



Build a Web service from an RPG program using iSeries Web services tools

Contents

Build a Web service from an RPG program using iSeries Web service tools 1

Introduction	1
Creating and Testing an iSeries Web service	3
Starting the product	3
Opening the Remote System Explorer perspective	6
Creating a connection to the iSeries server	8
Invoking the Web Service wizard	9
Setting Web service options	13
Setting runtime information and selecting a service program object.	14
Testing the Web service	20
Module summary	23
Deploying a Web service (optional)	23
Modifying the service project WSDL	24
Building the EAR file	24

Installing the enterprise application	27
Manage the installed application	30
Modifying the client project WSDL	31
Testing the Web service deployed and running on iSeries	32
Module summary	33
Calling a Web service from RPG	34
Setting up the QShell runtime environment.	35
Copying the WSDL file to IFS	37
Extracting the Web service interface information and creating the program interface	38
Editing and creating the RPG program	41
Running the program	46
Module summary	47
Summary	48
Notices	48

Build a Web service from an RPG program using iSeries Web service tools

This tutorial teaches you how to create a simple Web service from an RPG batch program and test it locally. The Web service will communicate with the business logic written in ILE RPG residing on an iSeries™ server. While creating a Web service, you will learn how to use the Web Services wizard to create an iSeries Program Web Service, generate a proxy and test the Web service in a local test client.

Learning objectives

- Create an iSeries Web service using the Web Service wizard
- Test an iSeries Web service on a local test client
- Deploy the iSeries Web service to WebSphere® Application Server for iSeries
- Test the deployed iSeries Web service
- Call a Web service from an RPG program

60 minutes



Introduction

In this tutorial, you learn how to wrap a customer inquiry application written in RPG as a Web service using iSeries Web service tools. You use the Web Service wizard to build the Web service, setting all the properties and letting the wizard generate the code for you. You will see the Web Service Description Language (WSDL) definition that the wizard will create for the deployment phase and also learn about the proxy that helps client applications invoke the service. Finally you learn how to package your application and Web service in an enterprise archive (EAR) so that you can deploy it on a machine running WebSphere Application Server.

Next you'll deploy the EAR on the WebSphere Application Server for iSeries. You'll then find the Web service and learn how to invoke the service.

After you learn how to invoke an RPG program or service program as a Web service you learn how to call a Web service from RPG. You will use the AXIS client on i5/OS® to invoke the Web service by calling AXIS C functions from RPG.

Learning objectives

- Create an iSeries Web service using the Web Service wizard
- Test an iSeries Web service on a local test client
- Deploy the iSeries Web service to WebSphere Application Server for iSeries
- Test the deployed iSeries Web service
- Call a Web service from an RPG program

This tutorial should take approximately 60 minutes to finish. If you explore other concepts related to this tutorial, it could take longer to complete.

Skill level

Introductory

Audience

iSeries programmer

System requirements

- IBM® WebSphere Development Studio Client for iSeries, Version 7 and all software updates provided through the IBM Installation Manager
- i5/OS V5R3 and V5R4

Prerequisites

- Basic knowledge of Microsoft® Windows® operations such as working with the desktop and basic mouse operations such as opening folders and performing drag-and-drop operations.
- Basic knowledge of how Web applications work.
- Basic knowledge of how to use a browser to navigate the Internet.
- Restore the WSSLABXX savefile (WSSLABXX.savf) on an iSeries system:
 - Import the WSSLABXX project to your workspace.
 - Open the Project Explorer view by clicking **Window** → **Show View** → **Other** → **General** → **Project Explorer**.
 - Expand the **wsslabxx** project.
 - Right-click the **wsslabxx.savf** savefile, and select **Restore on iSeries**.
- Extract the file wsslabxx.zip to your C drive
 - Open the Project Explorer view by clicking **Window** → **Show View** → **Other** → **General** → **Project Explorer**.
 - Expand the **wsslabxx** project.
 - Right-click the **WSSDEMO.zip** file, and select **Export**.
 - Export the file to the local file system.
 - On the local file system unzip the file WSSDEMO.zip.
 - Install the enterprise application IntelWSEAR.ear on WebSphere Application Server for iSeries instance.
 - Copy the GetProductInfo.wsdl file to your C: drive.
- Install the enterprise application IntelWSEAR.ear on WebSphere Application Server for iSeries instance
- Verify all OS/400® PTFs installed for Development Studio Client (Right-click **iSeries Objects** and click **Verify Connection** on the pop-up menu).

