue"/>
            <mq:server QueueName="server_queue"/>
        </port>
    </service>
</contract>
```

Check here to create another Port ☐

Figure 88: *New Service wizard - Summary panel (MQ)*

4. To add another port to this service, check the box provided under the summary panel and click **Next**. This will take you back to the Define Port panel (as shown in Figure 82 on page 112), where you can enter details for the new port.
5. Click **Finish** to close the wizard and return to the Artix Designer.

Adding a Tuxedo Port

Artix allows services to connect using Tuxedo's transport mechanism. This provides them with all of the qualities of service associated with Tuxedo.

To use the Tuxedo transport, you need to describe the port using Tuxedo in the physical part of an Artix contract. The extensions used to describe a Tuxedo port are defined in the namespace:

```
xmlns:tuxedo="http://schemas.iona.com/transports/tuxedo"
```

This namespace will need to be included in your Artix contract's `<definition>` element.

As with other transports, the Tuxedo transport description is contained within a `<port>` element. Artix uses `<tuxedo:server>` to describe the attributes of a Tuxedo port. `<tuxedo:server>` takes a single mandatory attribute, `serviceName`, which specifies the bulletin board name of the Tuxedo port being exposed.

Before you begin

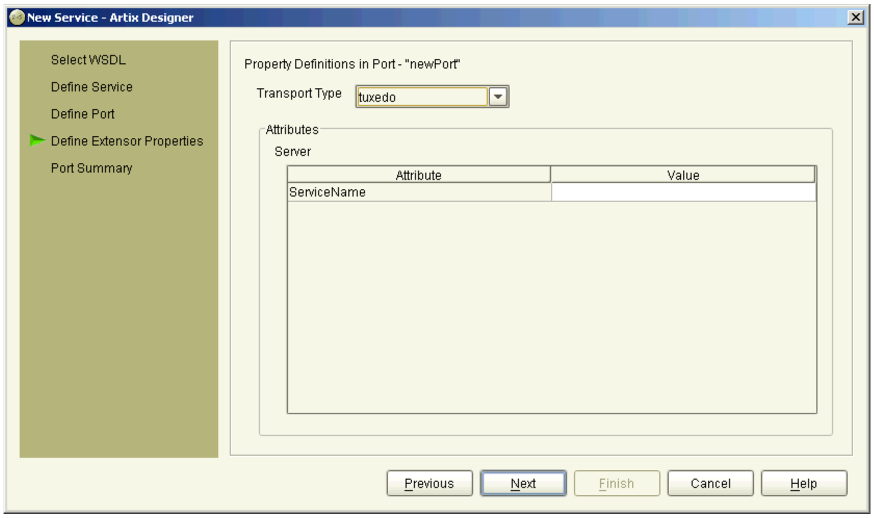
Note that your Artix contract must have an existing SOAP binding before you can add a Tuxedo port. For more information, see "Adding a SOAP Binding" on page 100.

Procedure

To add a Tuxedo port to an Artix contract:

1. At the **Define Port** panel (as shown in Figure 82 on page 112), select the SOAP binding which this port will expose to the network from the Binding drop-down list.
2. Click **Next** to display the Extensor Properties panel.

3. Select **Tuxedo** from the **Transport Type** drop-down list to display the Tuxedo attributes as shown in Figure 89.



Property Definitions in Port - "newPort"

Transport Type: tuxedo

Attributes

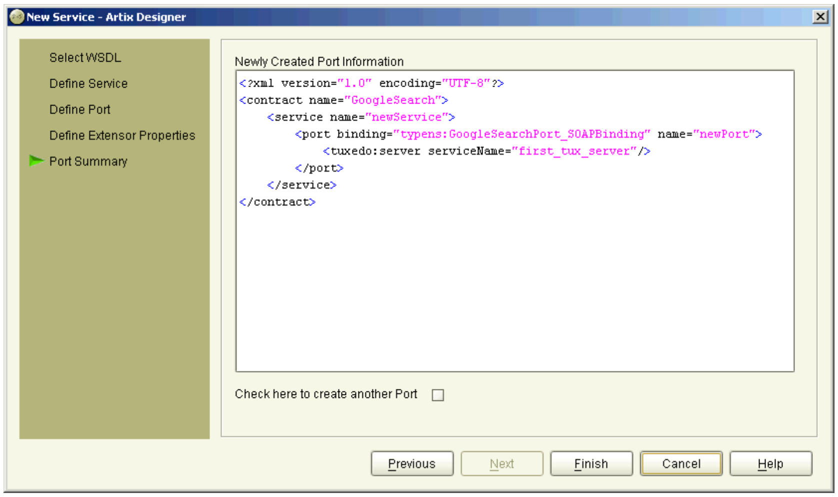
Server

Attribute	Value
ServiceName	

Figure 89: *New Service wizard - Define Tuxedo Port Properties panel*

4. Enter a valid Tuxedo service name in the **Service Name Value** field.

5. Click **Next** to display the Summary panel, as shown in Figure 90.



Newly Created Port Information

```
<?xml version="1.0" encoding="UTF-8"?>
<contract name="GoogleSearch">
    <service name="newService">
        <port binding="typens:GoogleSearchPort_SOAPBinding" name="newPort">
            <tuxedo:server serviceName="first_tux_server"/>
        </port>
    </service>
</contract>
```

Check here to create another Port ☐

Figure 90: *New Service wizard - Summary panel (Tuxedo)*

6. To add another port to this service, check the box provided under the summary panel and click **Next**. This will take you back to the Define Port panel (as shown in Figure 82 on page 112), where you can enter details for the new port.
7. Click **Finish** to close this wizard and return to the Artix Designer.

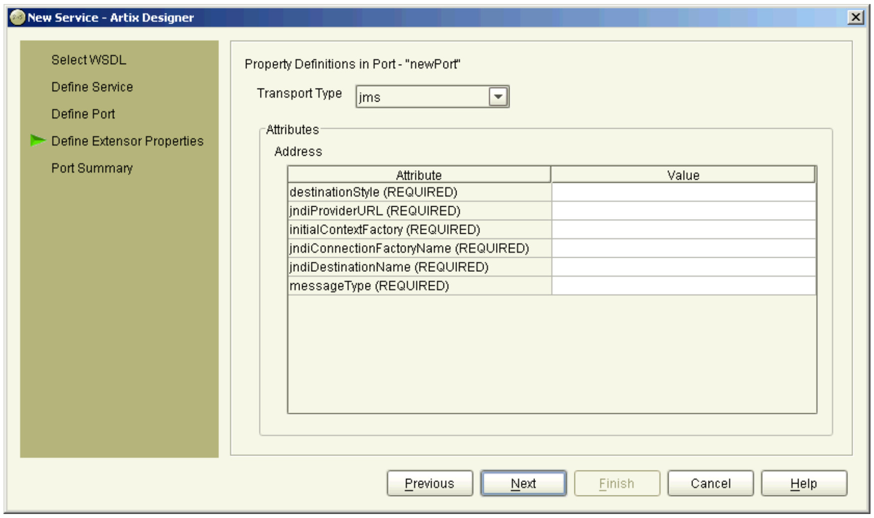
Adding a Java Message Service Port

The Java Messaging System (JMS) provides a standardized means for Java applications to send messages. Artix provides a transport plug-in that enables systems to place and receive messages from JMS implementations. One advantage of this is that Artix allows C++ applications to interact directly with Java applications over JMS.

Procedure

To add a Java Message Service (JMS) port to an Artix contract:

1. At the Extensor Properties panel, as shown in Figure 91, select **jms** from the **Transport Type** drop-down list.



Property Definitions in Port - "newPort"

Transport Type: jms

Attributes

Address

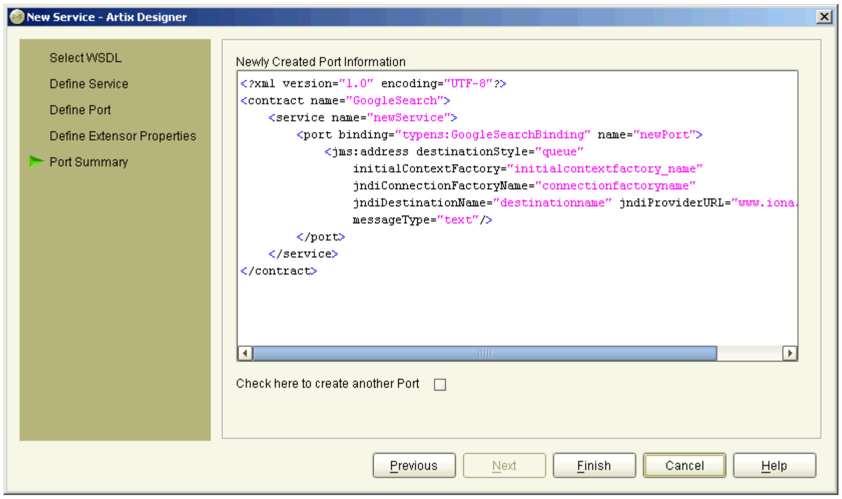
Attribute	Value
destinationStyle (REQUIRED)	
jndiProviderURL (REQUIRED)	
initialContextFactory (REQUIRED)	
jndiConnectionFactoryName (REQUIRED)	
jndiDestinationName (REQUIRED)	
messageType (REQUIRED)	

Figure 91: *New Service Wizard - Define WebSphere MQ Port Properties*

2. Enter values for the desired attributes. All attributes are required.
   - **destinationStyle** - Specifies the type of jms messaging object you're connecting to; options are topic (one-way only) or queue.
   - **jndiProviderURL** - Specifies the URL of the JNDI service where the connection information for the JMS destination is stored.

- **initialContextFactory** - Specifies the name of the `InitialContextFactory` class or a list of package prefixes used to construct URL context factory classnames.
- **jndiConnectionFactoryName** - Specifies the JNDI name bound to the JMS connection factory to use to connect to the JMS destination.
- **jndiDestinationName** - Specifies the JNDI name bound to the JMS destination to which Artix connects.
- **messageType** - Specifies how the message data will be packaged as a JMS message. `text` specifies that the data will be packaged as a `TextMessage`. `binary` specifies that the data will be packaged as an `ObjectMessage`.

3. Click **Next** to display the Summary panel as shown in Figure 88.


New Service - Artix Designer
Select WSDL
Define Service
Define Port
Define Extensor Properties
Port Summary
Newly Created Port Information

