IBM WebSphere Business Connection

Web Services Overview and Samples

Version 1.1.1

Note!
Before using this information and the product it supports, be sure to read the general information under "Notices" on page 43.

Second Edition (December 2002)

This edition applies to Version 1, Release 1, Modification 1, of *IBM® WebSphere® Business Connection* (5724-D26) and to all subsequent releases and modifications until otherwise indicated in new editions.

IBM welcomes your comments. You can send them to the following address:

IBM Canada Ltd. Laboratory
Information Development
8200 Warden Avenue
Markham, Ontario, Canada L6G 1C7

Include the title and order number of this book, and the page number or topic related to your comment.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

WebSphere Business Connection and Web Services

The Web services documents describe how to use Web services with the IBM(R) WebSphere(R) Business Connection offering. The documents describe how to build and run sample Web services.

Using the documents on a UNIX system

If you are using a UNIX-based server, you must have a Windows(R) client system connected to the server so that you can use CrossWorlds graphical interface tools, such as the System Manager and Business Object Designer. The CrossWorlds artifacts are typically developed on the Windows client system and then deployed on the UNIX system. As part of the installation of CrossWorlds, you were instructed to set up this separate Windows client system. If you have not already set up the Windows client, refer now to the *CrossWorlds System Installation Guide for UNIX*.

The Web services documents include procedures and descriptions of development tasks. For example, a utility for generating CrossWorlds(R) artifacts (CWGenUtility) is described, and samples and a tutorial for building collaborations and a Web service are included. The development tasks described in the documents are designed to be run in a Windows environment only.

How the documents are organized

The documents are organized as follows:

- **Samples** (the remaining sections of this document) provide you with step-by-step instructions for installing and deploying sample services that are included with the WebSphere Business Connection offering. The samples contain an inbound and outbound collaboration, and the document describes the collaborations and their associated artifacts. It tells you how to install the collaborations in CrossWorlds, how to install an Enterprise Application Resource (EAR) file that hosts Web services, and how to use the Test Connectors to operate the collaborations. You'll see how the Simple Object Access Protocol (SOAP) connector is used to send requests.
  - The basic "CrossWorlds Sample" on page 3 shows you how a CrossWorlds-to-CrossWorlds service flows through the various components of Business Connection and CrossWorlds. The sample provides the basic knowledge you'll need to learn more about Web services.
  - "Adding the Web Services Gateway" on page 27 builds on the basic sample. It discusses fundamental concepts of Web services and the Web Services Gateway, such as how the Web Services Invocation Framework (WSIF) and Web Services Descriptor Language (WSDL) files are used to transport and transform information related to service requests. Target services and gateway services are described. You will learn about channels and how to deploy them as well as how to deploy a WSDL file into the Web Services Gateway.

  Note: The samples in this document must be run on a Windows platform. If you are running a Business Connection edition on a UNIX platform, you can read the material to gain an understanding of the concepts described.
- Development Tutorial shows you how to build the collaborations and associated artifacts and deploy them. The tutorial is divided into three parts:

  - In Part 1, you will create a Web service by developing business objects and collaboration templates and objects as well as inbound and outbound maps. You will develop a Java(TM) proxy and create a WSDL file using the CrossWorlds WSGenUtility.
  - In Part 2, you will deploy the Java proxy into a SOAP-enabled application server.
  - In Part 3, you will deploy the WSDL file into the Web Services Gateway.

  Note: The Development Tutorial assumes you have a WebSphere Business Connection edition running on a Windows platform. If you are running a Business Connection edition on a UNIX platform, you cannot follow the procedures in the tutorial; however, you can read the material to gain an understanding of how collaborations are built.
- Advanced Topics covers the following:
  - **CrossWorlds topics** shows you how to install and use a sample in which the SOAP connector is used to invoke a Java-based, non-CrossWorlds Web service. The Web service that is called is based on the remote procedure call (RPC) model of Web services (as opposed to the document-style Web service used in the previous samples). The document discusses the differences in SOAP messaging between the two styles.
  - **Web Services Gateway advanced topics** describes how you can expand on the samples by adding advanced features, such as routing filters. It also provides a description of audit logging.
- Technical Reference provides in-depth information about the relationship of the CrossWorlds artifacts to each other and provides detailed information on naming conventions. It also provides step-by-step instructions for using the WSGenUtility and for creating a SOAP service.

  Note: Most of the material in this section describes methods for developing collaborations and thus pertains only to a Business Connection edition running in a Windows environment.
- Troubleshooting gives you some hints on how to monitor your system and how to catch error messages. Included are discussions of using TcpMon as well as CrossWorlds log4j tracing and WebSphere tracing.

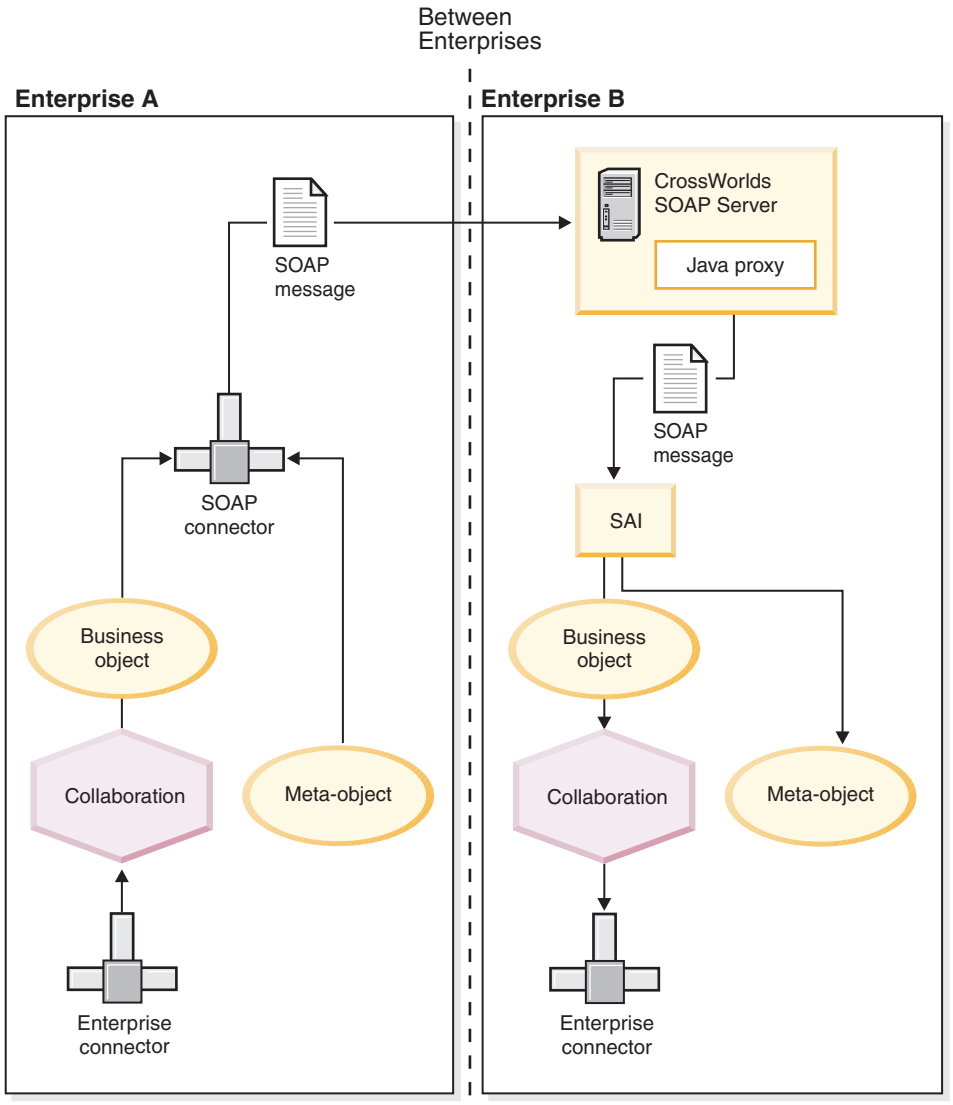
Although the documents provide step-by-step instructions, they assume you have some familiarity with CrossWorlds. To get an overview of CrossWorlds, refer to the *CrossWorlds Technical Introduction*.

CrossWorlds Sample

This section describes the basic CrossWorlds sample that is a part of the WebSphere Business Connection offering. By following the procedures in this section, you will see how CrossWorlds-to-CrossWorlds services are implemented and deployed.

The basic sample enables CrossWorlds-to-CrossWorlds connectivity using SOAP messages. The sample provides CrossWorlds collaborations and associated artifacts that demonstrate how an outbound call from a collaboration via the CrossWorlds SOAP connector can call a Web service deployed on another machine. The target machine must include WebSphere Application Server to host the Web service. The inbound target collaboration can run on any CrossWorlds machine accessible from the WebSphere machine.

The figure below shows the topology of this case. The SOAP connector converts the business object generated by the collaboration on Enterprise A into a SOAP message. The SOAP message is sent to the CrossWorlds SOAP server and through the Server Access Interface (SAI) on Enterprise B to the collaboration on Enterprise B. Details of the way that the inbound collaboration is exposed as a Web service based on a Java proxy class that connects to the target using the SAI are covered later in this document.



Business object going through SOAP connector on one system and arriving at the other system

After you install and work with this sample, you can add to it by following the instructions in "Adding the Web Services Gateway" on page 27.

The Business Connection offering includes a second sample that shows how CrossWorlds collaborations can call an RPC-style Web service that does not use CrossWorlds for processing the request. This sample is described in a later document, Web Services Advanced Topics.

Before you begin

This section tells you how to:

- Install the collaborations in CrossWorlds
- Install an EAR file in WebSphere that hosts the Web services
- Use CrossWorlds Test Connectors to operate the collaborations

In addition, this section will help you understand how the collaborations are designed from a high-level view.

Here are some things you should know, before you begin:

- The sample uses CrossWorlds *projects*, which contain the required business objects, meta-objects, connector definitions, collaboration templates, maps, and collaboration objects, packaged together.
- The Web service for this sample is supplied in an EAR file, bctwssamples.ear. You deploy the file in WebSphere Advanced Edition in the normal manner.
- The bctwssamples.ear file includes one service, which is implemented as a Java proxy class that was produced using the CrossWorlds WSGenUtility. The CrossWorlds JAR files needed by the proxy to connect to the Interchange Server are packaged in the EAR file.
- Additional files, which configure the Java proxies for your computer's particular environment and which provide a lookup source used by the outbound collaboration to determine the service URLs, are also supplied.
- The sample is configured for a single-machine environment. You can experiment with multiple-machine configurations by changing the URLs in the file used by the outbound collaboration to look up Web-service URLs. This will cause the SOAP connector on one machine to call the service on WebSphere running on another machine.
- Before using this sample, the tutorial, or the follow-on sample where you use the Web Services Gateway, you must complete the steps from the *WebSphere Business Connection Installation and Configuration Guide* for installing WebSphere and the Web Services Gateway. In particular, note that CrossWorlds 4.1.1 with the Web Services 1.0.1 (or later) upgrade applied and WebSphere Advanced Edition must be installed before the sample can be used.
- The samples shown in this document work only on a Windows platform. If you have WebSphere Business Connection installed on a UNIX platform, read the document to gain an understanding of how collaborations and Web services work, but do not perform the procedures described.

CrossWorlds-to-CrossWorlds sample theme

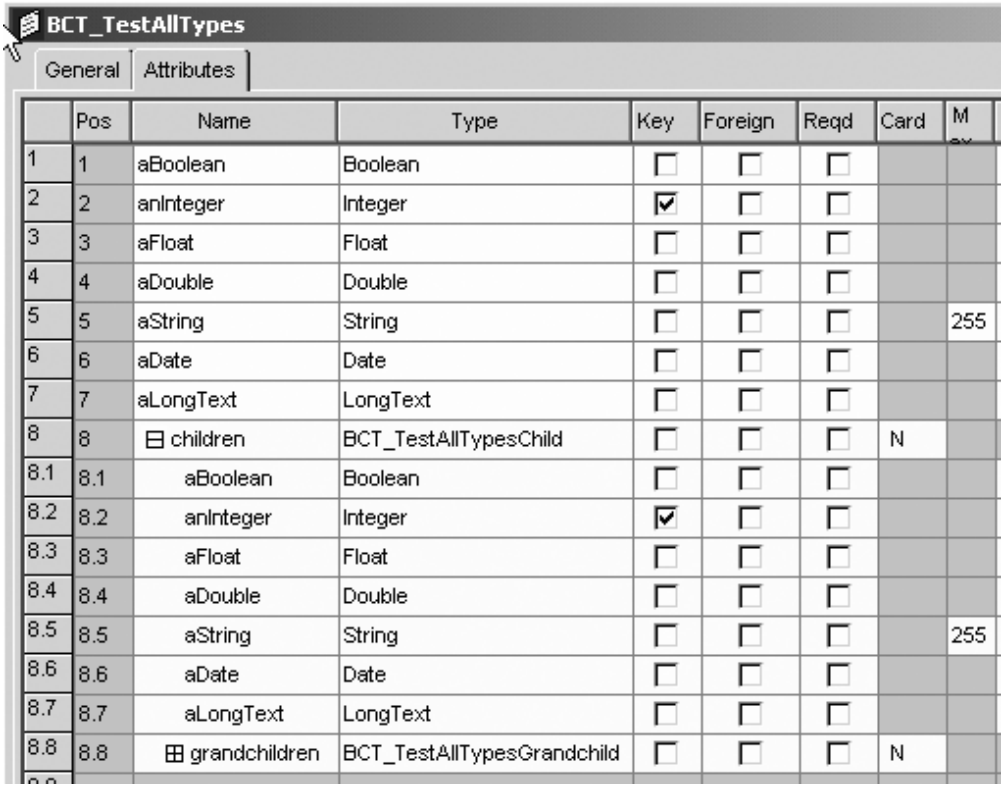
The theme of the CrossWorlds-to-CrossWorlds sample is a test scenario in which all the native types (for example, string and float) supported by CrossWorlds are used in the business object that is sent using the SOAP connector. You will use CrossWorlds Test Connectors to initiate requests from the outbound collaboration and to respond from the inbound collaboration.