Experience with Web services or technologies such as Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL), or Universal Description, Discovery, and Integration (UDDI) standard, is not necessary but helpful.

Expected results

Upon completion of this tutorial, you will know how to take an iSeries RPG application and wrap it as a Web service and also how to call an existing Web service from an RPG program.

Conventions used in this tutorial

- **Bold** font for user interface controls
- Mono-spaced font for user input and code blocks

- *Italic font* for variable names and glossary terms

Creating and Testing an iSeries Web service

In this module, you learn how to create an iSeries Web service from an RPG program and test it locally. You will also learn about a Web service, a client proxy and the test client.

Learning objectives

After completing the lessons in this module you will understand the concepts and know how to do the following:

- Start the product
- Set the default workspace
- Check the Remote System Explorer perspective is open
- Set up a new connection to an iSeries server
- Locate an RPG source file you want to create a Web service from
- Start the Web Service wizard
- Select Web Service options (create a Web project, generate a proxy, test the Web service)
- Specify runtime information and select the service program object
- Specify the service deployment configuration
- Test the service in the test client

This module should take approximately 30 minutes to complete.

Starting the product

In this lesson, you start Development Studio Client from the Windows Start menu and set the workspace location to WSSLABxx to store your files and folders.

To start the product and set the workspace location:

1. Click **Start > Programs > IBM Software Development Platform > WebSphere Development Studio Client for iSeries > WebSphere Development Studio Client for iSeries**



A dialog for workspace selection will appear asking you for the workspace location, unless you used the product before and selected not to show this dialog again.



The workspace contains all the information about your Development Studio projects. You can accept the default or store the work related to this tutorial, in a separate workspace.

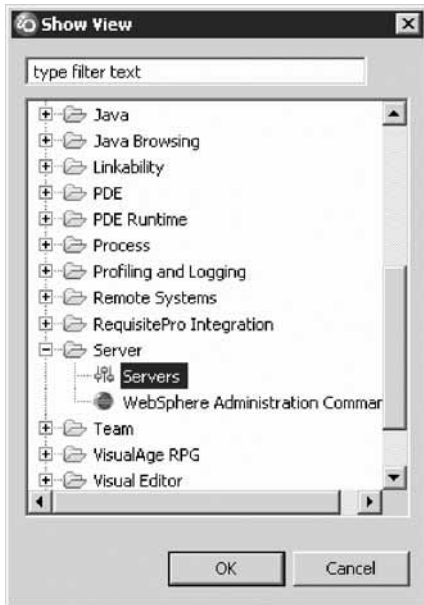
2. To name the directory of the workspace as shown above, use the directory name WSERVLABxx.
3. Click **OK**.

After a few moments of loading, the workbench opens, and the initial window of the product appears on your desktop.



Before creating a Web service, the WebSphere Application Server on which the Web service will run must be started. Although you can start the server in the Web service wizards, it may take several minutes to start depending on the speed of your machine. Starting the server before you begin will both increase the speed with which the wizard completes and reduce the chance that the wizard will generate an error because the server is taking too long to start.