```
<?xml version="1.0" encoding="UTF-8"?>
<contract name="GoogleSearch">
    <service name="newService">
        <port binding="typens:GoogleSearchBinding" name="newPort">
            <jms:address destinationStyle="queue"
                initialContextFactory="initialcontextfactory_name"
                jndiConnectionFactoryName="connectionfactoryname"
                jndiDestinationName="destinationname" jndiProviderURL="www.iona.
                messageType="text"/>
        </port>
    </service>
</contract>
```

Check here to create another Port ☐

Previous
Next
Finish
Cancel
Help

Figure 92: *New Service wizard - Summary panel (JMS)*

4. To add another port to this service, check the box provided under the summary panel and click **Next**. This will take you back to the Define Port panel (as shown in Figure 82 on page 112), where you can enter details for the new port.
5. Click **Finish** to close the wizard and return to the Artix Designer.

Adding an IIOP Tunnel Port

An IIOP tunnel provides a means for taking advantage of existing CORBA services while transmitting messages using a payload format other than CORBA. For example, you could use an IIOP tunnel to send fixed format messages to an endpoint whose address is published in a CORBA naming service.

Supported payload formats

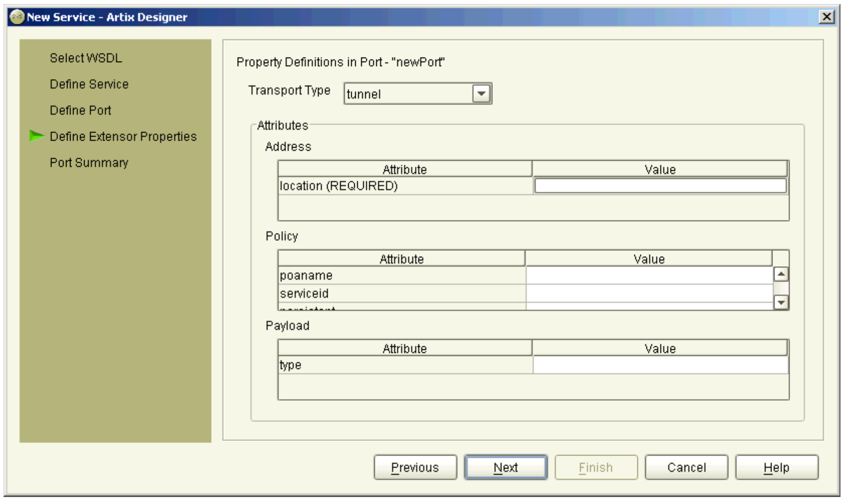
IIOP tunnels can transport messages using the following payload formats:

- SOAP
- Fixed format
- Fixed record length
- G2++
- Octet streams

Procedure

To add an IIOP tunnel port to your service contract:

1. At Extensor Properties panel, as shown in Figure 93, select **tunnel** from the **Transport Type** drop-down list.



Property Definitions in Port - "newPort"

Transport Type: tunnel

Attributes

Address

Attribute	Value
location (REQUIRED)	

Policy

Attribute	Value
poaname	
serviceid	

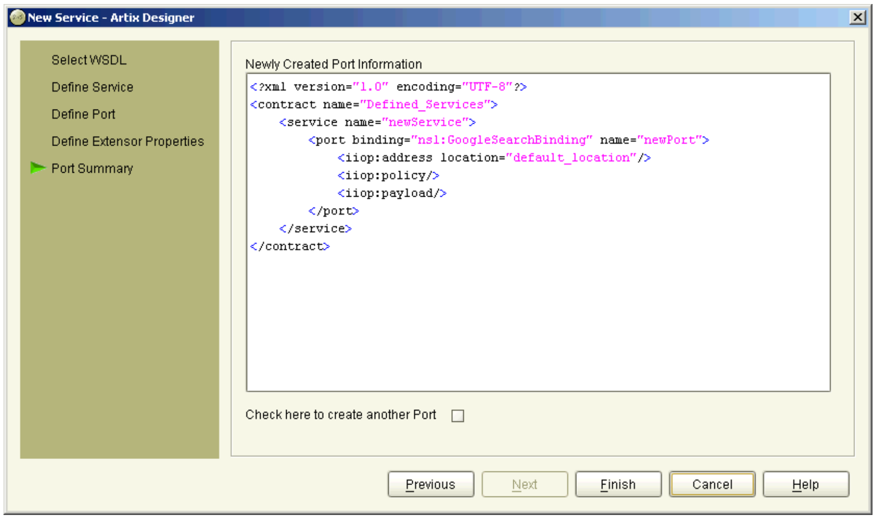
Payload

Attribute	Value
type	

Figure 93: *New Service wizard - Define IIOP Port Properties panel*

2. From the drop down list in the **Transport** box, select **tunnel**.
3. In the **Address** table, enter the address in the line for **Location**.
4. If you want to set any of the supported POA policies, place a check in the **Specified** box on the appropriate line in the **Policy** table and enter a valid value.

5. Click **Next** to display the Port Summary panel, as shown in Figure 94.


New Service - Artix Designer
Select WSDL
Define Service
Define Port
Define Extensor Properties
Port Summary
Newly Created Port Information
Check here to create another Port
Previous
Next
Finish
Cancel
Help

```
<?xml version="1.0" encoding="UTF-8"?>
<contract name="Defined_Services">
    <service name="newService">
        <port binding="ns1:GoogleSearchBinding" name="newPort">
            <iiop:address location="default_location"/>
            <iiop:policy/>
            <iiop:payload/>
        </port>
    </service>
</contract>
```

Figure 94: *New Service wizard - Summary panel (IIOP)*

6. To add another port to this service, check the box provided under the summary panel and click **Next**. This will take you back to the Define Port panel (as shown in Figure 82 on page 112), where you can enter details for the new port.
7. Click **Finish** to close this wizard and return to the Artix Designer.

Artix expects the IOR for the IIOP tunnel to be located in a file called `objref.ior`, and creates a persistent POA with an object id of `personalInfo` to configure the IIOP tunnel.

Adding a SOAP Port

Non-Secure Connections

This section describes how to add a port for SOAP over HTTP that does not enable secure connections.

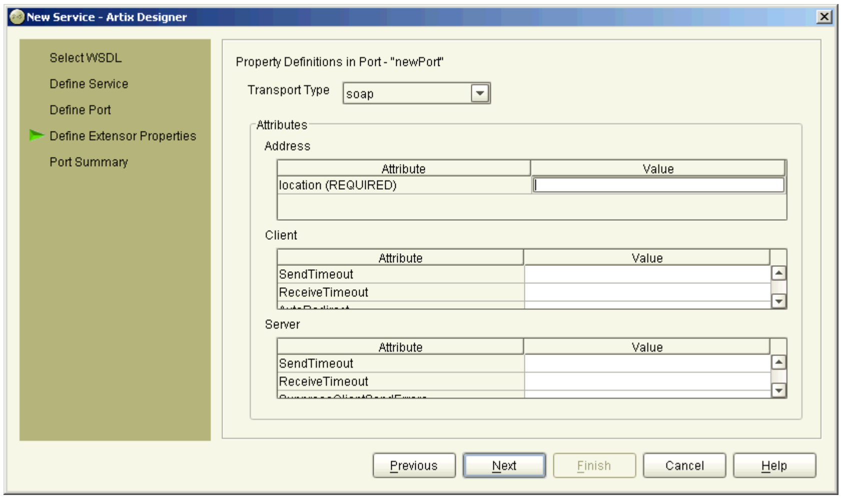
Before you begin

To add a port, you must have already created a payload format binding within the `<binding>` component of the contract. See "Adding Bindings" on page 93 for more information.

Procedure

To enable the use of SOAP over HTTP:

1. At the Extensor Properties panel, as shown in Figure 95, select **SOAP** from the **Transport Type** drop-down list.



Property Definitions in Port - "newPort"

Transport Type: soap

Attributes

Address

Attribute	Value
location (REQUIRED)	

Client

Attribute	Value
SendTimeout	
ReceiveTimeout	

Server

Attribute	Value
SendTimeout	
ReceiveTimeout	

Figure 95: *New Service wizard - Define SOAP Properties panel*

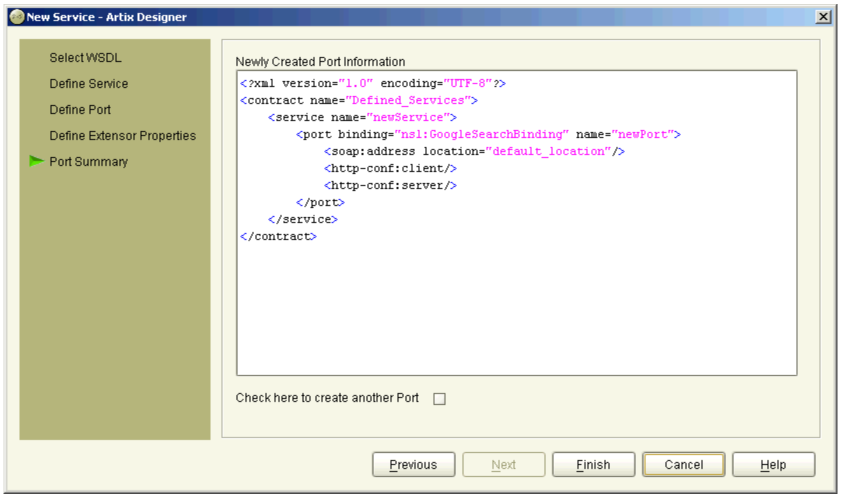
2. In the **Value** field corresponding to the **location** line of the **Address** configuration table, type the URL that represents the resource being requested.

   Note: The **Address** configuration table relates to the `soap:address` element within the port component of the WSDL contract. You must specify a value for the **location** attribute.

3. To specify a value for another attribute, place a check in the **Specified** box on the appropriate line in the appropriate configuration table, and type or (in the case of certain true or false attributes) select the value you want.

   Note: All attributes are optional in the **Client** and **Server** configuration tables. These relate to the `http-conf:client` and `http-conf:server` elements that can be specified as peers of the `soap:address` element under the same port binding. See “SSL-related attributes” below for details of each attribute relating to `http-conf:client` and `http-conf:server`.

4. Click **Next** to display the Summary panel, as shown in Figure 96.


New Service - Artix Designer
Select WSDL
Define Service
Define Port
Define Extensor Properties
Port Summary
Newly Created Port Information
Check here to create another Port
Previous
Next
Finish
Cancel
Help