Outbound sample description

This section describes the outbound sample generic business object and collaboration object.

Generic business object

The BCT_TestAllTypes outbound sample sends a message composed of attributes that include all the native CrossWorlds types plus arrays of business objects that are themselves made up of attributes that include all the native CrossWorlds types. The BCT_TestAllTypes business object, which is used to compose the request, has the business object definition shown in the following screen:



BCT_TestAllTypes

General | Attributes

	Pos	Name	Type	Key	Foreign	Reqd	Card	M
1	1	aBoolean	Boolean	☐	☐	☐		
2	2	anInteger	Integer	☑	☐	☐		
3	3	aFloat	Float	☐	☐	☐		
4	4	aDouble	Double	☐	☐	☐		
5	5	aString	String	☐	☐	☐		255
6	6	aDate	Date	☐	☐	☐		
7	7	aLongText	LongText	☐	☐	☐		
8	8	⊟ children	BCT_TestAllTypesChild	☐	☐	☐	N	
8.1	8.1	aBoolean	Boolean	☐	☐	☐		
8.2	8.2	anInteger	Integer	☑	☐	☐		
8.3	8.3	aFloat	Float	☐	☐	☐		
8.4	8.4	aDouble	Double	☐	☐	☐		
8.5	8.5	aString	String	☐	☐	☐		255
8.6	8.6	aDate	Date	☐	☐	☐		
8.7	8.7	aLongText	LongText	☐	☐	☐		
8.8	8.8	⊞ grandchildren	BCT_TestAllTypesGrandchild	☐	☐	☐	N	

BCT_TestAllTypes business object attributes, such as name and type

Outbound collaboration object

The outbound collaboration object From port is bound to BCTSampleConnector1, which is associated with the Test Connector. The Test Connector allows you to create an instance of BCT_TestAllTypes.Retrieve, edit its fields, and send it to the From port.

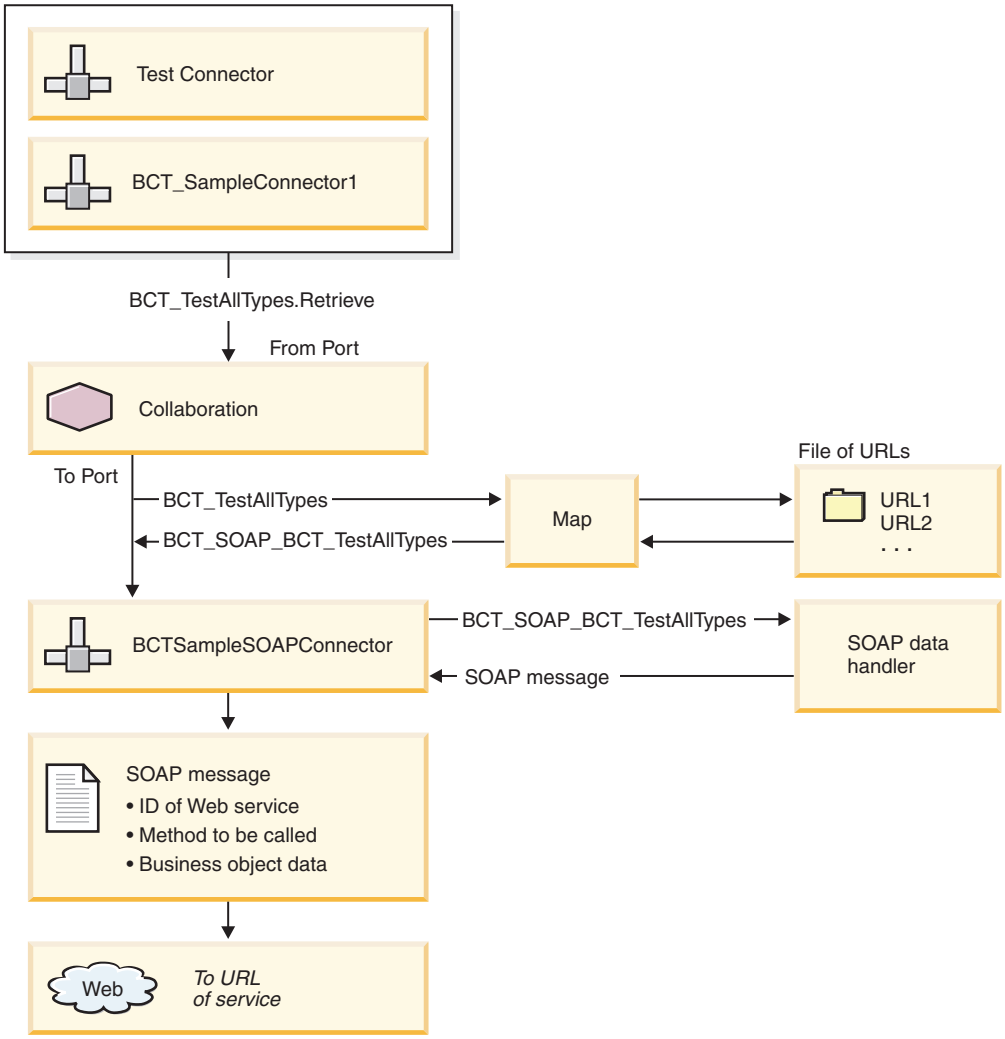
The sample outbound collaboration object subscribes to BCT_TestAllTypes.Retrieve, so it is triggered when you send this event to the From port.

The outbound collaboration object flow passes the business object to its To port, which is bound to BCTSampleSOAPConnector. An outbound map from the collaboration port to the BCTSampleSOAPConnector maps the BCT_TestAllTypes (the generic business object) into BCT_SOAP_BCT_TestAllTypes, the correct application-specific business object expected by the BCTSampleSOAPConnector. This map uses a Java class that was written for this sample. The Java class reads an external file to determine the URL of the server to handle the SOAP message it reads. The application-specific business object used by BCTSampleSOAPConnector includes an attribute for this URL.

Meta-objects used by the BCTSampleSOAPConnector define how the business object is to be converted into a SOAP message by the SOAP data handler. One of the meta-objects specifies the ID of the Web service and the method name to be called on the server. This information is included in the SOAP message formed by the SOAP data handler, along with the business object data.

The SOAP data handler sends the SOAP message to the URL.

The flow described above, from Test Connector to the URL, is illustrated in the following diagram:



SOAP data handler converts the business object to a SOAP message

Because SOAP/HTTP is a synchronous protocol, the BCTSampleSOAPConnector waits for the response. The inbound processing will be discussed in more detail in a later section.

When a response is sent, the flow is reversed. When the BCTSampleSOAPConnector receives the response message, the SOAP data handler uses meta-objects to determine how to convert the response message to business-object form. There is a map from the application-specific business object to the generic business object.

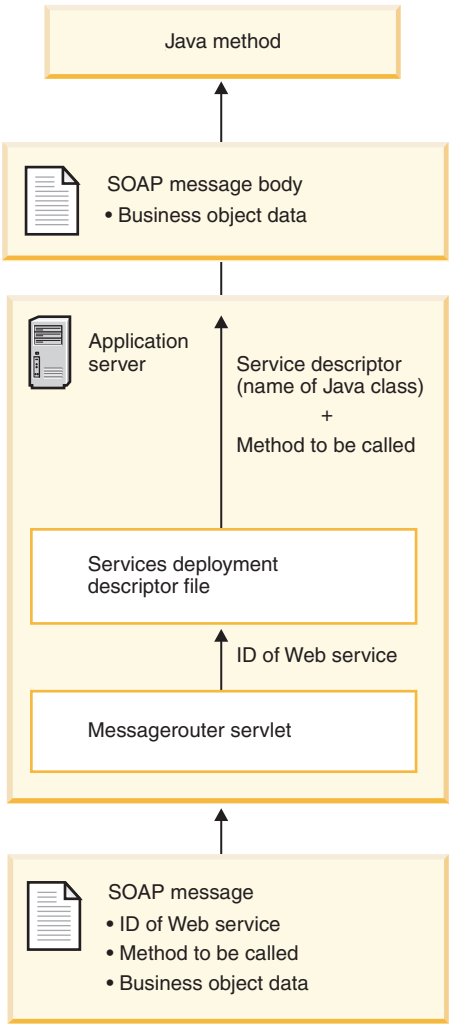
After receiving the BCT_TestAllTypes business object, the collaboration flow passes it to a port bound to BCT_SampleConnector1. You use a Test Connector instance associated with this connector to accept requests sent to it. In this way, you can see the response.

To complete the flow, you use the Test Connector to reply success.

Inbound sample description

The target URL used by the BCTSampleSOAPConnector is the Apache SOAP messagerouter servlet that is deployed in the application server. The deployment of the Apache SOAP servlets in the application server provides the ability to receive and process the SOAP messages. The messagerouter servlet extracts the ID of the service to invoke by looking in the SOAP message. This ID is used to find the service descriptor in the application server's Web-services deployment descriptor file. The service descriptor tells the application server the name of the Java class to load to perform the service. The method name to invoke is also extracted from the SOAP message. With this information, the application server can invoke a Java method, passing it the SOAP message Body and its business object information, for processing.

The flow of the SOAP message to the Java method, described above, is illustrated in the following diagram:



How the ID of the service in the SOAP message is used to find the Java method to invoke

In this sample, the Java class is generated by the CrossWorlds utility WSGenUtility. The utility produces a proxy class for a particular collaboration object, port,

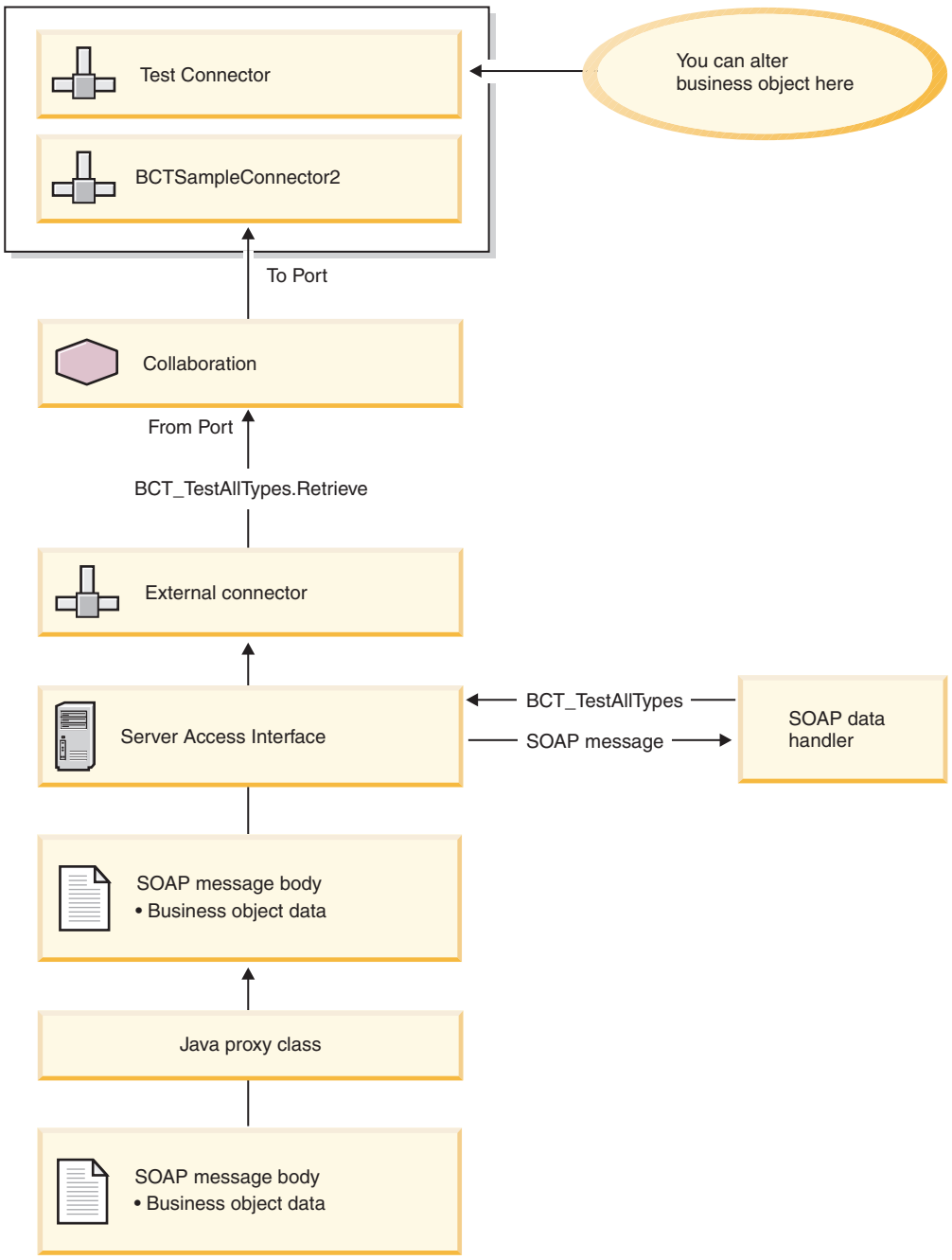
business object, and verb. In the case of this sample, the proxy class was generated for the inbound collaboration triggered by a BCT_TestAllTypes.Retrieve event.