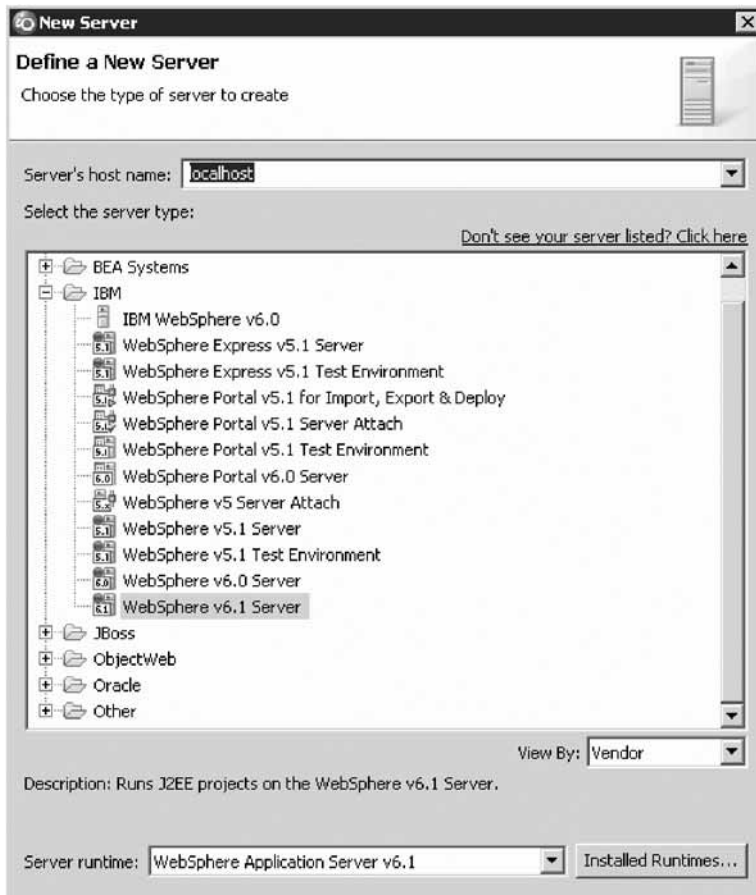
4. To start the application server before running the Web Service wizard:
 - a. From the Servers view (**Window > Show View > Other**), expand **Server** and double-click **Servers**.



- b. Next, right-click the **WebSphere Application Server v6.1** server in the list and select **Start**, if the server is not already started. You don't need to wait for the server to start to continue with the next lesson.



- c. To see if the server is started, check that the status of the server shows **Started**.
5. If you need to create a server instance, follow these steps:
 - a. Right-click in the Servers view and click **New > Server** on the pop-up menu. A Server Selection dialog opens.



- b. Click **WebSphere v6.1 Server**. This is the WebSphere Test Environment for Version 6.1.
- c. Click **Finish**.
- d. If you want to close the Servers view, right-click on the **Servers** tab and click **Close** on the pop-up menu.

You have started the product, set the default workspace, and started the server.

Lesson checkpoint

You learned the following:

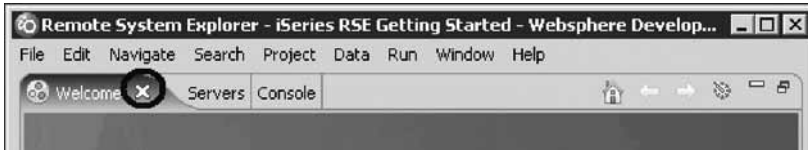
- How to start the product
- About workspace
- How to specify the name of the workspace
- How to start the WebSphere Application Server instance



Opening the Remote System Explorer perspective

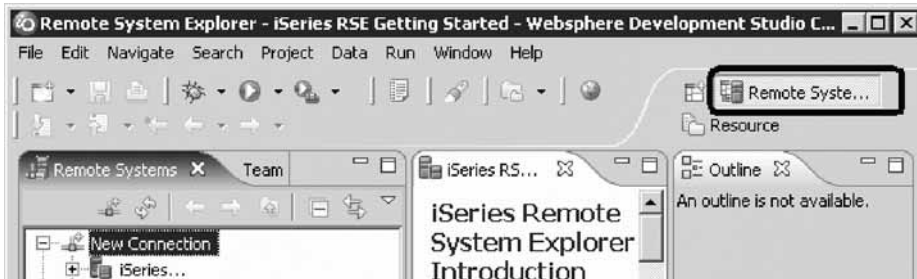
In this lesson, you learn how to check to see if the Remote System Explorer perspective is open in the workbench.