```
<?xml version="1.0" encoding="UTF-8"?>
<contract name="Defined_Services">
    <service name="newService">
        <port binding="ns1:GoogleSearchBinding" name="newPort">
            <soap:address location="default_location"/>
            <http-conf:client/>
            <http-conf:server/>
        </port>
    </service>
</contract>
```

Figure 96: *New Service wizard - Summary panel (SOAP)*

5. To add another port to this service, check the box provided under the summary panel and click **Next**. This will take you back to the Define Port panel (as shown in Figure 82 on page 112), where you can enter details for the new port.
6. Click **Finish** to close this wizard and return to the Artix Designer.

Secure Connections

This section describes how to add a port for SOAP over HTTP that enables secure connections.

Before you begin

To add a port, you must have already created a payload format binding within the `<binding>` component of the contract. See "Adding Bindings" on page 93 for more information.

SSL-related attributes

The SSL-related attributes that can be configured to be included in the `<http-conf:client>` and `<http-conf:server>` elements of an HTTP port binding are as follows:

Client SSL Attributes	Server SSL Attributes
`UseSecureSockets`	`UseSecureSockets`
`ClientCertificate`	`ServerCertificate`
`ClientCertificateChain`	`ServerCertificateChain`
`ClientPrivateKey`	`ServerPrivateKey`
`ClientPrivateKeyPassword`	`ServerPrivateKeyPassword`
`TrustedRootCertificate`	`TrustedRootCertificate`

Procedure

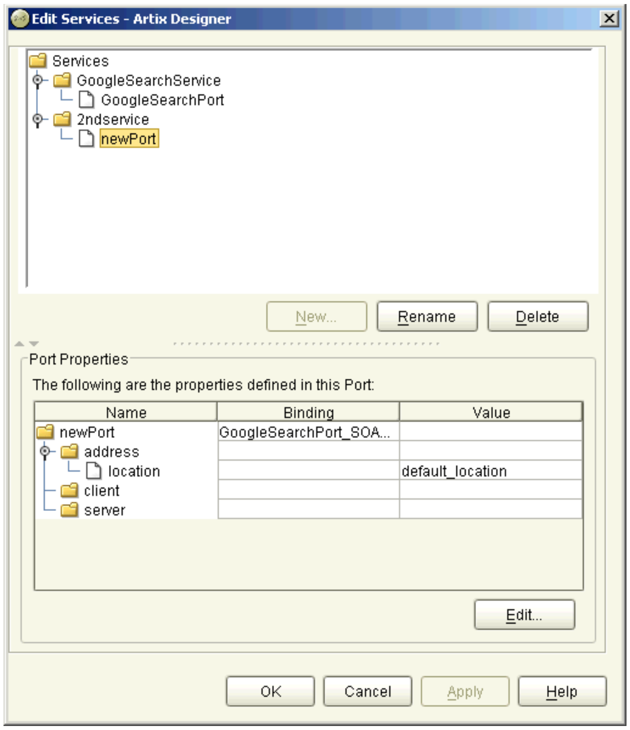
Follow the steps in "Procedure" on page 128, with the following minor changes:

- Specify `https://` rather than `http://` as the prefix for the value of the **location** attribute in the **Address** configuration table.
- Enter values for the various SSL-related attributes in the **Client** and **Server** configuration tables. See "SSL-related attributes" above for a listing of these attributes.

Note: When you specify `https://` as the prefix for the value of the **location** attribute in the **Address** configuration table, a secure HTTP connection is automatically enabled, even if **UseSecureSockets** is not set to `true`.

Editing Services

You can edit a service by selecting **Contract** | **Edit** | **Services**, to display the Edit Services panel as shown in Figure 97.



Name	Binding	Value
newPort	GoogleSearchPort_SOA...	
address		
location		default_location
client		
server		

Figure 97: *Edit Services panel*

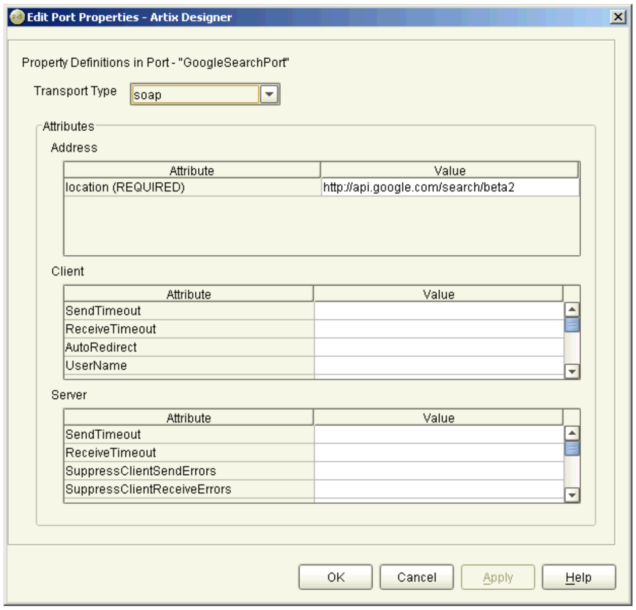
At the Edit Services panel, all of your services and their associated ports are listed in the top half of the dialog. From here you can:

- Rename a service or a port by selecting it and clicking **Rename**
- Delete a service or a port by selecting it and clicking **Delete**.
- Add a new service by clicking **Add** to display the New Service wizard.

Editing port properties

When you select a port in the top of this dialog, the port properties are displayed in the Port Properties panel at the bottom of the dialog.

To change any of the Port Properties, click the **Edit** button to display the Edit Port Properties dialog, as shown in Figure 98.


Edit Port Properties - Artix Designer
Property Definitions in Port - "GoogleSearchPort"
Transport Type
soap
Attributes
Address
Client
Server
OK
Cancel
Apply
Help

Attribute	Value
location (REQUIRED)	http://api.google.com/search/beta2

Attribute	Value
SendTimeout	
ReceiveTimeout	
AutoRedirect	
UserName	

Attribute	Value
SendTimeout	
ReceiveTimeout	
SuppressClientSendErrors	
SuppressClientReceiveErrors	

Figure 98: *Edit Port Properties dialog*

To change values of attributes in this dialog, click on the value field to either select or type the new value.

When you have finished making your changes, click **Apply** to update the port, and **OK** to close the wizard and return to the Edit Services dialog, where your changes are displayed in the Port Properties panel.

Click **OK** to close this dialog and return to the Artix Designer.

CHAPTER 7

Routing Messages

Artix provides messages routing based on operations, ports, or message attributes.

In this chapter

This chapter discusses the following topics:

What is a Route?

Overview

Artix routing is implemented within Artix collections and is controlled by rules specified in the collection's contract. Artix collections that include routing rules can be deployed into an Artix service.

Artix supports the following types of routing:

- "Port-based"
- "Operation-based"

A router's contract must include definitions for the source services and destination services. The contract also defines the routes that connect source and destination ports, according to some specified criteria. This routing information is all that is required to implement port-based or operation-based routing. Content-based routing requires that application code be written to implement the routing logic.

Port-based

Port-based routing acts on the port or transport-level identifier, specified by a `<port>` element in an Artix contract. This is the most efficient form of routing. Port-based routing can also make a routing decision based on port properties, such as the message header or message identifier. Thus Artix can route messages based on the origin of a message or service request, or based on the message header or identifier.

Operation-based

Operation-based routing lets you route messages based on the logical operations described in an Artix contract. Messages can be routed between operations whose arguments are equivalent. Operation-based routing can be specified on the interface, `<portType>`, level or the finer grained operation level.

Creating a Route

Overview

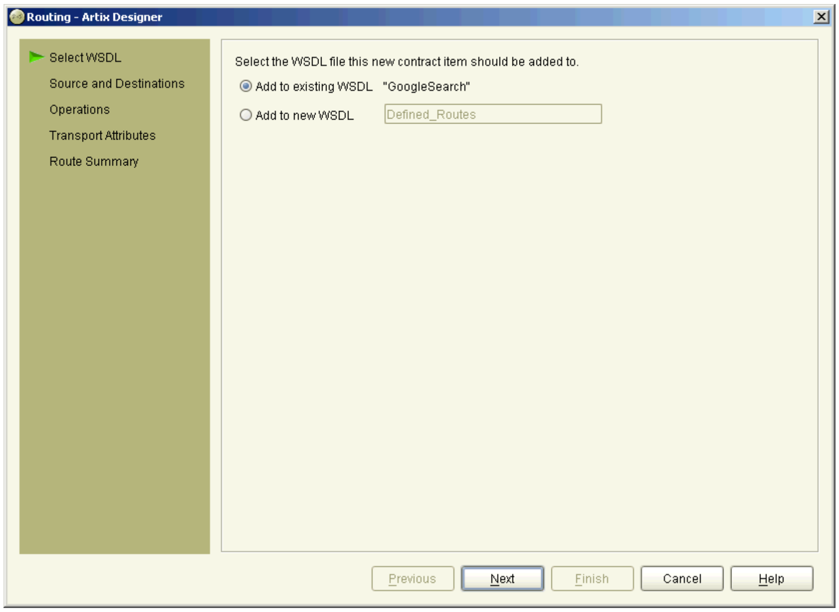
The Artix Designer includes a routing wizard that assists you in creating routes from the services available in your contract. It walks you through the steps of creating a route and provides you with the valid options for the services available. It performs all of the compatibility testing for you and will never allow you to create an invalid route.

Procedure

To create a route:

1. From the Designer Tree, select a contract with multiple service definitions that have operations that can be routed.
2. Select **Contracts | New | Route** from the menu bar to display the New Route wizard, as shown in Figure 99.

> **Note:** If the Route option is not available, your contract does not have any compatible operations for routing. For a contract to be able to be routed, it needs to contain two or more services with compatible port types. See "Adding Services" on page 109 for more information.



Select the WSDL file this new contract item should be added to.

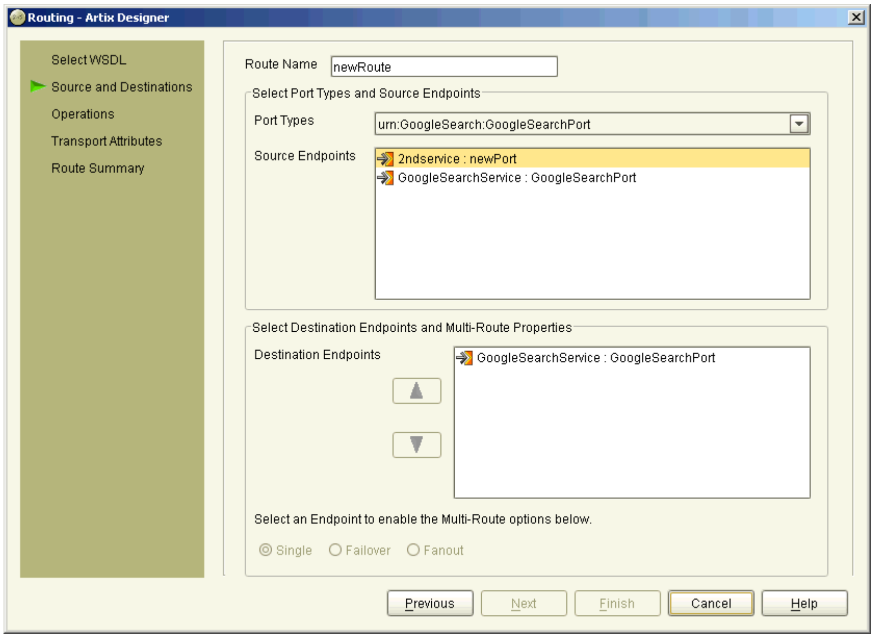
(●) Add to existing WSDL "GoogleSearch"

( ) Add to new WSDL Defined_Routes

Figure 99: *New Route wizard*

3. Select where you want to add the routing information.
   - **Add to existing WSDL** adds the route information to the existing contract.
   - **Add to new WSDL** creates a new WSDL document that contains the route information.

4. Click **Next** to display the Source and Destination panel, as shown in Figure 100.



Route Name: newRoute

Select Port Types and Source Endpoints

Port Types: urn:GoogleSearch:GoogleSearchPort

Source Endpoints:
- 2ndservice : newPort
- GoogleSearchService : GoogleSearchPort

Select Destination Endpoints and Multi-Route Properties

Destination Endpoints:
- GoogleSearchService : GoogleSearchPort

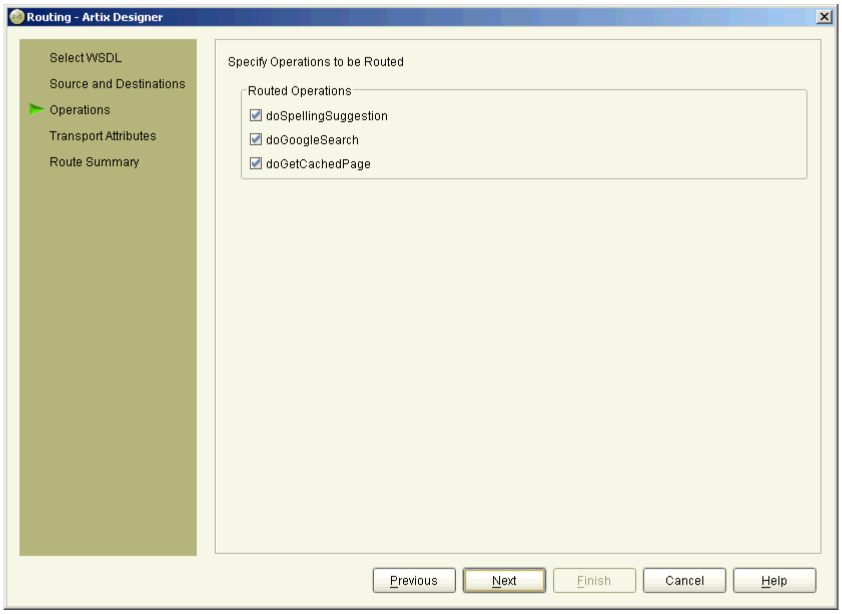
Select an Endpoint to enable the Multi-Route options below.

[✓ Single] Failover Fanout

Figure 100:*New Route wizard - Source and Destination panel*

5. Enter a name for the route, or accept the default provided
6. Select the source `portType` for the route from the **PortType** pull-down list.
7. Select the source endpoint from the available options in the **Source Endpoints** list.
8. Select the destination endpoint from the available options in the **Destination Endpoints** list.
9. If you selected multiple destination endpoints on the previous screen, select either **Failover** or **Fanout** under **Multiple Route Destination Preference**.

10. Click **Next** to display the Operation Routing panel, as shown in Figure 101.



Specify Operations to be Routed

Routed Operations

- ✓ doSpellingSuggestion
- ✓ doGoogleSearch
- ✓ doGetCachedPage

Figure 101:*New Route wizard - Operation Routing panel*

11. Select the operations you want to route from the list provided. By default, all operations are pre-selected.

12. Click **Next** to display the Transport Attributes panel, as shown in Figure 102.

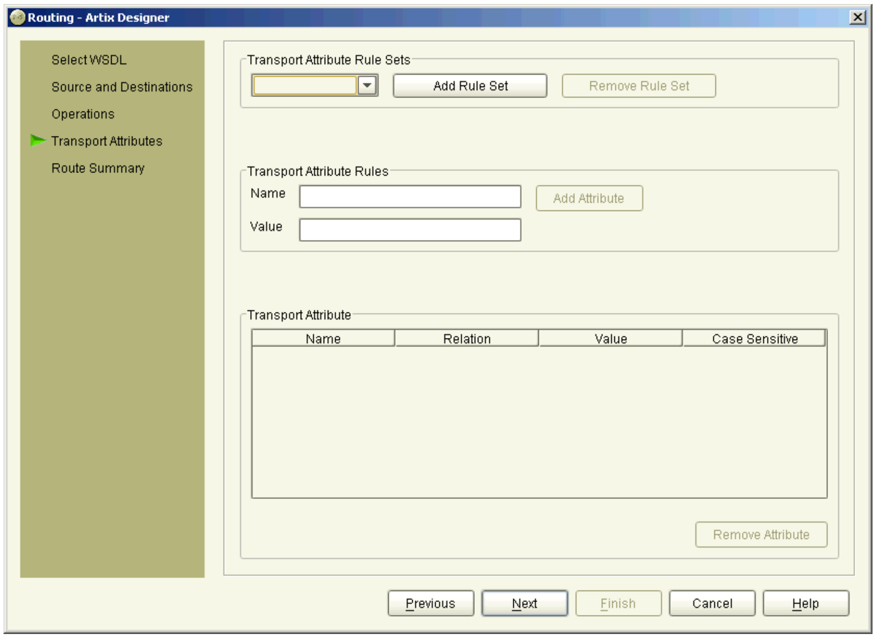
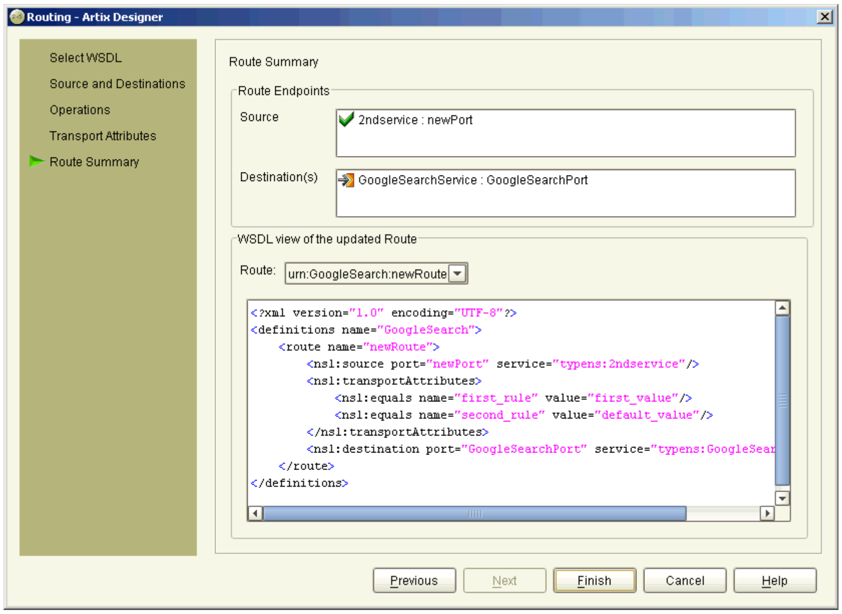


Figure 102:*New Route wizard - Transport Attributes panel*

13. Click **Add Rule Set** to add transport attribute based routing rules. The counter will automatically start at **0**.
14. Enter the name of the transport attribute.
15. Enter the value to be used for the attribute.
16. Click **Add Attribute** to add the attribute to the Transport Attribute table. When the attribute is in the table you can edit it to determine how matching attributes are compared to the value.
17. Repeat **steps 13-16** for all the attributes you want to use in your route.

Note: The editor has no knowledge of the valid attribute names and will allow you to enter any names and values.

18. Click **Next** to display the Summary panel, as shown in Figure 103.



Route Summary

Route Endpoints

Source: 2ndservice : newPort

Destination(s): GoogleSearchService : GoogleSearchPort

WSDL view of the updated Route

Route: urn:GoogleSearch:newRoute

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="GoogleSearch">
    <route name="newRoute">
        <nsl:source port="newPort" service="typens:2ndservice"/>
        <nsl:transportAttributes>
            <nsl:equals name="first_rule" value="first_value"/>
            <nsl:equals name="second_rule" value="default_value"/>
        </nsl:transportAttributes>
        <nsl:destination port="GoogleSearchPort" service="typens:GoogleSear
    </route>
</definitions>
```