The SOAP message that was sent is a representation of a BCT_TestAllTypes.Retrieve event. The message is passed to the proxy class, which passes it via the CrossWorlds Server Access Interface (SAI) to the InterChange Server, where the inbound collaboration object is running. The inbound collaboration template is defined with its From port set to subscribe to the BCT_TestAllTypes.Retrieve event. The From port is bound to the External Connector, so it will receive events arriving through the Server Access Interface.

The SOAP data handler uses meta-objects to determine how this SOAP message is to be converted to a business object. After the data handler produces the business object, the inbound collaboration object From port that is bound to the External Connector is triggered with this event.

The inbound collaboration flow passes the triggering object to a port that is bound to BCTSampleConnector2. When you run the sample, you will use a Test Connector associated with this connector to accept requests. The Test Connector permits you to alter the request and send it as a response.

The flow of the SOAP message to the Test Connector, described above, is illustrated in the following diagram:



How the object passes to the Test Connector

After responding, the flow is reversed. The response BCT_TestAllTypes.Retrieve is sent back to the data handler, which prepares it as a SOAP message. This message is passed back to the proxy class, which provides it as a response to the caller (the BCTSampleSOAPConnector of the outbound collaboration).

Generating CrossWorlds artifacts

You will notice that building a set of collaborations that communicate with the SOAP connector requires a large number of related business objects, meta-objects,

and maps. A naming utility called CWGenUtility is included with the samples to help you deal with this complexity. This utility:

- Takes the generic business object name, the verb for the triggering event, and a base-name that characterizes the inbound and outbound processes
- Produces a naming text file that uses a naming convention to provide the necessary names for the outbound and inbound collaborations and their associated objects
- Produces an import file used to import the business-object definitions needed to build the Web-services scenario in CrossWorlds
- Produces Java-code templates for the code needed in the maps used by the inbound and outbound collaboration objects
- Produces a file with names needed when the CrossWorlds WSGenUtility is used to produce the proxy and WSDL files for the inbound collaboration

The names in the samples were produced by CWGenUtility. Its usage is discussed in detail in the Web Services Technical Reference.

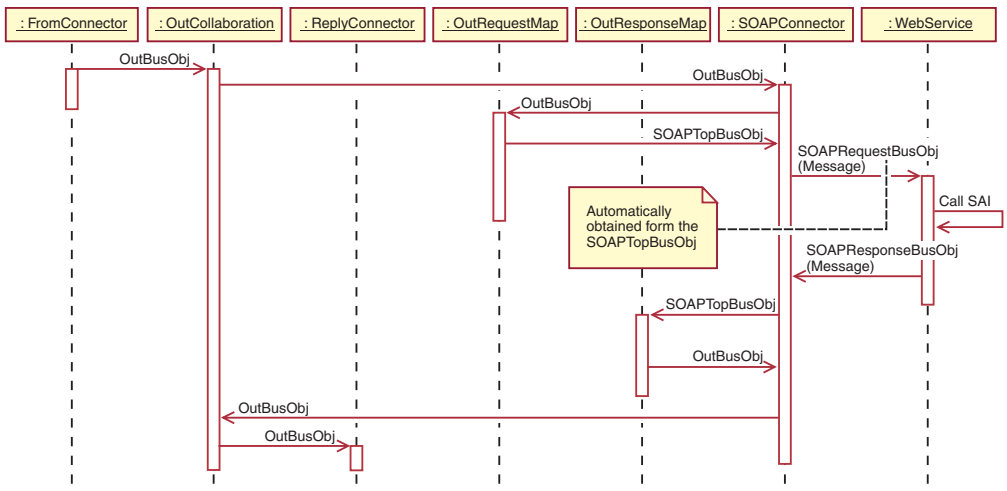
Collaboration flows

Outbound and inbound sequence diagrams in this section show you how the samples work. Although the names of the sample artifacts are used, you can consider these to be generic diagrams that are common to all SOAP messaging flows.

The names in the sequence diagrams are tied to the actual sample by the list of artifacts that follow each diagram. For example, for the outbound flow, the From Connector is BCTSampleConnector1. Use the artifact lists and the sequence diagrams together to understand how the samples work.

Outbound flow

The following diagram depicts the outbound flow to and from the SOAP connector. The flow is described in later sections of the document.



Outbound flow to and from the SOAP connector

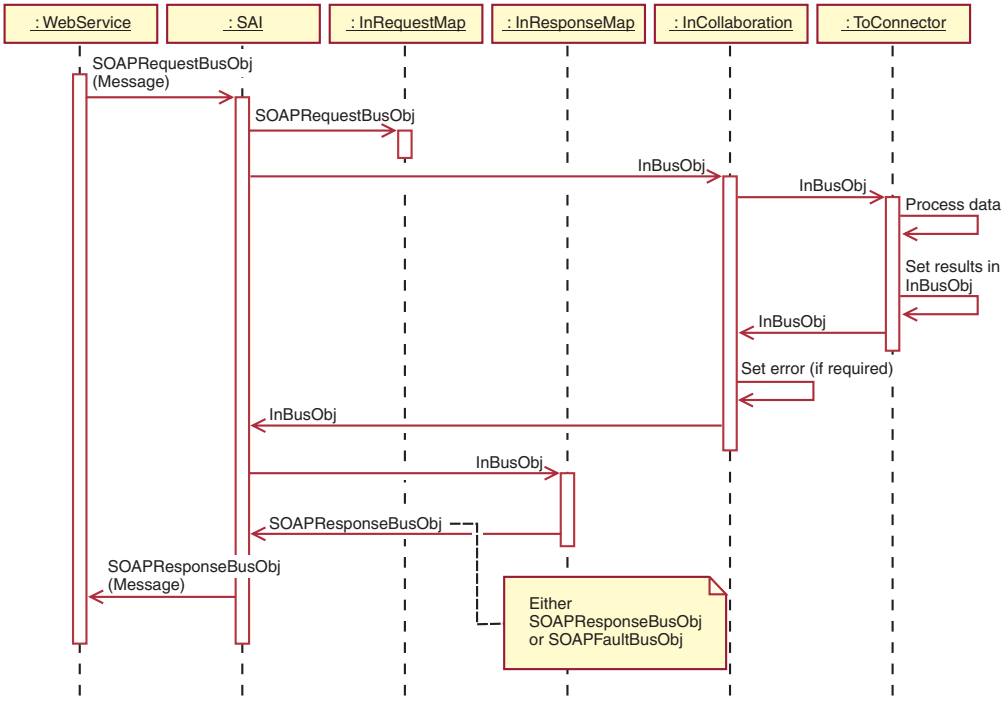
The following table lists the names of the projects, templates, collaborations, business objects, meta-objects, and maps:

Table 1. Outbound artifact list

Artifact type	Artifact name
Project	BCT_TestAllTypes
Template	BCT_TestAllTypesOutbound
Out Collaboration	BCTSampleConnector1_to_BCTSample SOAPConnector_BCT_TestAllTypes_Outbound
OutCollaboration Input	See GBO OutBusObj
FromConnector	BCTSampleConnector1
ReplyConnector	BCTSampleConnector1
ToConnector	BCTSampleSOAPConnector
OutRequest Map	BCT_TestAllTypes_to_BCT_SOAP_BCT _TestAllTypes
OutRequest Map Input BusObj	See GBO OutBusObj
OutRequest Map Output BusObj	See ASBO TopSOAPBusObj
OutResponse Map	BCT_SOAP_BCT_TestAllTypes_to_BCT _Test AllTypes
OutResponse Map Input BusObj	See ASBO TopSOAPBusObj
OutResponse Map Output BusObj	See GBO OutBusObj
GBO OutBusObj	BCT_TestAllTypes
ASBO TopSOAPBusObj	BCT_SOAP_BCT_TestAllTypes
Out SOAP MO Request	MO_Client_BOtoSOAP_BCT_TestAllTypes _Request_Retrieve
Out SOAP MO Response	MO_Client_SOAPtoBO_BCT_TestAllTypes _Response_Retrieve
Out SOAP MO Fault	MO_Client_SOAPtoBO_BCT_SOAP_BCT _TestAllTypes_Fault_Fault_Retrieve

Inbound flow

The following diagram depicts the inbound flow to and from the SAI. The flow is described in later sections of the document.



Inbound flow through the SAI

The following table lists the names of the projects, templates, collaborations, business objects, meta-objects, and maps:

Table 2. Inbound artifact list

Artifact type	**Artifact name**
Project	BCT_TestAllTypes
Template	BCT_TestAllTypesInbound
In collaboration	SAI_to_BCTSampleConnector2_BCT _TestAllTypesInbound
InCollaboration Input	See GBO InBusObj
From connector	SAI
To connector	BCTSampleConnector2
InRequest Map	None
InRequest Map Input BusObj	N/A
InRequest Map Output BusObj	N/A
InResponse Map	Poly_BCT_TestAllTypes_to_BCT_SOAP_BCT _TestAllTypes_Wrapper
InResponse Map Input BusObj	See GBO_OutBusObj

Table 2. Inbound artifact list (continued)

Artifact type	Artifact name
InResponse Map Output BusObj	1) See ASBO SOAPResponseBusObj 2) See ASBO SOAPFaultBusObj
InResponse Sub Maps	BCT_TestAllTypes_to_BCT_SOAP_BCT _TestAllTypes_Fault
GBO InBusObj	BCT_TestAllTypes
ASBO SOAPResponse BusObj	BCT_TestAllTypes
ASBO SOAPFaultBusObj	BCT_SOAP_BCT_TestAllTypes_Fault
In SOAP MO Request	MO_Service_SOAPtoBO_BCT_TestAllTypes _Request_Retrieve
In SOAP MO Response	MO_Service_BOtoSOAP_BCT_TestAllTypes _Response_Retrieve
In SOAP MO Fault	MO_Service_BOtoSOAP_BCT_SOAP_BCT _TestAllTypes_Fault_Fault_Retrieve

CrossWorlds sample installation files

The CrossWorlds samples are installed (as a zip file) as part of a Business Connection installation.

1. Open a command window or prompt.
2. Change to the following directory:

   ***<BCT_HOME>*\samples\CW**
3. Locate the file bctws.zip and unzip it.

The remainder of this document assumes that the samples are installed in the C drive of your system.

The following files and directory structure should be present on your machine:

```
\bctws
+---cw
|   | BCT_SOAP_StockquoteService_getCompanyQuotes_1.BO
|   | BCT_TestAllTypes_1.BO
|   | BCT_SampleData.BO
|   | bct_ws_configuration.txt
|   | start_BCTSampleSOAP.bat
|   |
|   +---CWGenUtil
|   |   | runCWGenUtility.bat (and many supporting files)
|   |   |
|   |   \---Generated
|   |
|   \---imports
|         BCT_WS_Samples.in
|         BCT_WS_SampleDH.in
|         BCT_Basic_Tutorial.in
|         cw_wsgenutility_inputfile.txt
|
\---was
    |
```

```
  +---cw
| |   BCT_TestAllTypes_Retrieve.cfg
| |   BCT_SampleData_Create.cfg
| |
| +---logs
| |
| \---wsdl
|      BCT_TestAllTypes_Retrieve.wsdl
|      BCT_SampleData_Create.wsdl
|
\---wsgw
  \---logs
```

Editing start_server.bat

Update the CrossWorlds file named start_server.bat, which is in <CrossWorlds>\bin. This is necessary because the Interchange Server needs to have access to the classes that are used in custom maps used in the sample. You will also add the JAR files used for basic SOAP messaging support to the ICS classpath. Refer to the CrossWorlds Web Services connector documentation for more information.

Be very careful making these changes as you follow the instructions. Back up the original start_server.bat before beginning.

1. Edit <crossworlds>\bin\start_server.bat by adding the following lines just before the line where environment variable JCLASSES is set. *(Note that each set command must be on a single line in the batch file.)*:

   ```
   rem Set a variable with the directory where the samples were unzipped
   rem This example uses c:\wbc\bctws\cw
   set BCT_DIR=c:\bctws\cw
   rem Set a variable to point to the URL look up file used by the map
   set BCT_CONFIG="%BCT_DIR%"\bct_ws_configuration.txt
   rem Set a variable to point to jar files used by the sample maps
   set BCT_JAR="%BCT_HOME%"\lib\bctwssamples.jar;"%BCT_HOME%"\lib\bctcocommon.jar

   rem Set a variable to point to jar files needed for all SOAP activities
   set WEBSERVICES="%CROSSWORLDS%"\connectors\SOAP\Dependencies\
   soap.jar;"%CROSSWORLDS%"\connectors\SOAP\Dependencies\
   activation.jar;"%CROSSWORLDS%"\connectors\SOAP\Dependencies\mail.jar
   ```

2. Add the following to the very end of the existing line which begins with set JCLASSES=. *(Note that there is a semicolon at the beginning of the text you are adding!)*

   ```
   ;%BCT_JAR%;%WEBSERVICES%
   ```

3. Finally, modify the line that starts the ICS by adding a definition for BCT_WS_CONFIGURATION. You can copy this:

   ```
   -DBCT_WS_CONFIGURATION=%BCT_CONFIG%
   ```

   to the location shown.