To open the Remote System Explorer perspective:

1. Check if the window title bar says: Remote System Explorer. If it does go directly to “Creating a connection to the iSeries server” on page 8. If not, continue with the steps below.



2. In the workbench, check whether you have the Remote System Explorer perspective already open. Look for the RSE icon  in the taskbar of the workbench.
3. If you cannot see the RSE icon because the Welcome view is open, click X to close the Welcome view.
4. If you see it click the  RSE icon.



5. Otherwise, click **Window > Open Perspective > Remote System Explorer** on the workbench menu.

The Remote System Explorer perspective opens.



Lesson checkpoint

You learned the following:

You learned the following:

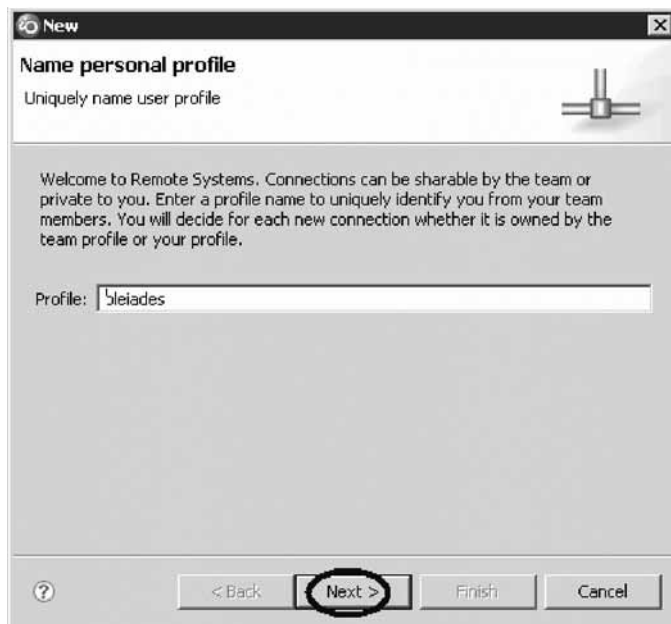
- How to check the Remote System Explorer perspective is open
- How to open the Remote System Explorer perspective

Creating a connection to the iSeries server

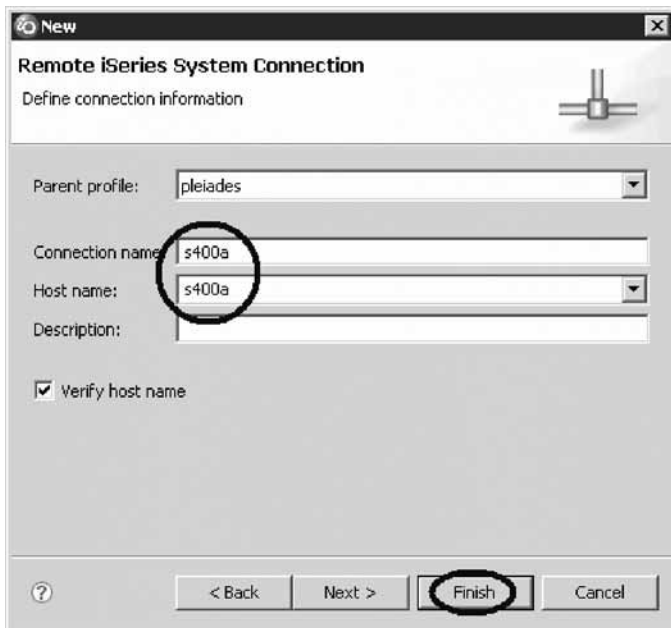
When you first start the workbench and open a perspective, you are not connected to any system except your local drive on your workstation. In this lesson, you learn how to configure and connect to an iSeries server.