Figure 103:*New Route wizard - Summary panel*

19. Click **Finish** to close this wizard and return to the Artix Designer.

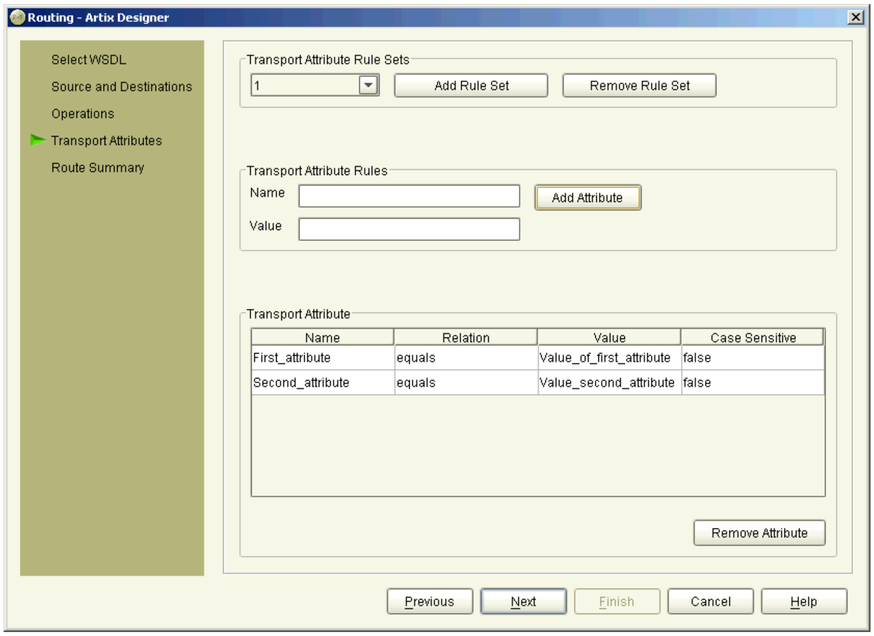
Editing a Route

The only things you can edit in a route are the transport attributes. When you choose to edit a route, the Transport Attributes panel for the New Route wizard is displayed for that route, enabling you to change any transport attributes that you previously created, or to add new transport attributes.

Procedure

To edit a route:

1. Select the Route in the Contract Navigator (graph view) and select Contract | Edit | Route to display the Transport Attributes panel, as shown in Figure 104.

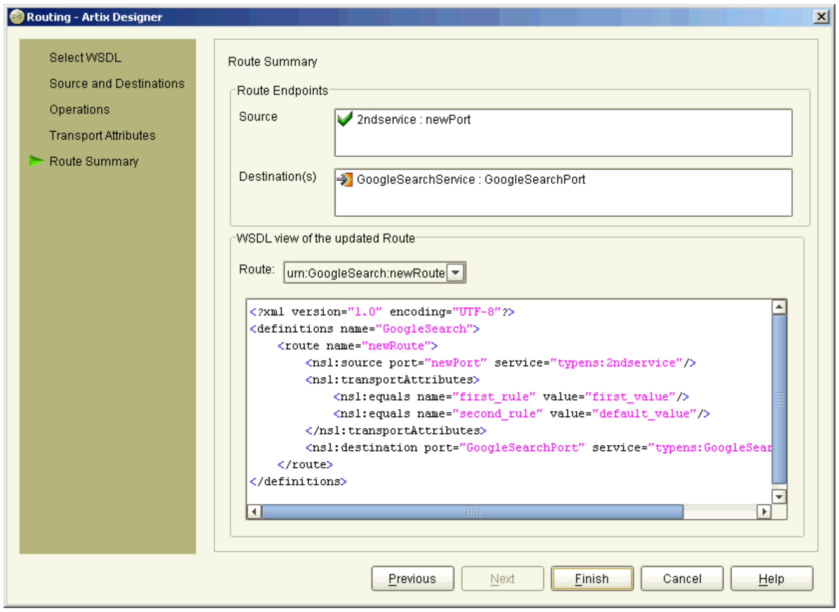


Name	Relation	Value	Case Sensitive
First_attribute	equals	Value_of_first_attribute	false
Second_attribute	equals	Value_second_attribute	false

Figure 104:*Transport Attributes panel - Editing a Route*

2. You can change the values for any of the existing transport attributes, or add new transport attributes.

3. When you have finished your changes, click OK to display the Summary panel, as shown in Figure 105. This panel will display the route with your changes included.