   ```
   "%CROSSWORLDS%\bin\java" -server -DBCT_WS_CONFIGURATION=%BCT_CONFIG%
   ```

Importing the CrossWorlds project

This section describes how to import the CrossWorlds project.

Starting CrossWorlds

Start the CrossWorlds components, in the order shown:

1. Start the CrossWorlds MQ Listener from Windows by clicking **Start > Programs** > **CrossWorlds** > **MQSeries** > **Start Listener.**
2. Start the CrossWorlds InterChange Server (ICS) from Windows by clicking **Start > Programs** > **CrossWorlds** > **Server and Tools** > **InterChange Server.**
3. Start the CrossWorlds System Manager (CSM) from Windows by clicking **Start > Programs** > **CrossWorlds** > **Server and Tools** > **CrossWorlds System Manager.**
4. Connect from CSM to ICS.

Importing projects

Two CrossWorlds import files provide the sample environment. These files are located in: \bctws\cw\imports

The first file is named BCT_WS_Samples.in. This file contains all the CrossWorlds sample artifacts except for two system meta-objects. The second file is named BCT_WS_SampleDH.in. The two system meta-objects are contained in this file.

In the procedure that follows, you will import the BCT_WS_Samples.in file, and then you will back up the two system meta-objects using the CrossWorlds System Manager. Finally you will import the BCT_WS_SampleDH.in file. By following these steps carefully, you will be able to run the samples and later restore your system using the backed-up versions of the system meta-objects.

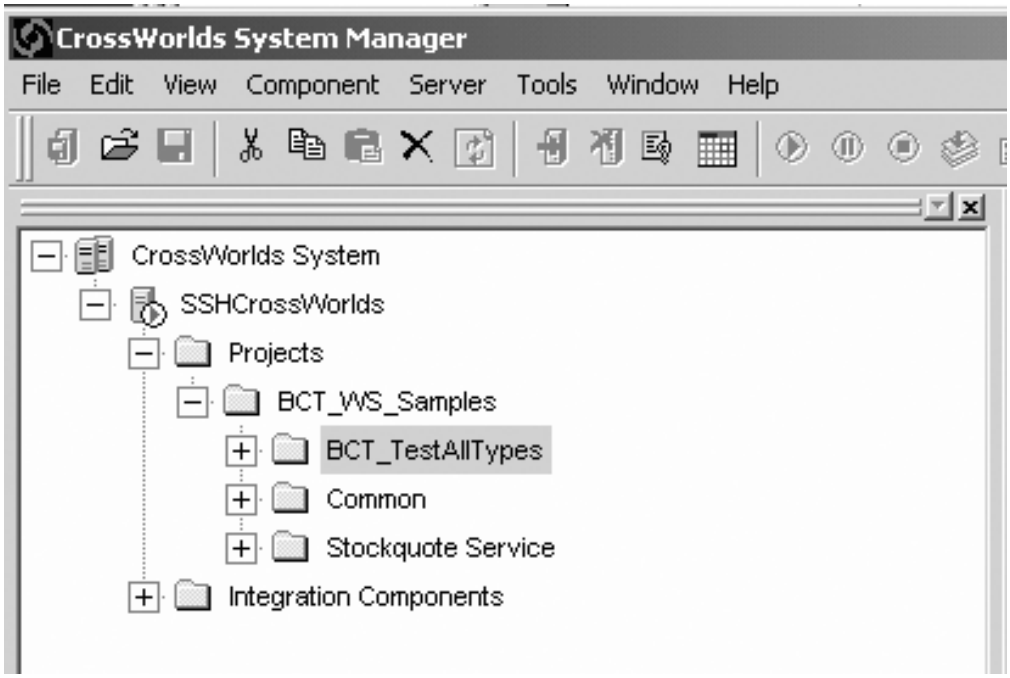
To prepare CrossWorlds to use the samples:

1. Click your InterChange Server name in the CrossWorlds System Manager to select it.
2. Click **File > Open from File**.
3. Select and import **BCT_WS_Samples.in** from the following directory: bctws\cw\imports
4. Back up the business objects named **MO_DataHandler_Default** and **MO_Server_DataHandler**. To do this:
   a. Locate the objects under the Integration Components/Business Objects heading in the CSM.
   b. Select each and then select **File > Save as > To Server**.
   c. Save each with a name you will remember, such as SavedMO_DataHandler_Default and SavedMO_Server_DataHandler, because you will need to restore them later when you are finished using the samples.
5. Click **File > Open from File.**
6. Select and import **BCT_WS_SampleDH.in** from the following directory: \bctws\cw\imports
   This provides the new copies of MO_DataHandler_Default and MO_Server_DataHandler that are used by the sample.
7. Later, when you want to restore the original MOs, consider the following approach:
   a. Back up the sample meta-objects by using **Save as** to other names, such as SampleMO_DataHandler_Default and SampleMO_Service_DataHandler.
   b. Then delete MO_DataHandler_Default and MO_Server_DataHandler. Note that you will not be able to delete MO_DataHandler_Default unless you remove all references that connectors have to it in their Supported Business Objects pages.

c. Finally, restore the original files by using **Save as**.

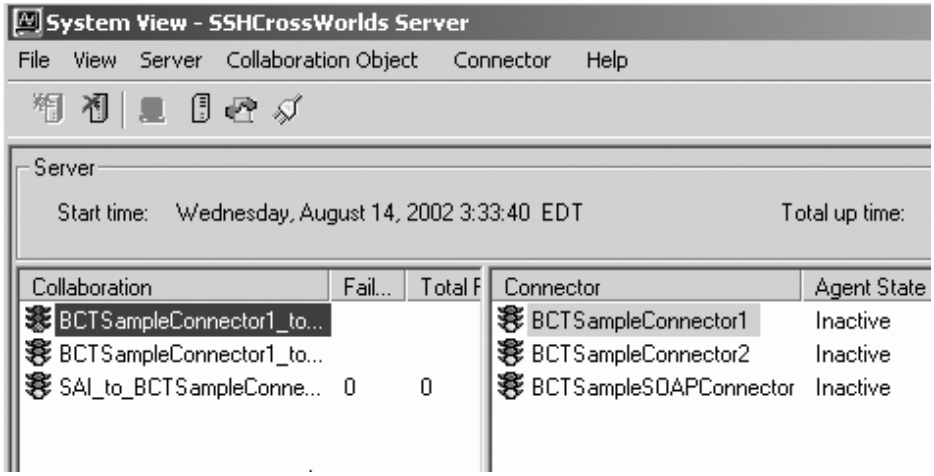
You might see errors in the map imports when you import for the first time. These occur because all of the maps have not been compiled yet. The errors will be corrected when you compile the maps.

After the files have been imported, you will see a new project in the CrossWorlds System Manager window, as shown below:



CrossWorlds System Manager screen displaying the new project

1. Expand the project, and then expand its Collaborations Template folder and Maps folder.
2. Compile the collaboration templates and maps in order, as they are listed (to ensure that the sub maps used are ready when they are needed), by right-clicking on their names and clicking **Compile**.
   If you have trouble compiling classes, be sure that you modified start_server.bat correctly. Confirm that the bctcocommon.jar and bctwssamples.jar files are located in the following directory, because the modifications to start_server.bat require them to be in this directory: *<BCT_HOME>*\lib
3. When you are finished, shut down the InterChange Server and restart it.
4. Reattach the CrossWorlds System Manager to the InterChange Server.
5. Click **Server > System view** and confirm (as indicated by the green lights in the following screen) that all the connector and collaboration objects are started.



Collaborations and connectors and their status

Note that you will have more collaborations and connectors than what is shown in the screen above. The items in the screen are the ones used by the sample.

Configuring WebSphere

To configure WebSphere, you create an application server and then deploy the application.

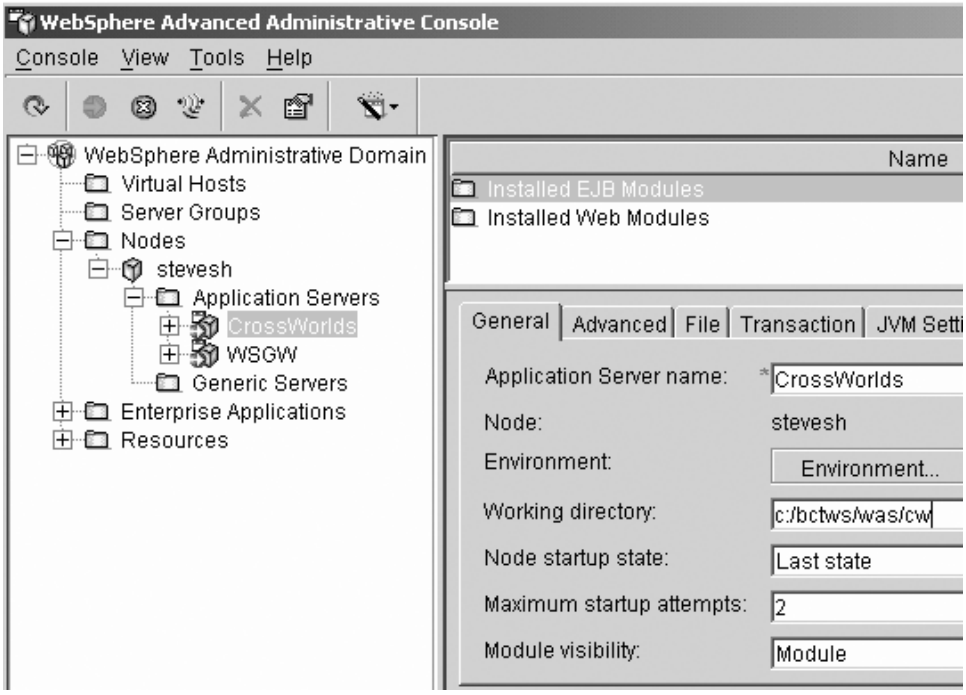
To display the WebSphere Admin Console:

1. Start the IBM Http Server Service from the Windows Services Window.
2. Start the WebSphere Admin Service from the Windows Services Window.
3. Start the WebSphere Administrative Console from Windows by clicking **Start > Programs > IBM WebSphere > Application Server V4.0 AE > Administrator's Console**.

Creating the CrossWorlds application server

To create the CrossWorlds Application Server:

1. Create a new application server called CrossWorlds App Server.
2. Change the working directory (as shown on the General tab in the following screen) to where you have unzipped the samples bctws.zip. For the CrossWorlds application server, use *your_drive*:/bctws/was/cw. Note that the working directory must be set correctly for the sample to work because there are configuration files in the directory that are used by the application server.



WebSphere console showing the correct working directory

3. Under the **File** tab, change the names of the standard output and error logs to be **logs/stdout.txt** and **logs/stderr.txt**. This causes the log files to be located in the logs directory under the working directory of the application server.
4. Next, open the BCT_TestAllTypes_Retrieve.cfg file for edit. This file is located in: \bctws\was\cw
5. Type the correct information for ICSName, UserName, PassWord, and IORFileName. Make sure you match the format of the original IORFileName field with regard to backslashes.

```
#Configuration properties for BCT_TestAllTypes_Retrieve.java
#Tues April 16 18:23:41 EDT 2002
CollabName=SAI_to_BCTSampleConnector2_BCT_TestAllTypes
PassWord=null
ICSName=bctws_your_CrossWorlds
CWLDVersion=4.x
UserName=admin
MimeType=xml/soap
CollabPort=From

IORFileName=D\:\\CrossWorlds\\bctws_your_CrossWorldsInterchangeServer.ior
```

6. If CrossWorlds is not running on the same machine as WebSphere, copy the IOR file from the CrossWorlds machine to a local drive on the WebSphere machine and refer to it on the local drive.
   Note: WebSphere will not be able to load the IOR file from a mapped network drive. That is why you must copy it to a local drive.
   The proxy class, which is included in the EAR file, reads this configuration when it is called to perform the Web service. It uses the information to attach to your InterChange Server.

Deploying the enterprise application

The next step is to deploy bctwssamples.ear from the following directory: *<BCT_HOME>*\lib

Take the defaults for each installation step, selecting the CrossWorlds application server if given a choice.

1. Stop the CrossWorlds application server if it is running.
2. Regenerate the Web-server plugin.
3. Start the application server.

Testing WebSphere

To test the WebSphere setup:

1. Open a Web browser.
2. Type the following address:

   `http://<localhost>/bctwssamplesweb/servlet/messagerouter`

   You should receive the following message:

   ```
   Sorry, I don't speak via HTTP GET - you have to use HTTP POST to talk to
   me.
   ```
3. Type the following address:

   `http://<localhost>/bctwssamplesweb/admin`

   This time, you see a screen titled XML-SOAP Admin.
4. Click **List all services**.

   You see at least one service (urn:ibmwsgw#BCT_TestAllTypes_Retrieve) listed. Note that because of the # symbol in this name, you cannot view the deployment information by clicking on it. The # symbol is required by the Web Services Gateway.