To configure a connection:

1. Click the plus sign + beside **iSeries** under **New Connection** in the Remote Systems view.
2. Accept the default profile name.



3. Click **Next**.



In this lesson s400a is used as the name for the iSeries server. You will need to use the name or IP address of your particular iSeries server.

4. Type a host name, for example, s400a in the **Host name** field.
5. The **Connection name** field will be filled automatically with the same entry.
6. Click **Finish**.

Your new connection appears in the Remote Systems view and you are ready to work with objects on this iSeries system.

Lesson checkpoint

You learned the following:

- How to connect to an iSeries system

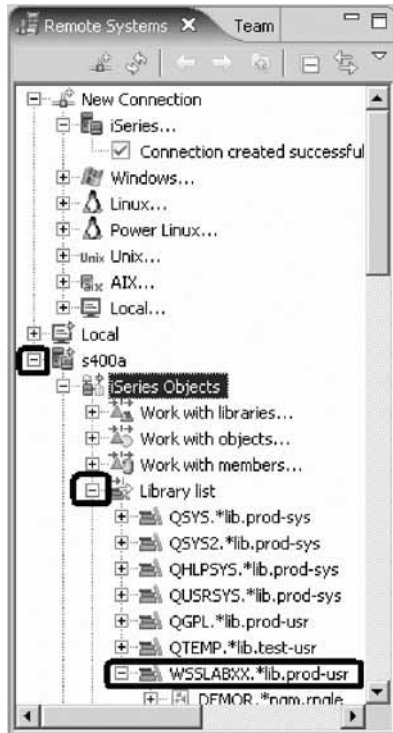
Invoking the Web Service wizard

You want to create a Web service wrapper for a procedure in a service program. During this lesson you will find the source member for this service program to extract the necessary API information.

Tip: Before creating a Web service, the WebSphere Application Server on which the Web service will run must be started. Although you can start the server in the Web service wizards, it may take several minutes to start depending on the speed of your machine. Starting the server before you begin will both increase the speed with which the wizard completes and reduce the chance that the wizard will generate an error because the server is taking too long to start. You should have already started the server during “Starting the product” on page 3.

To invoke the Web Service wizard:

1. Expand **iSeries Objects** by clicking the plus sign + beside it, to locate the source member.
2. Expand **Library List**.



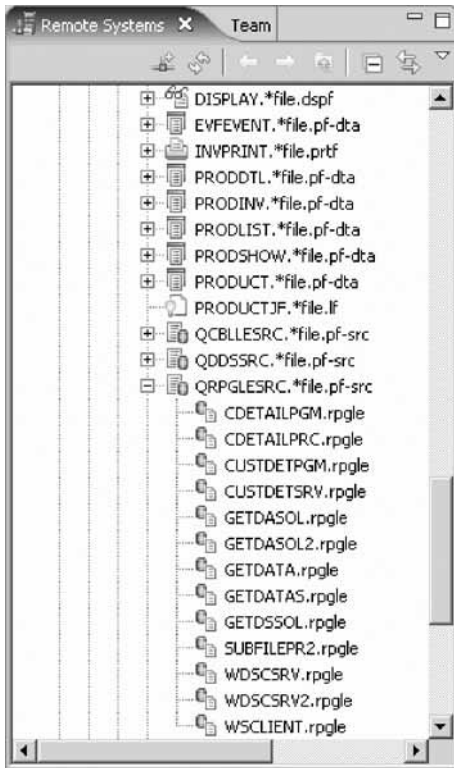
A User ID and password dialog opens.

3. Enter your User ID and password.



Note: If you don't see **WSSLABxx** in your library list, then you need to add the library. You can right-click the Library list and click **Add Library List Entry** from the pop-up menu to add **WSSLABxx** to your library list.