Route Summary

Route Endpoints

Source: 2ndservice : newPort

Destination(s): GoogleSearchService : GoogleSearchPort

WSDL view of the updated Route

Route: urn:GoogleSearch:newRoute

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="GoogleSearch">
    <route name="newRoute">
        <nsl:source port="newPort" service="typens:2ndservice"/>
        <nsl:transportAttributes>
            <nsl:equals name="first_rule" value="first_value"/>
            <nsl:equals name="second_rule" value="default_value"/>
        </nsl:transportAttributes>
        <nsl:destination port="GoogleSearchPort" service="typens:GoogleSear
    </route>
</definitions>
```

Figure 105:*Summary panel - Editing a Route*

CHAPTER 8

Deploying Collections

Artix collections can be deployed as often as you like using different configurations to satisfy your solution requirements.

In this chapter

This chapter discusses the following topics:

Deployment Explained

Overview

Artix Collections can be deployed as Java, C++, or CORBA-based applications. As part of the deployment process, you can use a collection to create a client, a server, or a switch, or any combination of all three options.

Deployment involves three steps:

1. Creating a Deployment Profile - see page 147
2. Creating the Deployment Bundle - see page 151
3. Deploying the bundle - see page 155

You do not have to perform all the steps in one go - you can perform one or more and then complete the rest later. You must, however, perform them in the order shown here. That is, you need to create a Deployment Profile before you can create a Deployment Bundle, and you cannot run the deployer until you have created a bundle.

As part of the deployment process, Artix generates four directories in your specified save location:

- `src` - contains the generated source code in the language you specified (C++, Java, or IDL)
- `etc` - contains the configuration information required for the application to run successfully
- `wsdl` - contains the locally defined WSDL contracts
- `bin` - contains the environment scripts and the start and stop (UNIX only) scripts

Creating a Deployment Profile

Overview

The Deployment Profile defines machine level-information such as the Artix save location, the compiler location, and the operating system being used. This profile can be used multiple times as it is not specific to any particular collection defined within the workspace.

If you create your workspace using one of the Fast Track templates, Artix creates a default local profile for you automatically, which you can use for deploying to your local machine. The details of this profile are displayed on the Workspace Details panel. For deploying to other machines, you need to create your own profiles.

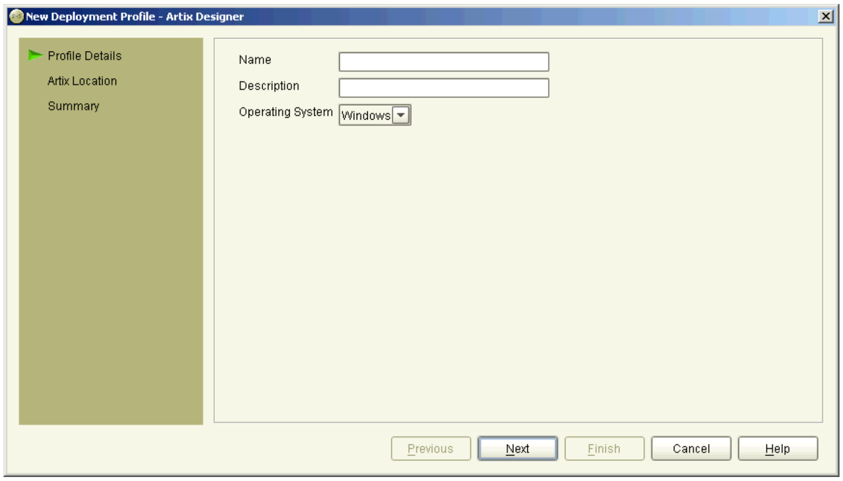
The next step after creating a Deployment Profile, is to create a Deployment Bundle to capture specific information about the deployment of a collection. Deployment Bundles are explained further in the next section of this chapter.

You can have as many Deployment Profiles as you like within your workspace, but each Deployment Bundle can reference only one Deployment Profile.

Procedure

To create a Deployment Profile:

1. Select **File | New | Deployment Profile** from the menu bar to display the Deployment Profile wizard, as shown in Figure 106.

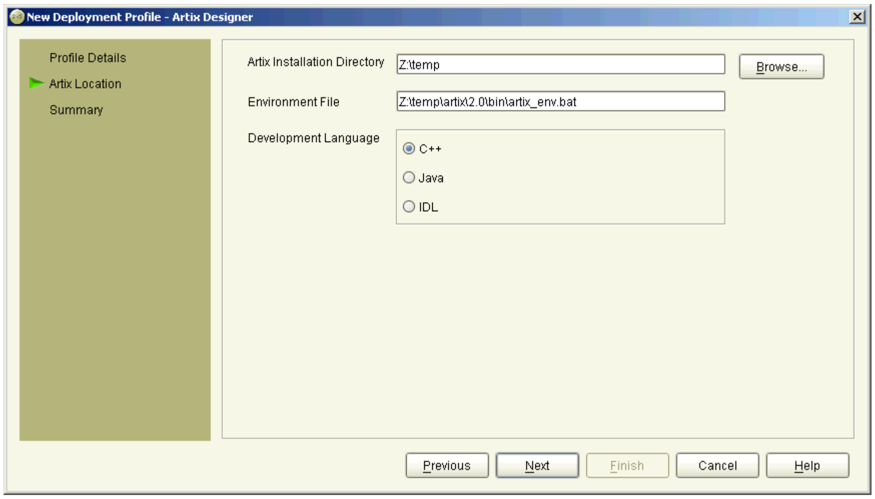


Name:
Description:
Operating System: Windows

Figure 106:*Deployment Profile wizard*

2. Enter a name for this profile.
3. Enter a description for this profile to help distinguish it from other profiles you may create.
4. Select the operating system for this profile from the list provided. Artix currently supports Windows or UNIX.

5. Click **Next** to display the Artix Location panel, as shown in Figure 107.



Artix Installation Directory: Z:\temp

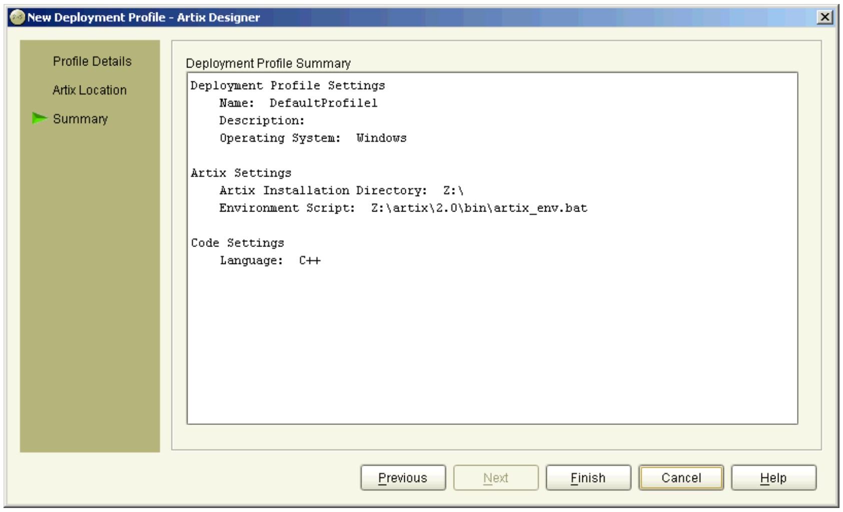
Environment File: Z:\temp\artix\2.0\bin\artix_env.bat

Development Language: [✓ C++] Java IDL

Figure 107:*Deployment Profile wizard - Artix Location panel*

6. Enter values for each of the fields provided on the panel, or accept the defaults provided. Changes you make to the **Location** field will be reflected in the **Environment File** field.
7. Select a **Development Language** for this profile from the options provided (**C++, Java, IDL**).

8. Click **Next** to display the Summary panel, as shown in Figure 108.


New Deployment Profile - Artix Designer
Profile Details
Artix Location
Summary
Deployment Profile Summary
Previous
Next
Finish
Cancel
Help

```
Deployment Profile Settings
    Name:  DefaultProfile1
    Description:
    Operating System:  Windows

Artix Settings
    Artix Installation Directory:  Z:\
    Environment Script:  Z:\artix\2.0\bin\artix_env.bat

Code Settings
    Language:  C++
```

Figure 108:*Deployment Profile wizard - Summary panel*

9. Click **Finish** to close this wizard and return to the Artix Designer. Your new Deployment Profile is listed on the Workspace Details panel.

Creating a Deployment Bundle

Overview

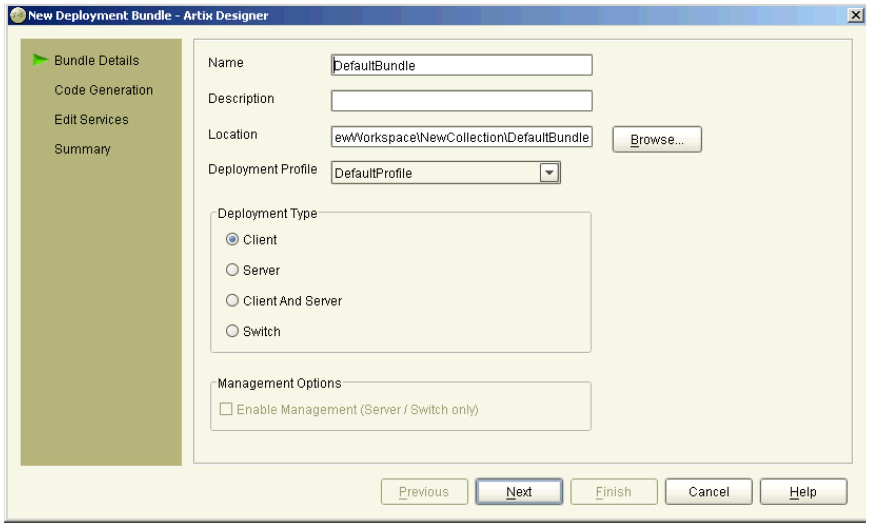
The Deployment Bundle defines the deployment characteristics for a collection, such as the deployment type (client, server, or switch), code generation options, and configuration details. You can also modify the service WSDL for each deployment bundle, if necessary.

You can have as many Deployment Bundles per collection as you like, but you must have at least one Deployment Profile created before you can create a Bundle. Typically you would create a new profile for each different operating system you intended using for deployment.

Procedure

To create a Deployment Bundle:

1. Select the Collection for which you want to create a bundle in the Designer Tree.
2. Select **File | New | Deployment Bundle** from the menu bar to display the New Deployment Bundle wizard, as shown in Figure 109.



Name: DefaultBundle
Description:
Location: ewWorkspace\NewCollection\DefaultBundle
Deployment Profile: DefaultProfile

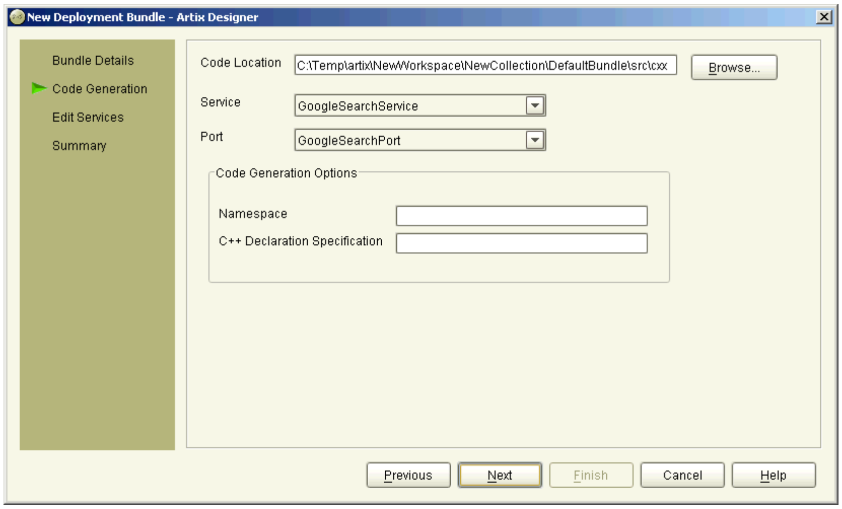
Deployment Type
[✓ Client] Server Client And Server Switch

Management Options
☐ Enable Management (Server / Switch only)

Figure 109:*New Deployment Bundle wizard*

3. Enter a name for this bundle, or accept the default provided.
4. Enter a description for this bundle to help distinguish it from other bundles you may create.
5. Enter a save location for this bundle, or accept the default provided.
6. Select a **Deployment Profile** to reference for this bundle. If there are no profiles listed, you need to create one before you can continue. See "Creating a Deployment Profile" on page 147 for more information.
7. Select the **Deployment Type** from the list provided - options are **client, server, client and server, or switch**.
8. If you have selected a Deployment Type of server or switch, you have the option to enable the Artix **Management Options**. Check the box provided if you want to do so.

   The Management Option generates scripts required to manage your deloyment through a management console.
9. Click **Next** to display the Code Generation panel, as shown in Figure 110.



Code Location: C:\Temp\artix\NewWorkspace\NewCollection\DefaultBundle\src\cxx

Service: GoogleSearchService

Port: GoogleSearchPort

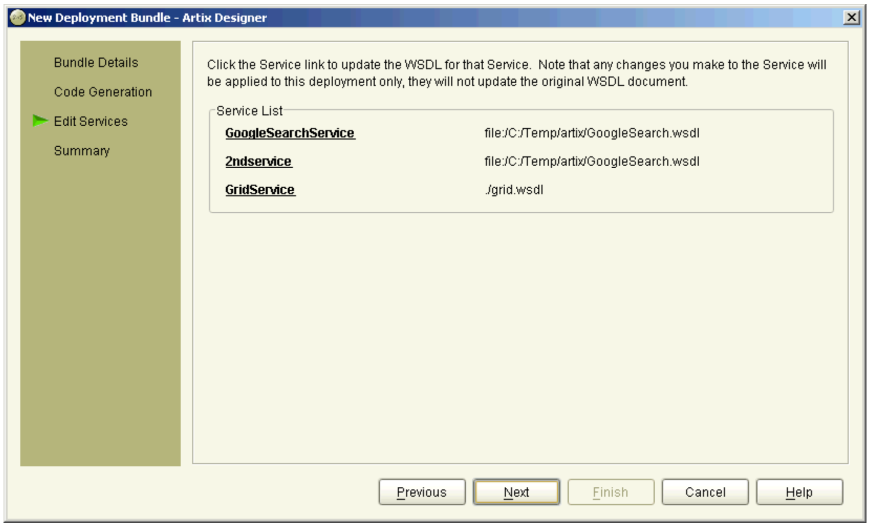
Namespace:

C++ Declaration Specification:

Figure 110: *Deployment Bundle wizard - Code Generation panel*

10. Enter a save location for the code, or accept the default provided.

11. Select a **service** from the list provided.
12. Select a **port** from the list provided.
13. Complete the Code Generation Options as required. Depending on which code you are working with, the options displayed here will differ. Possible options are:
    - Namespace (C++) - The namespace you want to use in the C++ code.
    - Declaration Specification (C++) - If this collection is being built as a DLL on Windows, this specifier is required for the symbols exported from the library.
    - WSDL to C++ Mapping Options (C++) - The options that will be used when mapping the WSDL to C++.
    - Package Name (Java) - The name you want to use for the Java package. Enter a name or accept the default provided.
    - Filename (IDL) - The name of the IDL file that will be generated. Enter a name or accept the default provided.
14. Click **Next** to display the Edit Services panel, as shown in Figure 111.



Click the Service link to update the WSDL for that Service. Note that any changes you make to the Service will be applied to this deployment only, they will not update the original WSDL document.

Service List

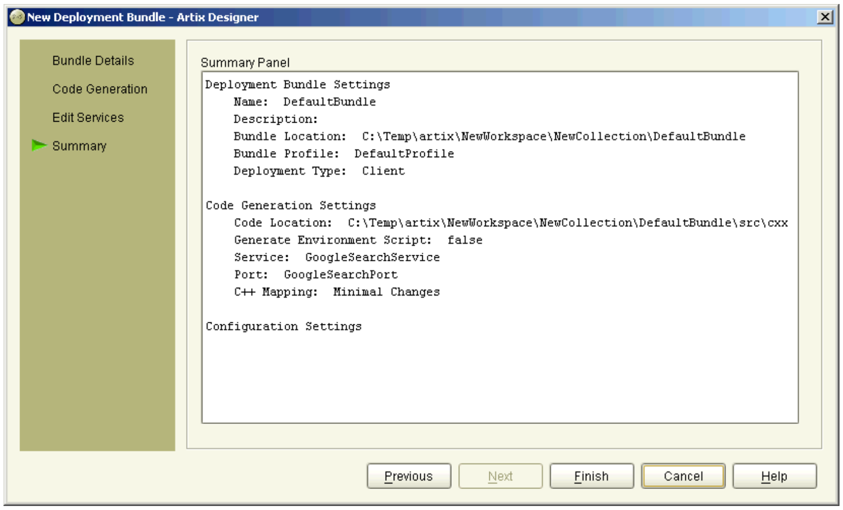
GoogleSearchService	file:/C:/Temp/artix/GoogleSearch.wsdl
2ndservice	file:/C:/Temp/artix/GoogleSearch.wsdl
GridService	./grid.wsdl

Figure 111:*Deployment Bundle wizard - Update Service panel*

15. Click any of the Services links to update them for this deployment.

    Note that any changes made to the Service details from within this wizard will only apply to that bundle; they will not be applied to the WSDL document itself.

16. Click **Next** to display the Summary panel, as shown in Figure 112.