   You have now verified the WebSphere deployment and configuration.

Preparing to use the inbound service

The next step is to start the inbound collaboration (if it is not already started) and prepare to receive messages using a Test Connector.

From the CrossWorlds System Manager window:

1. Expand the **BCT_TestAllTypes** project.
2. Expand the **Collaboration Objects** folder.
3. If it is not already started, right-click **SAI_to_BCTSampleConnector2_BCT_TestAllTypesInbound** and click **Start**. Notice that the ports are bound to the External Connector and BCTSampleConnector2.
4. If it is not already started, right-click **BCTSampleConnector2** and click **Start**.

Next, you'll display the Test Connector and configure it so that it can access the BCTSampleConnector2:

1. From Windows, click **Start > CrossWorlds > Connectors > Test Connector.**
2. Click **File > New Profile.**
3. From the Profiles window, click **Add**.
4. Verify that the **Server Name**, **Password**, and **Configuration file** fields are correct. Then type the following in the **Connector** field:

   `BCTSampleConnector2`
5. Click **OK**.
6. Click **File > Open Profile.**

7. Click the **BCTSampleConnector2** to select it, and click **OK**.
8. Connect to the InterChange Server by clicking **File > Connect Agent.**

   The agent should connect and list subscribed business objects in the left-hand pane of the window.

At this point, the Test Connector is attached to BCTSampleConnector2, which is bound to the To port of the inbound collaboration. The From port is bound to the External Connector, which is where the Java proxy calls when a Web service request arrives in WebSphere.

Understanding the outbound collaboration

This section describes the outbound collaboration.

Inspecting the outbound collaboration

1. Open the **Collaboration Objects** folder.
2. Double-click the following object so you can look at its ports and how they are bound to the connectors:

   `BCTSampleConnector1_to_BCTSampleSOAPConnector_BCT_TestAllTypesOutbound`

As you can see, there are three ports:

From port

The From port is where you trigger the collaboration with a BCT_TestAllTypes.Retrieve event. The From Port is bound to BCTSampleConnector1.

You will use BCTSampleConnector1 to create and initialize a BCT_TestAllTypes.Retrieve event to be sent from the From port.

To port

The To port is where the BCT_TestAllTypes.Retrieve is sent to the Web service. This port is bound to the BCTSampleSOAPConnector.

To display the maps for BCTSampleSOAPConnector:

1. Right-click the **To** port.
2. Click **Show maps**.

Here you see that there are outbound and inbound maps for the BCTSampleSOAPConnector. This connector requires a special top-level business object that encapsulates child business objects for the request, the response, and the fault (in case an error occurs). This top-level business object also contains an attribute for the URL for the Web-service application server. This business object is the application-specific business object for the SOAP connector. It is named BCT_SOAP_BCT_TestAllTypes.

The outbound map (BCT_TestAllTypes_to_BCT_SOAP_BCT_TestAllTypes) performs the following processing:

1. It converts the generic business object (BCT_TestAllTypes) into a BCT_SOAP_BCT_TestAllTypes application-specific business object (the top-level business object mentioned above). This application-specific business object has a child object into which the generic business object is mapped.
2. The map fills in the URL attribute for the Web-service application server:

a. A method in a class in bctwssamples.jar is called from the map to look up the URL. To see what it does, look at the file bct_ws_configuration.txt in the following directory: \bctws\cw
b. The mapping function takes the name of the source business object (in this case, BCT_TestAllTypes) and adds BCT_WS_ to the front of the name and _*Verb* (for example, _Retrieve) and _URL to the end of the name.
c. It then looks in the file for a key value based on this modified name. In this case, it looks for BCT_WS_BCT_TestAllTypes_Retrieve_URL.
d. When the key is found, the URL is determined from the file. In this case, the URL is seen to be **http://localhost/bctwssamplesweb/servlet/messagerouter**.
e. The map then fills in this value in the URL field of the target object.

There is also a response map for the BCTSampleSOAPConnector. This inbound map copies from the application-specific business object child field for the response back into the generic business object used in the collaboration flow. The name of this map is BCT_SOAP_BCT_TestAllTypes_to_BCT_TestAllTypes.

Reply port

After the SOAP connector finishes a successful call to the Web service, the response is in a BCT_TestAllTypes object. For testing purposes, the Reply port is bound to the BCTSampleConnector1.

When the BCTSampleSOAPConnector receives a reply, the reply business object flows to the Reply port.

Running the end-to-end scenario

Now you will use the outbound collaboration to trigger the inbound collaboration, using the Web service you deployed in WebSphere.

To review, the two collaborations are:

- An inbound collaboration object (SAI_to_BCTSampleConnector2_BCT_TestAllTypesInbound). This collaboration is exposed as a Web service and deployed in WebSphere. You receive requests and reply to them using the BCTSampleConnector2 and the Test Connector. Be sure that the inbound collaboration object and Test Connector for BCTSampleConnector2 are running (look under the BCT_TestAllTypes Project in the Collaboration Objects folder to confirm that the collaboration object and its connector are running).
- An outbound collaboration object that calls the inbound collaboration's Web-service interface using the SOAP connector. This collaboration object is named BCTSampleConnector1_to_BCTSampleSOAPConnector_BCT_TestAllTypesOutbound. You have not yet tested this collaboration object.

Testing the outbound collaboration object

To test the outbound collaboration object, you must make sure it has started and its connectors have been started. Do the following:

1. From the CrossWorlds System Manager menu bar, click **Server > System View.**
2. Check to see that the following connectors have been started:
   - BCTSampleConnector1
   - BCTSampleConnector2
   - BCTSampleSOAPConnector

You will start the agents for these connectors in the next section.

Starting the BCTSampleSOAPConnector agent

To start the agent:

1. Open a command prompt.
2. Navigate to the directory: %crossworlds%\connectors\SOAP.
3. Copy the start_BCTSampleSOAP.bat file into this directory. The file is located in: bctws\cw.
4. Type the following command, substituting your InterChange Server name (as it appears in the CrossWorlds System Manager left-hand pane) for <*servername*>:

   ```
   start_BCTSamplesoap BCTSampleSOAP <servername>
   ```

Starting the Test Connector

Next, you need a Test Connector that is associated with BCTSampleConnector1. If the Test Connector for BCTSampleConnector1 is not already running, bring it up now.

If this is the first time you are using the sample, you need to create a profile in the Test Connector for BCTSamplesConnector1 by following these steps:

1. Display the Test Connector window by clicking **Start > CrossWorlds > Connectors > Test Connector.**
2. Click **File > New Profile** to add a profile for the Test Connector to access BCTSampleConnector1.
3. In the Profiles window, click **Add**.
4. Verify that the **Server Name**, **Password**, and **Configuration file** fields are correct. Then type the following in the **Connector** field:

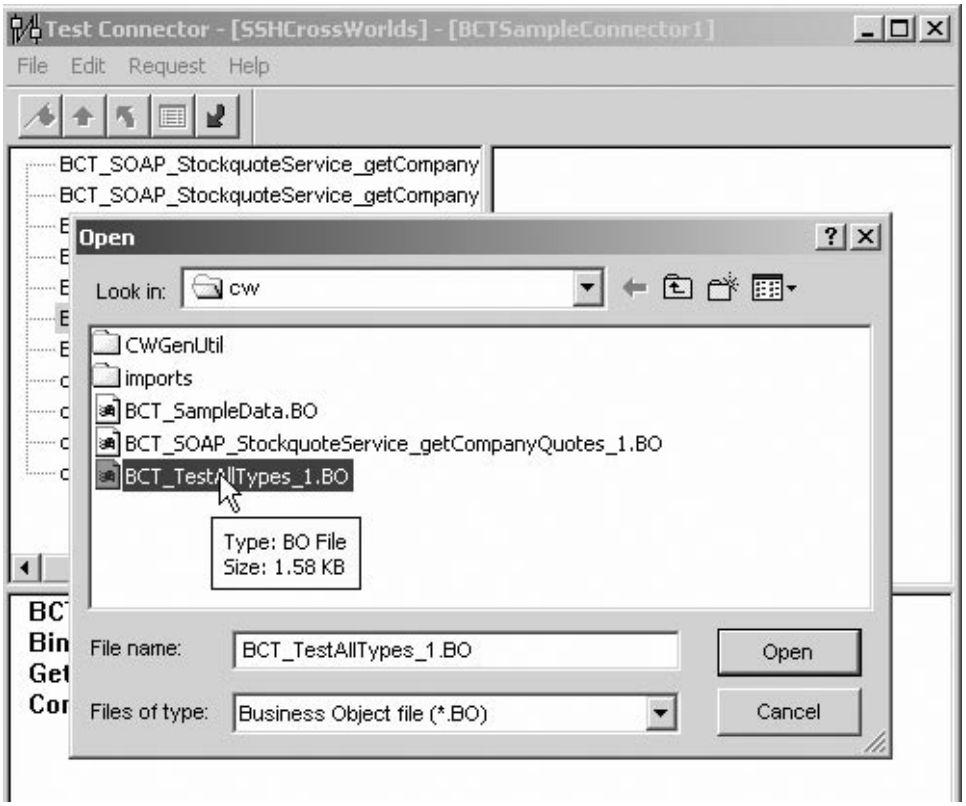
   ```
   BCTSampleConnector1
   ```
5. Click **OK**.
6. Click **File > Open Profile.**
7. Click **BCTSampleConnector1.**
8. Click **OK.**
9. Click **File > Connect Agent** to connect to the InterChange Server.

The BCT_TestAllTypes business object is now listed in the left pane of the BCTSampleConnector1 Test Connector. The business object BCT_TestAllTypes is saved in the directory: \bctws\cw

Loading the business object

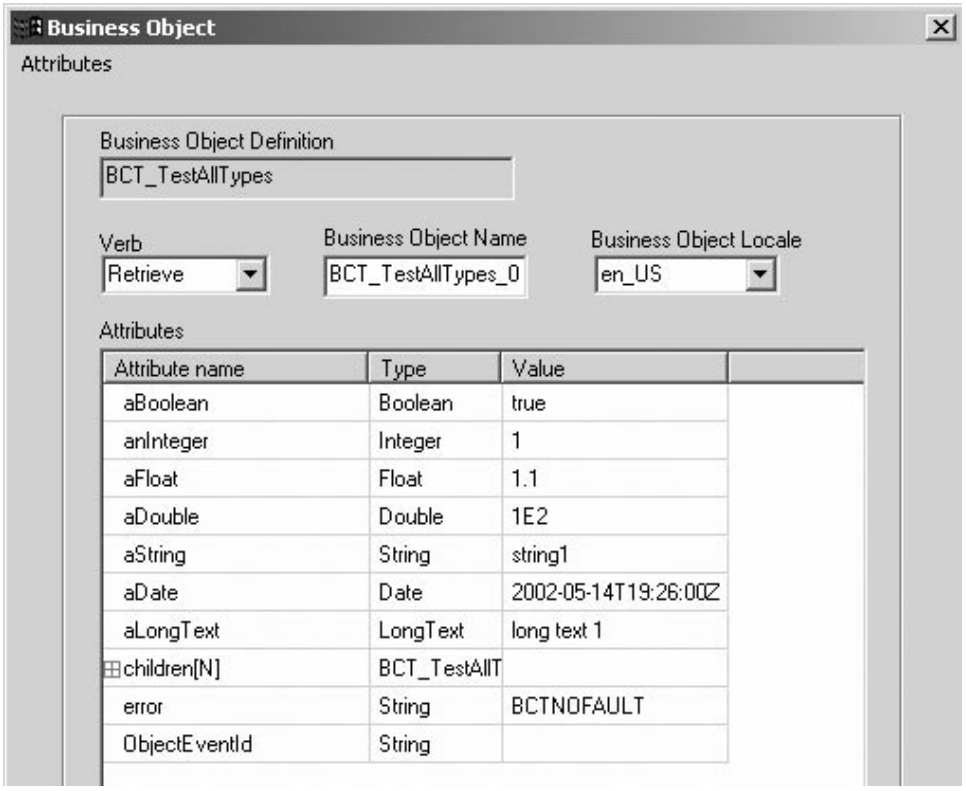
To load the business object:

1. Click **BCT_TestAllTypes** in the left pane to select it.
2. Click **Edit > Load BO** from the Test Connector menu bar.
3. Navigate to the directory: \bctws\cw
4. Click **BCT_TestAllTypes_1.BO**, as shown in the following screen, to load.



Test Connector screen and Open window with the business object highlighted

5. After you load the business object, double-click the resulting object instance and inspect its contents. This business object, shown in the following screen, will be sent to the Web service.



Sending the business object

To send the business object:

1. Click the instance of the business object to select it.
2. Click **Request > Send** from the Test Connector menu bar. This action sends the business object via the BCTSampleSOAPConnector and the Web service to the receiving collaboration.

Receiving the business object

Remember that the BCTSampleConnector2 is monitoring the To port of the receiving collaboration. To receive the business object:

1. Click the Test Connector for BCTSampleConnector2 to select it.
2. Click **Request > Accept Request.**

You see the business object that you sent in the right pane of the BCTSampleConnector2 Test Connector window.

Changing fields

Next, you will change some fields in the business object. In this way, you will be able to inspect the object when it is returned to make sure the service worked.

1. Double-click the **BCT_TestAllTypes** object.
2. In the editor window, change the values in the **Value** and **Name** fields.
3. Click **OK**.