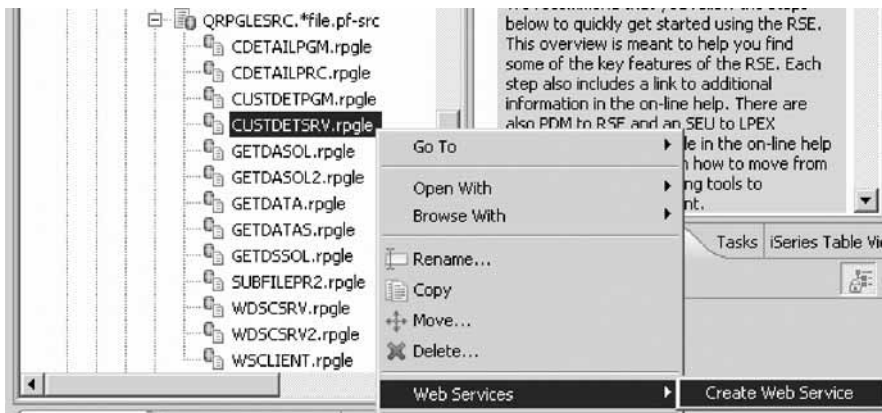
4. Expand the library **WSSLABxx**.
You will see all objects in this library.
5. Expand the source file **QRPGLESRC**.



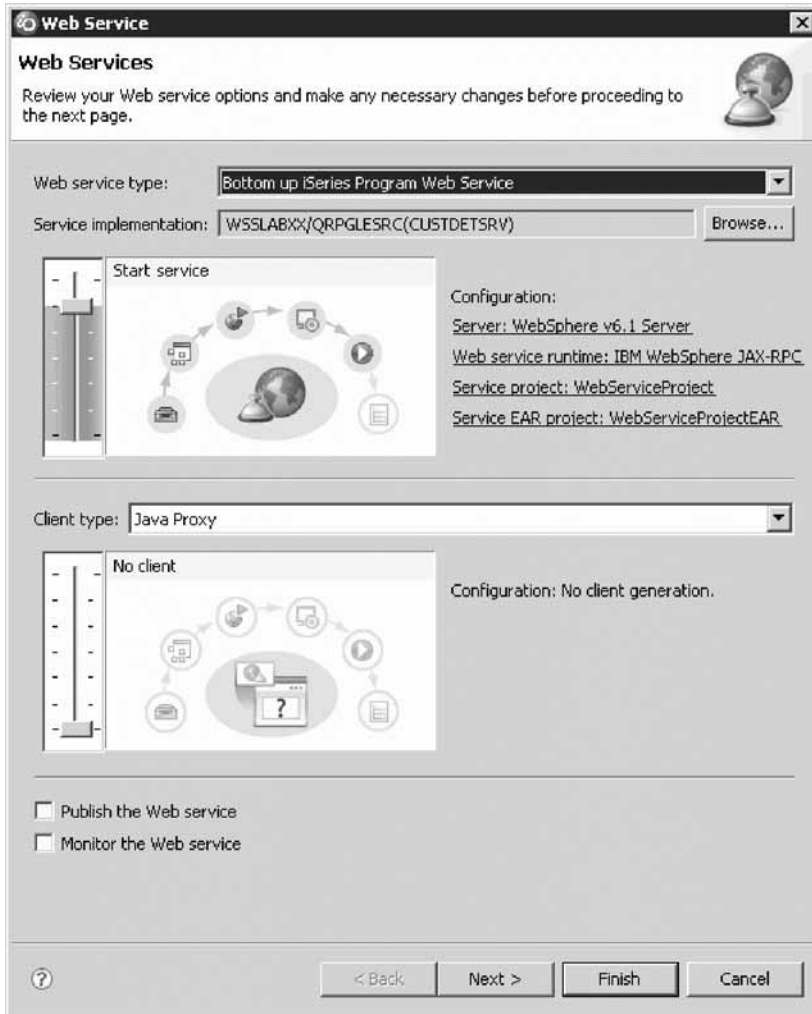
6. Right-click member **CUSTDETSRV**.