```
Deployment Bundle Settings
    Name:  DefaultBundle
    Description:
    Bundle Location:  C:\Temp\artix\NewWorkspace\NewCollection\DefaultBundle
    Bundle Profile:  DefaultProfile
    Deployment Type:  Client

Code Generation Settings
    Code Location:  C:\Temp\artix\NewWorkspace\NewCollection\DefaultBundle\src\cxx
    Generate Environment Script:  false
    Service:  GoogleSearchService
    Port:  GoogleSearchPort
    C++ Mapping:  Minimal Changes

Configuration Settings
```

Figure 112:*Deployment bundle wizard - Summary panel*

17. Click **Finish** to close this wizard and return to the Artix Designer. The new Deployment Bundle is listed on the Collection Details panel.

    As part of the deployment process, Artix has generated four directories in your specified save location:

    - `src` - contains the generated source code in the language you specified (C++, Java, or IDL)
    - `etc` - contains the configuration information required for the application to run successfully
    - `wsdl` - contains the locally defined WSDL contracts
    - `bin` - contains the environment scripts and the start and stop (UNIX only) scripts

Deploying the Bundle

Overview

Once you have created your Deployment Bundle, the actual deployment of a collection is very simple and quick. Artix deploys the solution based on the information you provided in the bundle, and generates the code, environment scripts, and configuration files in the locations you provided.

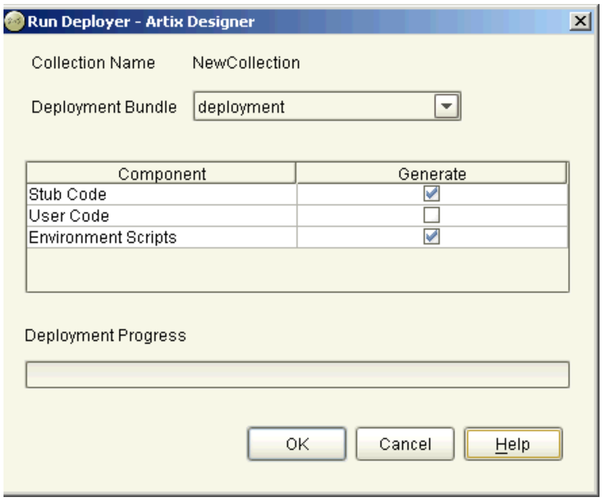
After the collection has been deployed, you can check to see that the code and associated files have been generated successfully - your bundle is now ready to run.

Note that if you make any changes to any of the contents of a collection after deployment, it could invalidate the generated files. For this reason, it is recommended that you redeploy any collection that has been modified.

Procedure

To deploy a collection:

1. Select the collection in the Designer Tree.
2. Select **Tools | Run Deployer** from the menu bar to display the Run Deployer dialog, as shown in Figure 113.



Collection Name NewCollection

Deployment Bundle deployment

Component	Generate
Stub Code	✓
User Code	☐
Environment Scripts	✓

Deployment Progress

Figure 113:*Run Deployer dialog*

3. Select a Deployment Bundle to use for this deployment from the **Deployment Bundle** drop-down list.
4. There are several components already pre-selected in the **Generate** column - if you want to change any of these settings you can do so by selecting and deselecting check boxes. The options provided are:
   - Stub Code - code that marshals and de-marshals the request. This is Required, and is always pre-selected.
   - User Code - generates a template for the implementation code. You will need to complete this code by hand.
   - Environment Scripts - scripts required to set up the Artix development and runtime on the deployment machine.
   - Start/Stop Scripts - scripts to start and stop the server. Only valid for server and switch implementations. For Windows operating systems there will be no stop scripts, as Artix is unable to determine which process to stop from the command line API.
   - Management Scripts - scripts required for integration into the BMC console. Only valid for server and switch implementations.
5. Click **OK** to run the deployer for this bundle.

   You will see a progress indicator, and messages stating things like "Generating Code", "Generating Configuration File". When the deployment process is complete (usually only 3-4 seconds), you will receive a message stating that it is "Finished".
6. Click **Close** to close this dialog and return to the Artix Designer.

Testing the solution

Now that you have deployed the bundle, and generated the code necessary for your application, you need to do some hand coding before you can test the solution.

For help with editing your code, see either:

- *Developing Artix Solutions with C++*; or
- *Developing Artix Solutions with Java*

These books will guide you through the steps required to get your code to a state where it is ready to run.

APPENDIX A

Use Case Examples

Two use cases have been provided to walk you through the Artix Designer, and give you an introduction to the different ways you can perform common tasks.

In this appendix

This appendix discusses the following topics:

Create a Web Service Client - *Fast Track*

Overview

This use case walks you through the procedure for creating a Web Service Client, using the *Fast Track,* or template-based, method. Artix applies defaults for almost every variable, thus making this the quickest way to get your Web Service Client up and running with almost no input from you.

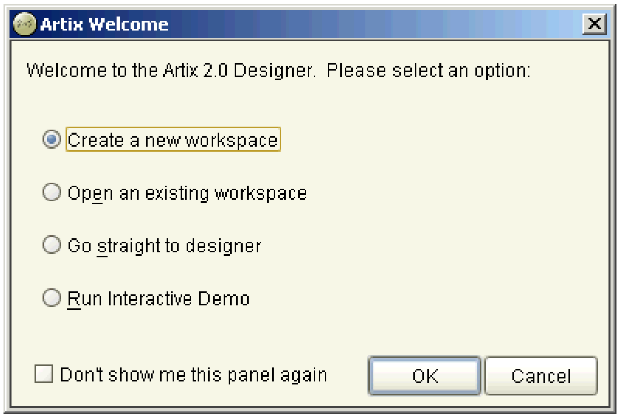
Before you begin

Before starting this procedure, you need:

- Artix installed on your local machine
- A WSDL document (or a URL address) that describes the target service
- A target SOAP/HTTP service to test your client against

Procedure

1. Start Artix from either the icon on your desktop or the Start menu, to display the Getting Started dialog, as shown in Figure 114.


Artix Welcome

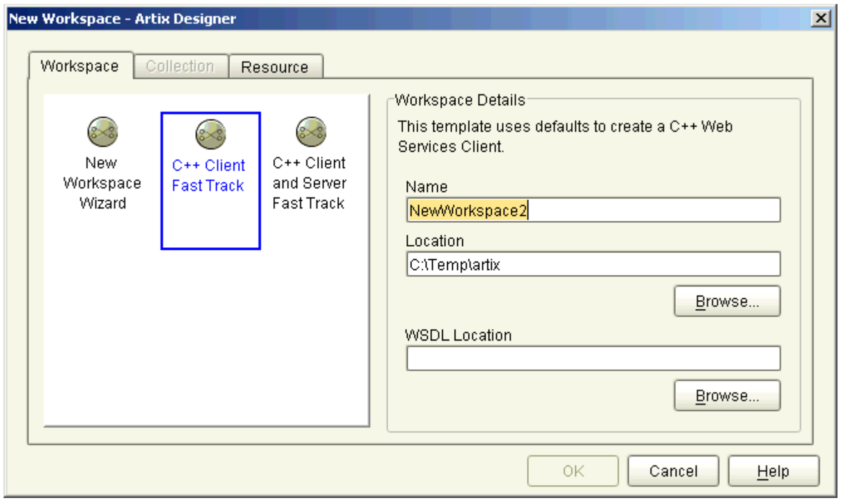
Welcome to the Artix 2.0 Designer. Please select an option:

- [✓ Create a new workspace]
- Open an existing workspace
- Go straight to designer
- Run Interactive Demo

☐ Don't show me this panel again

Figure 114:*Getting Started dialog*

2. Select **Create a New Workspace** and click **OK** to display the New Workspace dialog, as shown in Figure 115.



New Workspace - Artix Designer

Workspace Details

This template uses defaults to create a C++ Web Services Client.

Name: NewWorkspace2

Location: C:\Temp\artix

WSDL Location:

Figure 115:*New Workspace dialog*

3. Select the **C++ Client Client Fast Track** icon.
4. Enter a name and save location for your workspace, or accept the defaults provided. Click **Browse** to navigate to a specific location if you like.
5. Enter the file name or URL for your WSDL file in the field provided, or click **Browse** to navigate to a suitable file.

6. Click **OK** to display the Artix Designer with your Web Service Client contained in the Designer Tree, as shown in Figure 116.

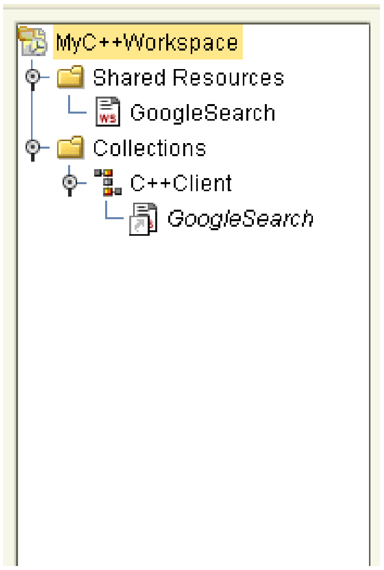


Figure 116:*Artix Designer with Web Service Client*

Behind the scenes

Behind the scenes, Artix has performed the following tasks:

- Created a project directory and project file in the save location you specified
- Imported your WSDL file and added it to the project file
- Created a deployment profile configured for C++ deployment of your client

Your Web Service Client is ready for deployment.

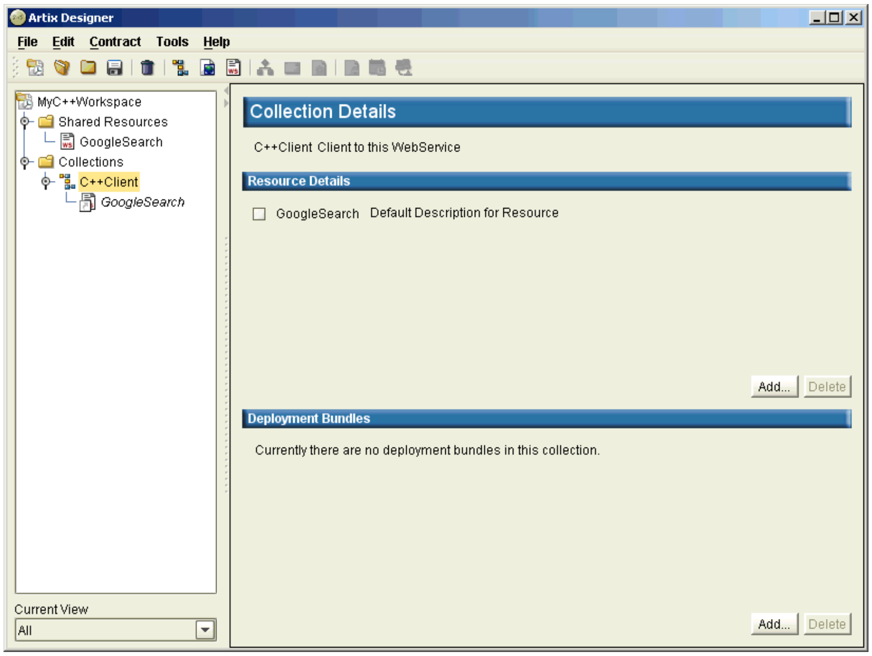
Deploying the client

Now that Artix has automatically created the required deployment profile information, deploying your Web Service Client involves two tasks:

- Create a deployment bundle
- Run the deployer to deploy the solutions

To create a deployment bundle:

1. Select the collection name (C++ Client) in the Designer Tree to display the Collection Details panel, as shown in Figure 117.



C++Client Client to this WebService

GoogleSearch Default Description for Resource

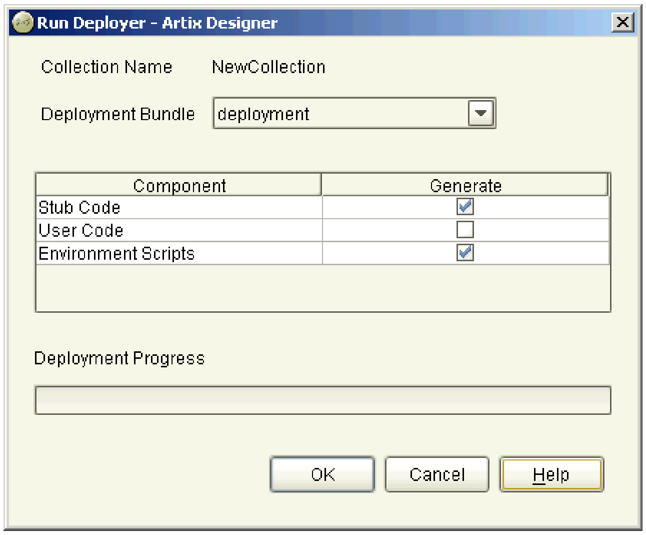
Currently there are no deployment bundles in this collection.