Sending the business object back to the requester

1. Click the **BCT_TestAllTypes** object to select it.
2. Click **Reply > Success** from the BCTSampleConnector2 Test Connector window.

The business object is now sent as a reply to the requestor.

Checking the reply

BCTSampleConnector1 is bound to the Reply port of BCT_SampleConnector1_to_SOAPConnector_BCT_TestAllTypesOutbound, so you can see the reply:

1. Click **BCTSampleConnector1** to select it.
2. Click **Request > Accept Request** on the Test Connector window for BCTSampleConnector1.

To finish the end-to-end flow, from the BCTSampleConnector1 Test Connector window:

1. Click **BCT_TestAllTypes** in the right-hand pane to select it.
2. Click **Request > Reply > Success.**

The values in the Value and Name fields have changed, indicating success.

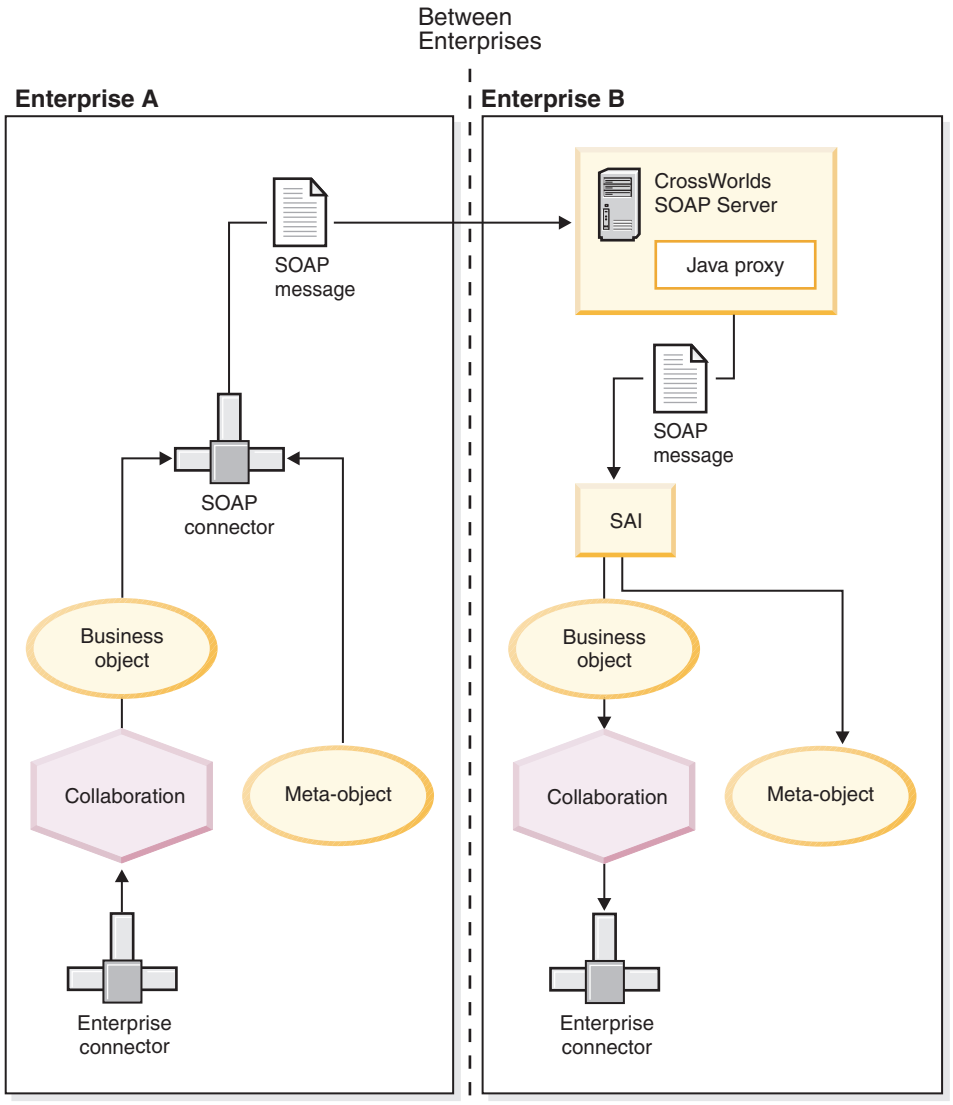
This sample did not make use of the Web Services Gateway on either the sending or receiving end. In "Adding the Web Services Gateway" on page 27, you will see how to expand the sample so that requests flow into and out of the Web Services Gateway.

Adding the Web Services Gateway

In the Basic samples section, you installed two CrossWorlds collaborations:

- An outbound sample to send a message using a SOAP connector to a SOAP-enabled application server. The server uses the SOAP message to call a method on a CrossWorlds Java proxy class.
- An inbound sample collaboration that is called by the proxy class. The proxy class uses the CrossWorlds Server Access Interface (SAI) classes to connect to the CrossWorlds instance where the inbound collaboration is deployed. The body of the SOAP message is passed via the SAI to the CrossWorlds instance. This message is converted into a business object and used to trigger the inbound collaboration's From port. The collaboration runs and returns a value to the caller synchronously via the SAI. The response flows back to the caller (the outbound collaboration) as a SOAP message. The SOAP connector calls the data handler for SOAP messages to convert the response to a business object. This business object is passed to the calling collaboration flow.

The basic connectivity sample that you have built so far (summarized above) is illustrated in the following diagram. Note that if you have deployed everything on a single machine, the diagram's Enterprise A and Enterprise B are logical rather than physical entities.



Flow of business object through the SOAP connector of one enterprise to another enterprise

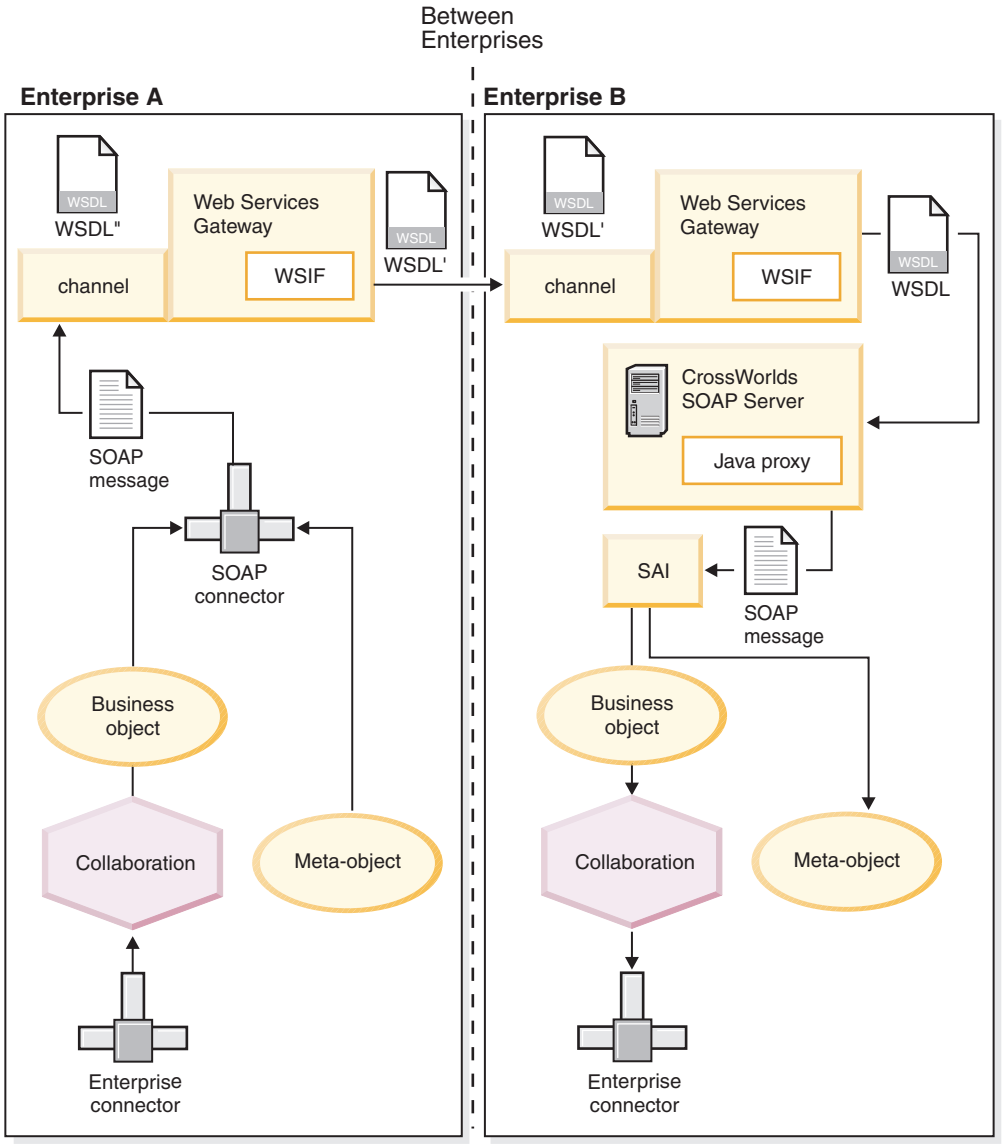
Connectivity using the Web Services Gateway

With this basic pattern for synchronous calls established, you will now learn how to add to the sample by using the Web Services Gateway. The Web Services Gateway provides the capability to:

- Log outgoing and incoming messages
- Secure access to services
- Route outgoing calls based on trading-partner information passed from the outgoing collaboration in the SOAP message header

The diagram below shows you how the Web Services Gateway is added to the basic connectivity scenario. In the basic connectivity sample, Enterprise A uses the outgoing collaboration to send a SOAP message directly to Enterprise B. To show how the Web Services Gateway is used by Business Connection, two gateway instances are introduced to the basic connectivity sample.

- Enterprise A uses a Gateway instance with a Private channel. All calls from Enterprise A are passed through the Private channel of Enterprise A.
- All messages that are to be processed by Enterprise B must be received by Enterprise B's Web Services Gateway on a Public channel.



Two enterprises, each with a Web Services Gateway

Note: For the basic connectivity sample, only one physical machine was required. If you are going to implement the Web Services Gateway portion of the sample scenario, you must use two physical machines. This is because two Web Services Gateway instances are needed, and only one Web Services Gateway instance can run on a physical machine. However, you can modify the sample to use just one Gateway if desired. That requires just one machine, but you will not be able to see how the WSDL provided by one Gateway is deployed into another Gateway if only one Gateway is used.

Web Services Gateway basic concepts

As the diagram above indicates, messages received on Web Services Gateway channels are forwarded by the Web Services Gateway using another component named Web Service Invocation Framework (WSIF). The diagram also indicates that WSDL is used in some manner by the Web Services Gateway and WSIF to enable the forwarding of messages. The basic concepts needed to understand how Business Connection uses the Web Services Gateway are explained in this part of the document. The explanations describe the concepts and steps needed to configure a Gateway to implement the sample scenario shown in the diagram.

For more in-depth discussion of how the Web Services Gateway works, see the Using the Web Services Gateway document.

Understanding Web Services Gateway channels

A Web Services Gateway channel is capable of receiving a message and passing it to the Web Services Gateway core component for processing. A channel has a URL that is used by callers to send it messages. The Web Services Gateway presently comes with several pre-configured channels.

For basic CrossWorlds-to-CrossWorlds interactions, Business Connection uses the pre-configured SOAP channels that are shipped with Web Services Gateway. There are two such pre-configured SOAP channels provided. The only difference between them is the pre-configured URL that they use to receive SOAP messages.

These SOAP channels use Apache SOAP 2.2 technology to receive messages and forward them to the Web Services Gateway core code. The reason for two pre-configured SOAP channels is to provide Public and Private SOAP channels for a Web Services Gateway instance.

By convention, the ApacheSOAPChannel1 will be called the Private channel and ApacheSOAPChannel2 will be called the Public channel.

To see if a channel is active, use a browser and enter its endpoint URL (for example, http://*yourhostname*/wsgwsoap1). Doing so issues an HTTP/GET request, which the channel handles by displaying the following message:

```
Welcome to the Apache SOAP channel for the IBM Web Services Gateway.

Currently the only one way to interact with this channel is to send web
services invocations to it.
```

Understanding WSIF and WSDL usage by the Gateway

In this section, you will learn about how the Business Connection uses Web Services Gateway services.

After a SOAP message is received on a channel, the channel passes it to the Web Services Gateway core component for processing. In the most basic use of the Web Services Gateway, the Gateway processes a message by using Web Services Invocation Framework (WSIF) to send the message to a different URL. (As previously noted, the Web Services Gateway can also provide logging, security, and routing services, which are discussed in Web Services Advanced Topics.)

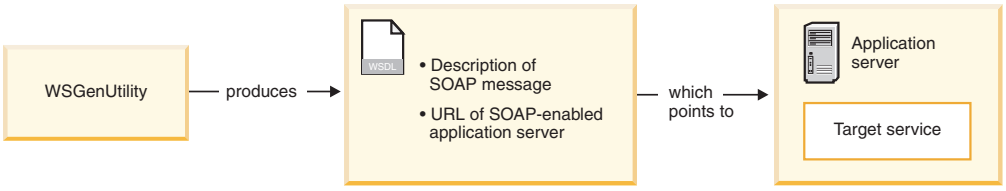
The question is—how does the Web Services Gateway core determine how and where to forward the message?