The RPG service program that provides the Web service is program **CUSTDETSRV**. The procedure the Web service will invoke is *readrecord*. The source for this program is in member **CUSTDETSRV** in source file **QRPGLSRC**. Since the service program object doesn't contain the program interface information needed to create the WSDL, you will need to identify the source member that contains the interface information.

7. Click **Web Services > Create Web Service** on the pop-up menu.



The Web Service wizard opens and you are ready to create an iSeries Web service.



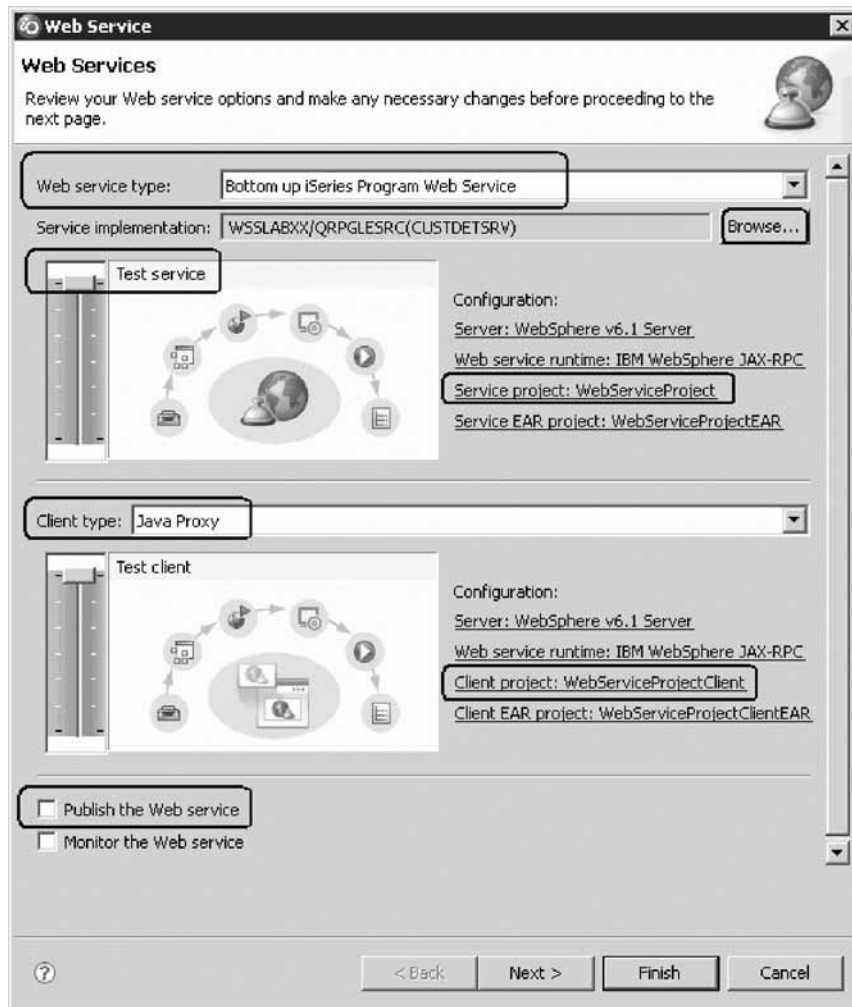
Lesson checkpoint

You learned the following:

- How to locate the source to create a Web service wrapper around an RPG service program
- How to invoke the Web Service wizard