Figure 117:*Collection Details panel*

2. Click the **Add** button under the **Deployment Bundles** section to display the New Deployment Bundle wizard.
3. Move through the wizard, clicking **Next** on every panel to accept the system defaults. For more information about this process, see "Creating a Deployment Bundle" on page 151.
4. Click **Finish** to exit the last panel and return to the Collection Details panel, where your new bundle is now listed.

To deploy your client:

1. Select the collection name (C++ Client) in the Designer Tree, and select **Tools | Run Deployer** from the menu bar to display the Deployer dialog, as shown in Figure 118.



Run Deployer - Artix Designer

Collection Name NewCollection

Deployment Bundle deployment

Component	Generate
Stub Code	✓
User Code	
Environment Scripts	✓

Deployment Progress

Figure 118:*Run Deployer dialog*

2. Click **OK** to deploy your client.

   You will receive a confirmation when the deployment is complete (usually 3-4 seconds).

Create a Web Service Server Using a Wizard

Overview

This use case walks you through the procedure for creating a Web Service server. Unlike the *Fast Track* method described in the previous use case ("Create a Web Service Client - Fast Track" on page 158), this use case walks you through the process of providing the required input for the server via the New Workspace wizard.

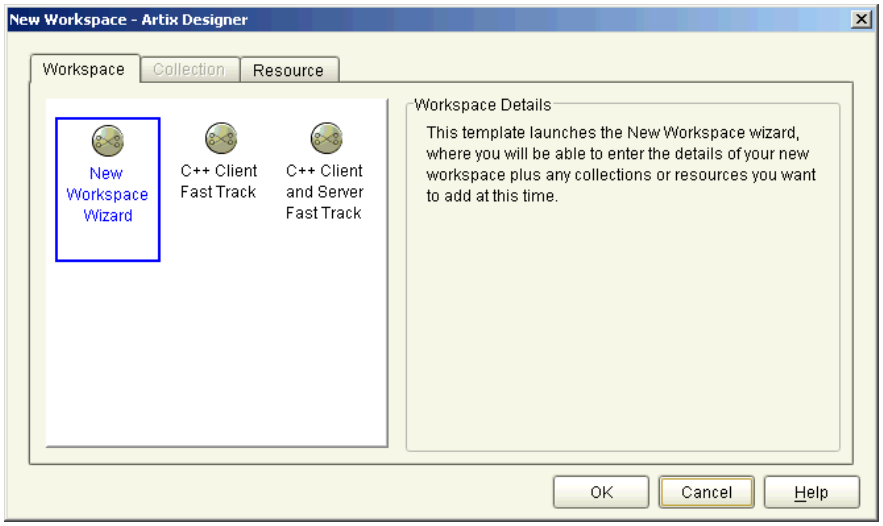
Before you begin

Before starting this procedure, you need:

- Artix installed on your local machine
- A WSDL document (or a URL address) that describes the target service
- A target SOAP/HTTP service to test your client against

Creating the WS Server

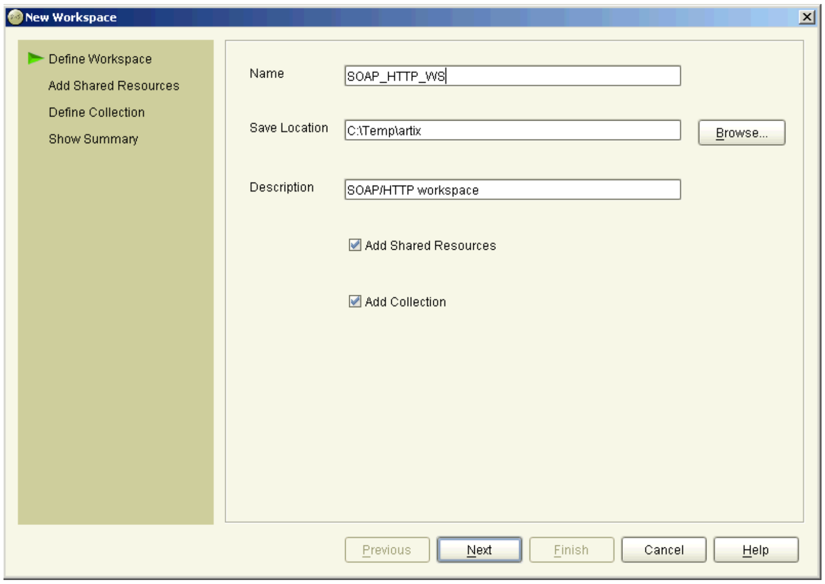
1. From the Getting Started dialog, select **Create a new Workspace** and click **OK** to display the New Workspace dialog, as shown in "New Workspace dialog" on page 163.



This template launches the New Workspace wizard, where you will be able to enter the details of your new workspace plus any collections or resources you want to add at this time.

Figure 119:*New Workspace dialog*

2. Select the **New Workspace Wizard** icon, to display the New Workspace wizard, as shown in Figure 120



Name: SOAP_HTTP_WS
Save Location: C:\Temp\artix
Description: SOAP/HTTP workspace
☑ Add Shared Resources
☑ Add Collection

Figure 120:*New Workspace Wizard*

3. Enter a name for the workspace, or accept the default provided.
4. Select the location where you would like to save your workspace, or accept the default provided.

   Tip: To define a new default save location for all future workspaces, go to the User Preferences dialog (under the Edit menu).
5. Add a description for this workspace in the field provided.
6. Select the **Add Shared Resources** check box if you want to add resources to this workspace that will be shared between all the collections in the workspace.

   Selecting this option will add an extra panel to the wizard for you to enter the shared resource details.

7. Select the **Add Collection** check box if you want to add a collection to this workspace now. Note that this is optional - you can always add a collection later if you don't want to add one now.

   Selecting this option will add an extra panel to the wizard for you to enter the collection details.

8. Click **Next** to display one of the following panels, depending on which check boxes you selected on the first panel:

   - If you checked the Add Shared Resources option, the Shared Resources panel is displayed, as shown in Figure 121. Continue with **step 8.**

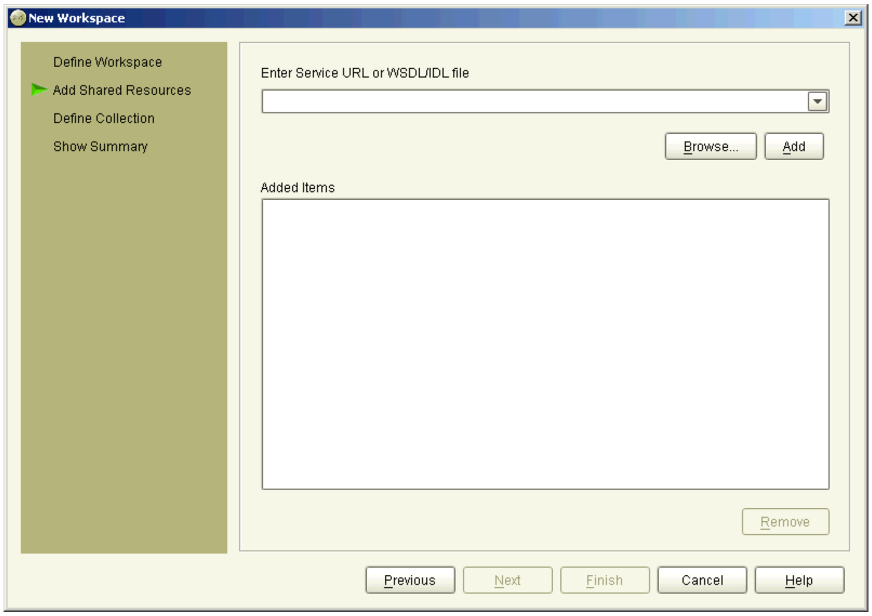


Figure 121:*New Workspace wizard - Shared Resources panel*

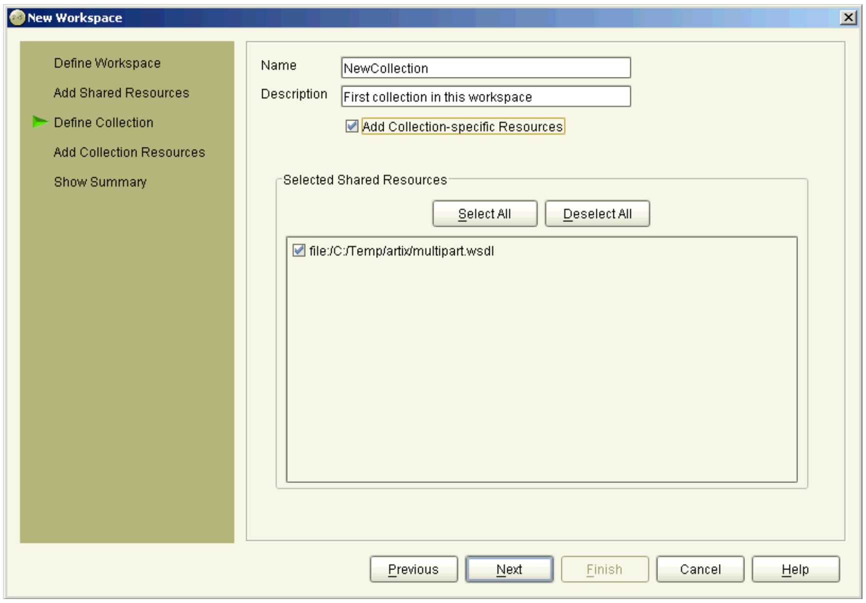
   - If you did not check the Add Resources option but did check the Add Collection option, the Define Collection panel is displayed, as shown in Figure 122. Continue with **step 10**.

- If you did not check either of the options on the first panel, the Summary panel is displayed as shown in Figure 123 on page 167. Continue with **step 14**.

9. Type the location of either a WSDL file or an IDL file in the Enter Service URL or WSDL/IDL file field, or click **Browse** to navigate to the file you would like to use.

   When you have selected a file to use, click **Add** to list it in the Added Items list.

10. Repeat **step 8** as many times as you like to continue adding resources to the list, then click **Next** to display Define Collection panel as shown in Figure 122. If you did not choose to Add a Collection, go to **step 14.**

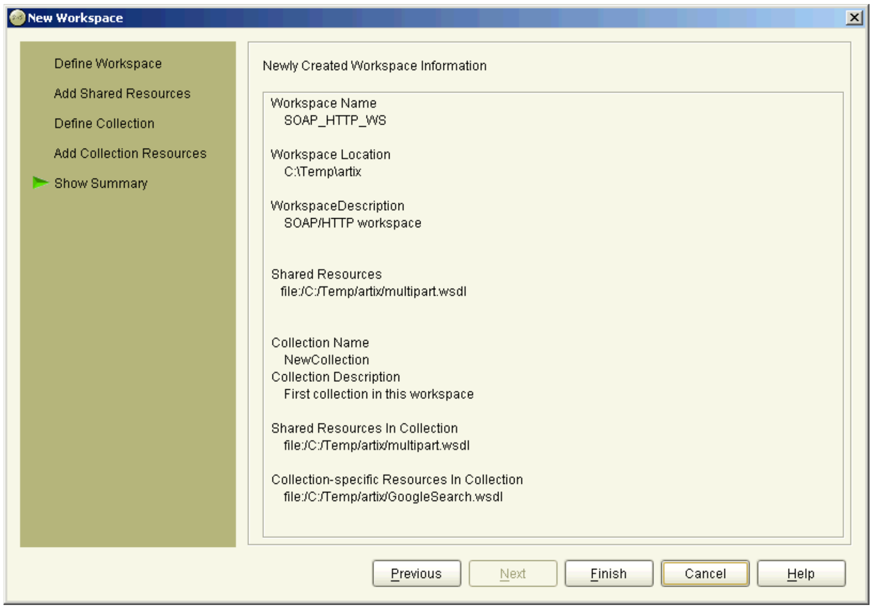

Name: NewCollection
Description: First collection in this workspace
☑ Add Collection-specific Resources

Selected Shared Resources
☑ file:/C:/Temp/artix/multipart.wsdl

Figure 122:*New Workspace wizard - Define Collection panel*

11. Enter a name for the new collection, or accept the default provided.
12. Enter a description for the new collection in the Description field.

13. By default, all shared resources you added to this workspace on the previous panel are selected to be added to this collection. If there are any resources you do not want added, click on their check box to deselect them.
14. Click **Next** to display the display the Summary panel, as shown in Figure 123. This panel lists everything you just specified in the wizard.



Newly Created Workspace Information

Workspace Name
SOAP_HTTP_WS

Workspace Location
C:\Temp\artix

WorkspaceDescription
SOAP/HTTP workspace

Shared Resources
file:/C:/Temp/artix/multipart.wsdl

Collection Name
NewCollection
Collection Description
First collection in this workspace

Shared Resources In Collection
file:/C:/Temp/artix/multipart.wsdl

Collection-specific Resources In Collection
file:/C:/Temp/artix/GoogleSearch.wsdl

Figure 123:*New Workspace wizard - Summary panel.*

15. Click **Finish** to close the wizard and display the Artix Designer, where the Designer Tree displays your newly created workspace.

Deploying the server

Now that you have created your workspace, deploying your Web Service Client involves three tasks:

- Create a deployment profile - this contains machine-specific information that you can use multiple times to deploy as many collections as you have in your workspace. For each machine operating system, however, you would need a separate deployment profile. Turn to "Creating a Deployment Profile" on page 147 for help with this task.
- Create a deployment bundle - this defines the type of deployment you want to perform, such as a client, server, or switch. Thus, you can create a deployment profile, then deploy the same collection as a client and/or a server, and/or a switch just by creating separate deployment bundles. Turn to "Creating a Deployment Bundle" on page 151 for help with this task.
- Run the deployer to deploy the solutions - a very simple (one-dialog) task once the profile and bundle have been created. Turn to "Deploying the Bundle" on page 155 for help with this task.

Glossary

B

Binding

A binding associates a specific protocol and data format to operations defined in a portType.

C

Collection

A group of related WSDL contracts that can be deployed as one or more physical entities such as Java, C++, or CORBA based applications. A collection can also be deployed as a switch process.