The answer lies in the concept of *gateway services* and *target services*.

Target service

A target service represents a forwarding destination for the Web Services Gateway. For Business Connection, a target service is described by a Web Services Description Language (WSDL) file, which contains a URL and an interface for the Web service.

As shown in the following illustration, the target WSDL is generated by the WSGenUtility, which is the CrossWorlds utility you use to create the web service. The WSDL includes a description of the SOAP message that can be processed by the target service along with the URL that can be called to obtain the service.

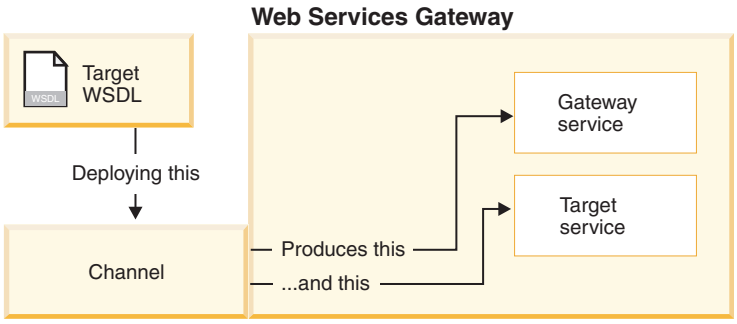


WSGenUtility produces a WSDL, which points to the target service

You might think of the target service in this case as the *actual* Web service, the service that will perform the request.

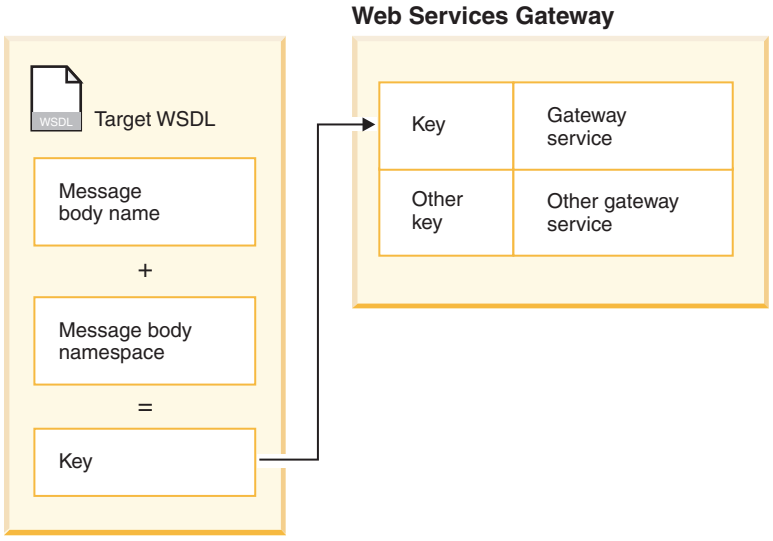
Gateway service

A gateway service is produced by the Web Services Gateway when a target WSDL file is deployed in the Gateway. (You use the Web Services Gateway administrative console to deploy the file, as will be described in a later section.) For Business Connection, a target WSDL is deployed in one of the SOAP channels depending on whether the call is considered to be publicly available or privately available. After deployment of the target WSDL, the Web Services Gateway produces a gateway service (as shown in the following illustration) using the URL of the SOAP channel where deployment was made. The gateway service is provided by the Gateway on the URL of the channel where you deployed the target WSDL.



WSDL deployed in a channel, producing a target service and gateway service

The relationship of the gateway service to the target service is maintained internally by the Web Services Gateway core component. The target WSDL message description includes the message body name (a string) and body namespace (another string). During deployment, the Web Services Gateway core hashes together (combines) the body name and the body namespace to form a key. As shown in the following illustration, this key is used to look up a gateway service when a message is received on the channel.



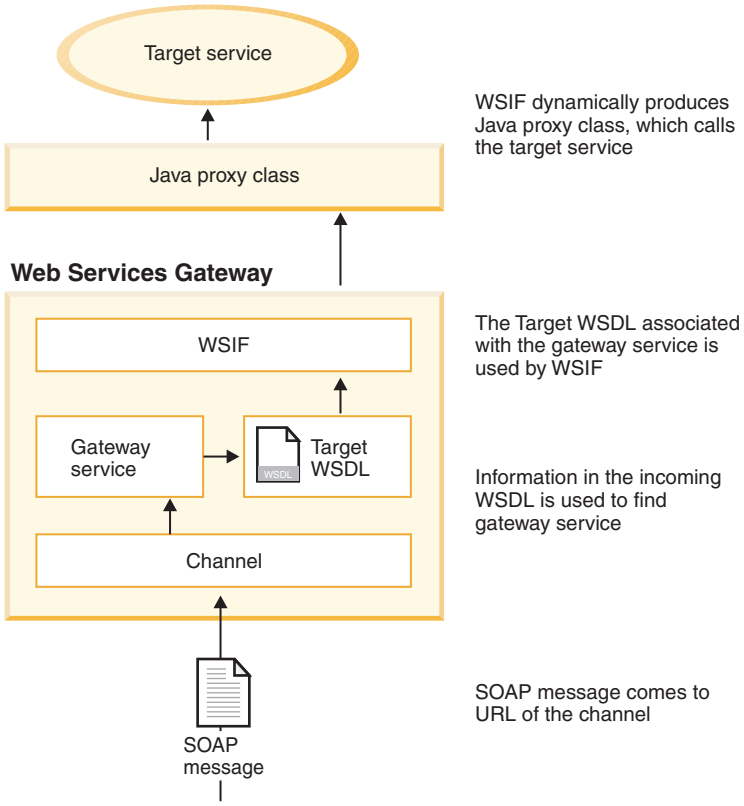
The body name plus the body namespace equals the key to the service

When the SOAP channel receives a SOAP message, it is passed to the Web Services Gateway core component. The SOAP message body name and body namespace are hashed together, and the gateway service is found using the resulting key.

Each gateway service is associated with one or more target services. Assume for now that only one target service is available. (The case of *n* target services is

discussed in the Routing Filter section.) The target service for the message that came on the channel is found in this manner. The Web Services Gateway stores a reference to the original target WSDL for each target service. To re-send the message to the target service, the target WSDL is used.

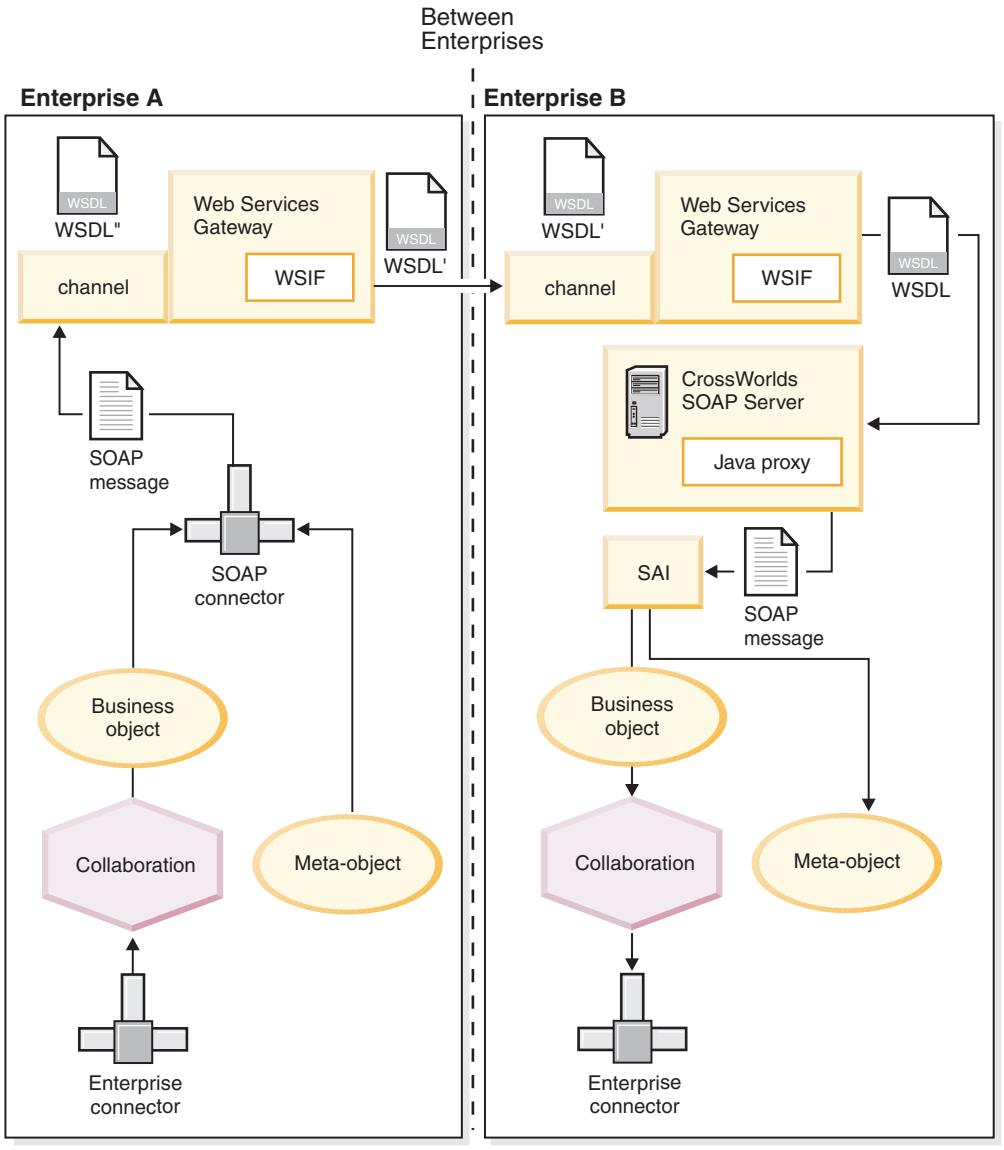
This is where WSIF come into use. WSIF (as shown in the following illustration) is able to use the target WSDL to dynamically produce a Java proxy class that can be used to call the service described by the WSDL. The gateway uses the target WSDL reference to produce such a proxy. The incoming message is then re-sent to the target service (on its original endpoint URL) using the proxy class.



The WSIF pointing to a Java proxy class, which calls the service

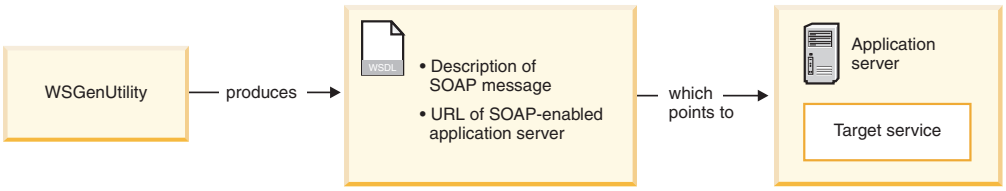
Inbound service deployment

Look again at the following drawing, which shows how the Web Services Gateway instances are used by Enterprise A and Enterprise B. The actual service is performed by the CrossWorlds on Enterprise B. To use the Gateways in the sample, you start with the actual service on Enterprise B.



Two enterprises, each with a Web Services Gateway

When the Web service for a collaboration based on BCT_TestAllTypesInbound is developed, the WSGenUtility is used by a developer to produce several files that are needed by Business Connection to enable Web services. (The Web Services Technical Reference document describes the use of WSGenUtility in depth.) As mentioned earlier, one of the files produced by the utility is a WSDL file describing the SOAP message for the Web service and the URL of the SOAP-enabled application server where the service is hosted.



How the WSDL file is used to find the service

In the CrossWorlds-to-Web Services Gateway drawing, this WSDL file is referred to as wsdl in Enterprise B. The drawing shows wsdl over the arrow going into the CrossWorlds Soap Server. This indicates that the server can process messages described by wsdl.

To provide the service on the Public channel of the Web Services Gateway-B, you deploy wsdl into the Web Services Gateway-B Public channel. This results in another WSDL file, which the drawing labels wsdl′ (wsdl prime). The wsdl′ file describes a message using the endpoint URL of the Public channel. Messages formed according to the description in wsdl′ and sent to the Public channel URL are forwarded to the target service described by wsdl. WSIF uses the reference that it holds to wsdl to produce a dynamic proxy, which is used to forward the message.

Now you will use the sample files to deploy the service on the Web Services Gateway at Enterprise B.

The WSDL that was produced for the BCT_TestAllTypesInbound service is named BCT_TestAllTypes_Retrieve.wsdl. It is saved in directory: \bctws\was\cw\wsdl

To deploy this WSDL file in Web Services Gateway-B:

1. Display the Web Services Gateway administrative console by entering:
   `http://enterpriseb.hostname/wsgw/admin`

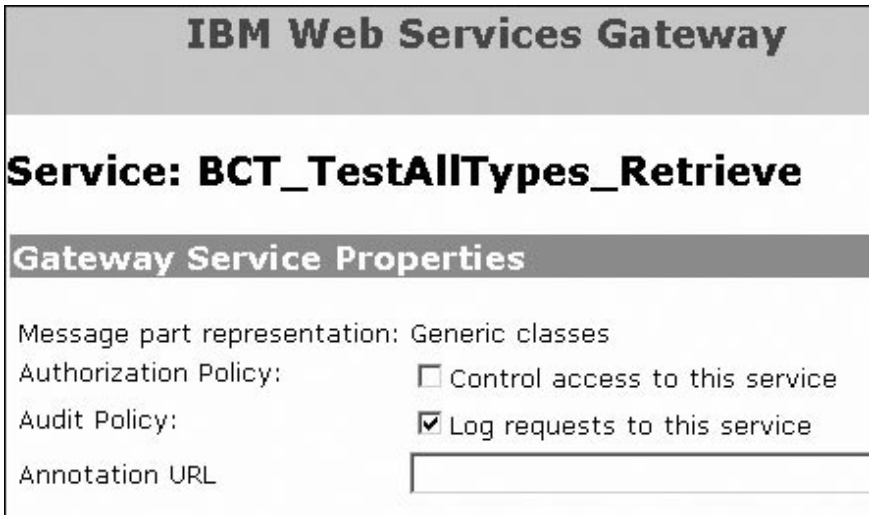
   at a browser (use the real host name for Enterprise B).
2. Click **Deploy** under the **Services** category on the left side of the screen.
3. From the Deploy Services screen, type the following in the **Gateway Service Name** field:
   `BCT_TestAllTypes_Retrieve`
4. Accept **Generic Classes** (the default) in the **Message part representation** field.
5. Click **ApacheSOAPChannel2** to select it from the **Channels** list.
6. In the **Target Service Location**, type:
   `\bctws\was\cw\wsdl\BCT_TestAllTypes_Retrieve.wsdl`

   Use your installed drive letter if it is different than C.
7. Accept **URL** (the default) in the **Location** field.
8. Leave the other fields blank.
9. Click **OK**.

The Web Services Gateway will deploy the WSDL file and provide a gateway service available on the Public SOAP Channel (ApacheSOAPChannel2) that is described by wsdl'.

After successful deployment, you see BCT_TestAllTypes_Retrieve listed under Gateway Services.

You can click the Gateway Service link on the list to see information held in the Web Services Gateway for this service, as shown in the following screens. Note that these screens are used for illustration and do not necessarily match the samples you just completed.



Web Services Gateway screen showing Gateway Service properties

This part of the screen shows the name of the service, how the message is represented internally by the Web Services Gateway during processing, the Authorization Policy and Audit Policy for the service (which are discussed in the Web Services Technical Reference), and the Annotation URL (which is not currently used by the Web Services Gateway). You can change the configuration of the gateway service by changing the information and clicking **Apply Changes**.

The next section of the screen shows information about the target services that are associated with the gateway service.

Web Services Gateway screen showing multiple targets

In the sample, there is only one target service at this point. You can see the URL of the file that was used to create the gateway service. This reference is used by WSIF to re-send messages that come in to the gateway service. You can remove a target service by clicking the **remove** button below it.

By entering a new WSDL location, you can define another target service for the gateway service. In this way, more than one target can be associated with a single gateway service. (In the screen shown above, three target services have been deployed for the gateway service.) If you define more than one target, each must have a unique Target Service Identity Information value assigned. When routing is discussed, you will see how one target service can be selected from many possibilities by the use of a routing filter that uses information held in the SOAP message header.

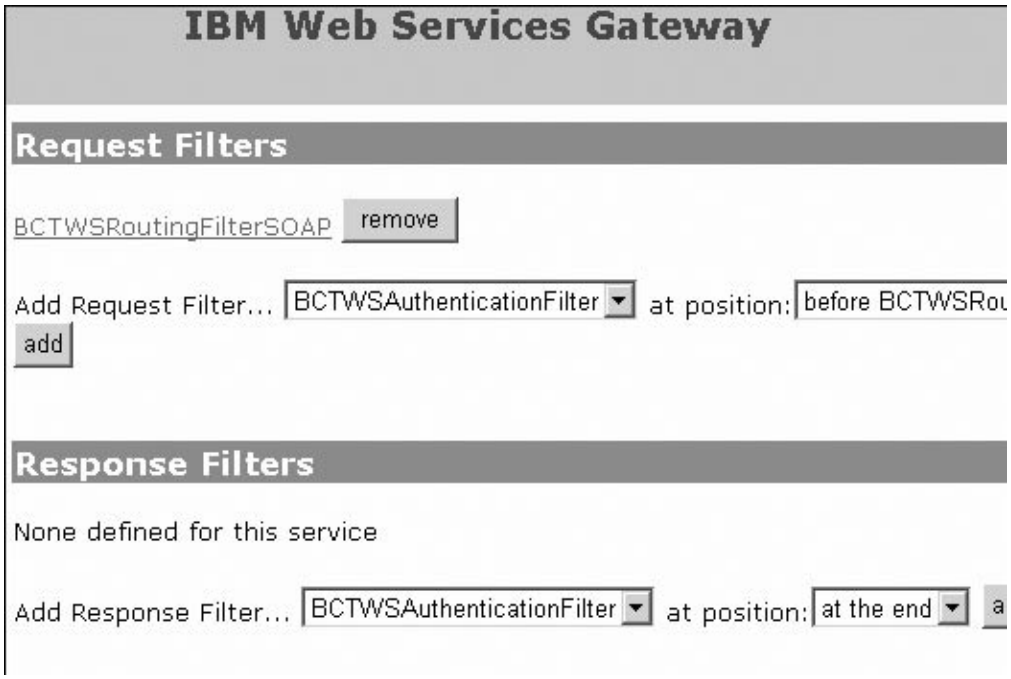
Also note that the **Target Service Name** and **Target Service Namespace** fields are not required. In this sample, there is only a single service described in the target WSDL, so the Web Services Gateway will use this one and only service when deploying. If there are several services in the target WSDL, you will need to enter these fields to select one of them when deploying.

Next is the Channels section of the screen. Again, this screen is being used for purpose of illustration and does not reflect the sample with which you have been working.

Web Services Gateway screen showing how channels can be added or removed

Here you see that the gateway service is available on ApacheAxisChannel1 and ApacheSOAPChannel1 but not on ApacheAxisChannel2. For the case shown in the figure, you could also add ApacheAxisChannel2 also, meaning that a caller could use three channels to call the service (but, for this sample, don't do it). You also could remove the gateway service from ApacheAxisChannel1 and ApacheSOAPChannel1 by clicking the **remove** button (again, do not do it at this time.)

Next is the Filters section of the screen.



Web Services Gateway screen showing how filters can be added or removed

Note that the figure above shows a deployed filter for a gateway service. The sample you have worked with in this chapter does not use a filter. A request filter is a module that is called just before the target service is called. A response filter is

a module called just after a target service has completed and before the response is sent back to the channel. Later you will see how a request filter is used to provide routing to select one of many possible target services. Refer to the Using the Web Services Gateway document for more discussion of filters.

At the bottom of the Gateway Service screen are two sections. The UDDI References capability of the Web Services Gateway is not used by the WebSphere Business Connection at this time and is not discussed here.

At the very bottom of the screen are links that can be used to view the gateway service wsdl'.

Web Services Gateway screen showing contents of WSDL

Click on the **External WSDL implementation definition (WSDL only)** link. Here you see that the gateway service is available on endpoint **URL http://stevesh/wsgwsoap2/soaprpcrouter**. Higher up in the file is an import element. The import shows where the interface can be obtained, which is where the structure and content of a SOAP message that can use the endpoint are described.

Clicking on the Interface link at the bottom of the Gateway Service screen gives the following (only the top of the screen is shown here):

Web Services Gateway screen showing contents of interface

This is the interface that is imported into the Service definition you looked at previously. You can see that there is information here that describes the structure and content of SOAP request and response messages that are valid for the service.

Note that wsdl' from the Web Services Gateway is comprised of two parts—the Service description and the Interface description. This is because it is considered to be good practice to separate the Service and Interface descriptions, with the Service importing the Interface. It promotes reuse of an interface by different service endpoints.

Obtaining wsdl'

As discussed above, clicking the hypertext links at the bottom of the Gateway Service screen gives a browser view of the two files (service and interface) that comprise a Gateway WSDL. The links are actually calls to URLs in the Web Services Gateway that provide the descriptions.

- The Service Definition is provided by calling:
  http://enterpriseb.hostname/wsgw/ServiceDefinition?name=BCT_TestAllTypes_Retrieve

- The Service Interface is provided by calling:
  http://enterpriseb.hostname/wsgw/ServiceInterface?name=
  BCT_TestAllTypes_Retrieve

(**Note**: Substitute the actual hostname in the URL.)

The way that the Interface is imported by the Definition is by using the URL. This means that the URLs as shown above provide one way to obtain wsdl'.

Another way involves producing actual files for the Service Definition and Interface. To do this, you do the following:

1. In Internet Explorer 5.5, right-click the **Service Definition** link and select **Save target as**.
2. Choose a directory and file name for the ServiceDefinition xml file. A suggested naming scheme is *ServiceName*_definition.wsdl (for example, BCT_TestAllTypes_Retrieve_definition.wsdl.)
3. Repeat for the Service Interface link, using a name like *ServiceName*_interface.wsdl.
4. Edit the location attribute of the import element of the definition file to import the interface from a file URL. For example, change from:

   ```
   <import namespace="urn:ibmwsgw#BCT_TestAllTypes_Retrieve"
      location="http://stevesh.charlotte.ibm.com/wsgw
      /ServiceInterface?name=BCT_TestAllTypes_Retrieve"/>
   ```

   to

   ```
   <import namespace="urn:ibmwsgw#BCT_TestAllTypes_Retrieve"
      location="BCT_TestAllTypes_Retrieve_interface.wsdl"/>
   ```

After this change is made, wsdl' is available as a file in the file system. This will be a useful technique for deployment when the servlet URLs are not available because of firewalls or other reasons.

Private service deployment

At this point, you have deployed the service in the Public channel of Enterprise B. This service can be called with messages that are formed from wsdl'**.** The next step is to provide a way for the SOAP connector of the collaboration inside Enterprise A to call a Private channel deployed in Web Services Gateway-A.

To do this, you need to deploy wsdl' from the Public channel of Web Service Gateway B into the Private channel of Web Services Gateway A.

In the drawing, the wsdl' file from Enterprise B is shown to the right of Web Services Gateway. This indicates that Web Services Gateway-B can call a target service described by wsdl'.

To provide the service on the Private channel of Web Services Gateway A, you deploy wsdl' into the Web Services Gateway A Private channel. This results in another WSDL file, which the drawing labels wsdl'' (wsdl double prime). The wsdl'' file describes a message using the endpoint URL of the Private channel. Messages formed according to the description in wsdl'' and sent to the Private channel URL are forwarded to the target service described by wsdl'. WSIF uses the reference that it holds to wsdl to produce a dynamic proxy, which is used to forward the message.

1. To deploy the service, at the Enterprise A machine, go to the Web Services Gateway Deploy Services administration screen and do the following (substituting the actual host name in the URL):
   a. In the **Gateway Service Name** field, type:
      ```
      BCT_TestAllTypes_Retrieve
      ```
   b. In the **Message part representation** field, accept **Generic Classes** (the default).
   c. In the **Channels** field, select **ApacheSOAPChannel1** by clicking it.
   d. In the **Target Service Location** field, type:
      ```
      http://enterpriseb.hostname/wsgw/ServiceDefinition?name=
      BCT_TestAllTypes_Retrieve
      ```
   e. In the **Location** field, accept **URL** (the default). As discussed previously, this uses the ServiceDefinition servlet on Web Services Gateway B to obtain the target wsdl.
2. Click **OK** to add the new gateway service to Web Services Gateway A.

The last step is to change the URL look-up file used by the outbound mapping from the generic business object to the SOAP application-specific business object. Here the URL of the Private channel must be used as the destination of the message that is sent from the SOAP Connector.

The URL lookup file is: \bctws\cw\bct_ws_configuration.txt.

1. Open this file for edit.
2. Change the entry for the BCT_TestAllTypes_Retrieve as follows (substituting the actual host name in the URL):
   ```
   BCT_WS_BCT_TestAllTypes_Retrieve_URL=http://enterprisea.hostname/
   wsgwsoap1/soaprpcrouter
   ```

Testing the end-to-end flow

At this point, you have completed the basic Gateway-to-Gateway deployment. You can now start both Gateways, start both CrossWorlds servers, and start the CrossWorlds application server on Enterprise B that hosts the Web service.

Use the same technique described in the basic CrossWorlds-to-CrossWorlds scenario to send messages from Enterprise A to Enterprise B through the Gateways.

Notices

IBM may not offer the products, services, or features discussed in this document in all countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

WebSphere Business Connection Lab Director
IBM RTP Laboratory
3039 Cornwallis Road
P.O. BOX 12195

Raleigh, NC 27709-2195
U.S.A

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples may include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Programming interface information

Programming interface information is intended to help you create application software using this program.

General-use programming interfaces allow you to write application software that obtain the services of this program's tools.

However, this information may also contain diagnosis, modification, and tuning information. Diagnosis, modification and tuning information is provided to help you debug your application software.

Warning: Do not use this diagnosis, modification, and tuning information as a programming interface because it is subject to change.

Trademarks and service marks

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM
alphaWorks
AIX
CrossWorlds
DB2
DB2 OLAP Server
DB2 Universal Database
DeveloperWorks
MQSeries
SecureWay
WebSphere

Lotus is a trademark of International Business Machines Corporation and Lotus Development Corporation in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.

Other company, product or service names may be trademarks or service marks of others.