Contract

An Artix contract is a WSDL file that defines the interface and all connection (binding) information for that interface. In the context of the Artix Designer, this contract is referred to as a *Resource*.

A contract contains two components: logical and physical. The logical contract defines things that are independent of the underlying transport and wire format: ‘portType’, ‘Operation’, ‘Message’, ‘Type’, and ‘Schema.’

The physical contract defines the wire format, middleware transport, and service groupings, as well as the mapping between the portType ‘operations’ and wire formats, and the buffer layout for fixed formats and extensors, The physical contract defines: ‘Port,’ ‘Binding’ and ‘Service.’

CORBA

CORBA (Common Object Request Broker Architecture) defines standards for interoperability and portability among distributed objects, independently of the language in which those objects are written. It is a robust, industry-accepted standard from the OMG (Object Management Group), deployed in thousands of mission critical systems.

CORBA also specifies an extensive set of services for creating and managing distributed objects, accessing them by name, storing them in persistent stores, externalizing their state, and defining ad hoc relationships between them. An ORB is the core element of the wider OMG framework for developing and deploying distributed components.

E

End-point
The runtime deployment of one or more contracts, where one or more transports and its marshalling is defined, and at least one contract results in a generated stub or skeleton (thus an end-point can be compiled into an application).

M

Marshalling Format
A marshalling format controls the layout of a message to be delivered over a transport. A marshalling format is bound to a transport in the WSDL definition of a Port and its binding. A binding can also be specified in a logical contract portType, which allows for a logical contract to have multiple bindings and thus multiple wire message formats for the same contract.

Message
A WSDL message is an abstract definition of the data being communicated. Each part of a message is associated with defined types. A WSDL message is analogous to a parameter in object-oriented programming.

O

Operation
A WSDL operation is an abstract definition of the action supported by the service. It is defined in terms of input and output messages. An operation is loosely analogous to a function or method in object-oriented programming, or a message queue or business process.

P

Port Type
A WSDL port type is a collection of abstract operations, supported by one or more endpoints. A port type is loosely analogous to a class in object-oriented programming. A port type can be mapped to multiple transports using multiple bindings.

R

Resource
A resource is a WSDL file that define the interface of your Artix solution, and is part of a collection. There can be one or more resources in a collection, and the resources can either be specific to that collection, or shared across several collections (shared resources).

Resources are created either from scratch using the Contract Editor wizards to define the contract elements (types, messages, services, etc), or are based on an existing file that can be either WSDL or IDL. In the case of IDL, the existing file is converted into WSDL and you are given the opportunity to specify options during the conversion.

Routing

The redirection of a message from one WSDL binding to another. Routing rules apply to an end-point, and the specification of routing rules is required for a some Artix services. Artix supports topic-, subject- and content-based routing. Topic- and subject-based routing rules can be fully expressed in the WSDL contract. However, content-based routing rules may need to be placed in custom handlers (C plug-ins). Content-based routing handler plug-ins are dynamically loaded.

S

Service

An Artix service is instance of an Artix runtime deployed with one or more contracts, but no generated language bindings (contrast this with end-point). The service acts as a daemon that has no compile-time dependencies. A service is dynamically configured by deploying one or more contracts on it.

SOAP

SOAP is an XML-based messaging framework specifically designed for exchanging formatted data across the Internet. It can be used for sending request and reply messages or for sending entire XML documents. As a protocol, SOAP is simple, easy to use, and completely neutral with respect to operating system, programming language, or distributed computing platform.

Switch

The implementation of an Artix WSDL service contract.

T

Transport Plug-In

A plug-in module that provides wire-level interoperation with a specific type of middleware. When configured with a given transport plug-in, Artix will interoperate with the specified middleware at a remote location or in another process. The transport is specified in the 'Port' property in of a contract.

Type

A WSDL data type is a container for data type definitions that is used to describe messages (for example an XML schema).

W

Workspace

The Workspace defines the structure of your Artix solution. It is the first thing you need to create when using the Designer, and all of the solution's components are included within it.

A workspace will typically have one or more collections, which in turn contain resources that define your solution's interface. A workspace also contains "shared resources" which are common across one or more collections.

WSDL

WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information.

A WSDL document defines services as collections of network endpoints, or ports. In WSDL, the abstract definition of endpoints and messages is separated from their concrete network deployment or data binding formats. This allows the reuse of abstract definitions: messages, which are abstract descriptions of the data being exchanged, and port types which are abstract collections of operations. The concrete protocol and data format specifications for a particular port type constitutes a reusable binding. A port is defined by associating a network address with a reusable binding, and a collection of ports define a service. Hence, a WSDL document uses the following elements in the definition of network services:

- **Types** - a container for data type definitions using some type system (such as XSD)
- **Message** - an abstract, typed definition of the data being communicated
- **Operation** - an abstract definition of an action supported by the service
- **Port Type** - an abstract set of operations supported by one or more endpoints
- **Binding** - a concrete protocol and data format specification for a particular port type

- **Port** - a single endpoint defined as a combination of a binding and a network address
- **Service** - a collection of related endpoints

Source: Web Services Description Language (WSDL) 1.1. W3C Note 15 March 2001. (http://www.w3.org/TR/wsdl)

X

XML

XML is a simpler but restricted form of SGML (Standard General Markup Language). The markup describes the meaning of the text. XML enables the separation of content from data. XML was created so that richly structured documents could be used over the web.

Index