Setting Web service options

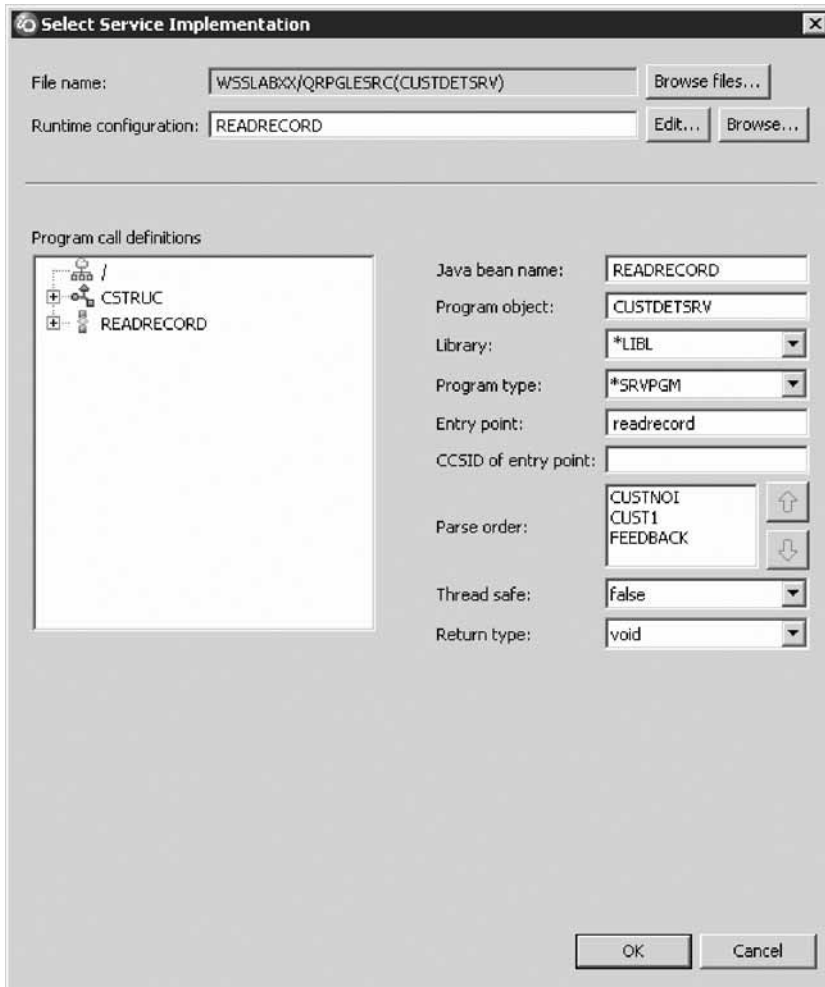
During this lesson you will complete the Web services page of the wizard.



To complete the Web services page:

1. In the Web Services window, in the **Web service type** field, **Bottom up iSeries Program Web Service** is already selected.
The default Web service type is set to bottom up iSeries Program Web Service. Keep this setting because you are wrapping an existing RPG service program with a Web service.
2. Move the top slider up and select **Test service** in the slider so that the wizard also creates a small test application for you.
The test application is covered later in this tutorial.
3. Move the bottom slider up and select **Test client** using the slider and make sure that the **Client type** is set to Java™ proxy.
This generates a client-side proxy through which you will test your Web service later in this tutorial. The proxy is covered later in this tutorial.
4. You will publish this Web service later in this tutorial, so make sure the **Publish the Web service** option is not selected at this time.
5. To specify the details of the service program, click **Browse** next to the **Service implementation** field.

The Select Service Implementation page opens.



Lesson checkpoint

You learned the following:

- How to set Web services options

Setting runtime information and selecting a service program object

During this lesson you will complete the Select Service Implementation page of the wizard.

In the Select Service Implementation page, you can specify how to connect and authenticate Web services that perform program calls to an iSeries host. By default the configuration file name is the same name as the Java bean name, with an extension of .config, and is stored in the source folder of the service project. You can change the authentication and runtime configuration values for the iSeries program that will be called from your Web service. You can browse for another configuration file or you can type in a new configuration file name.

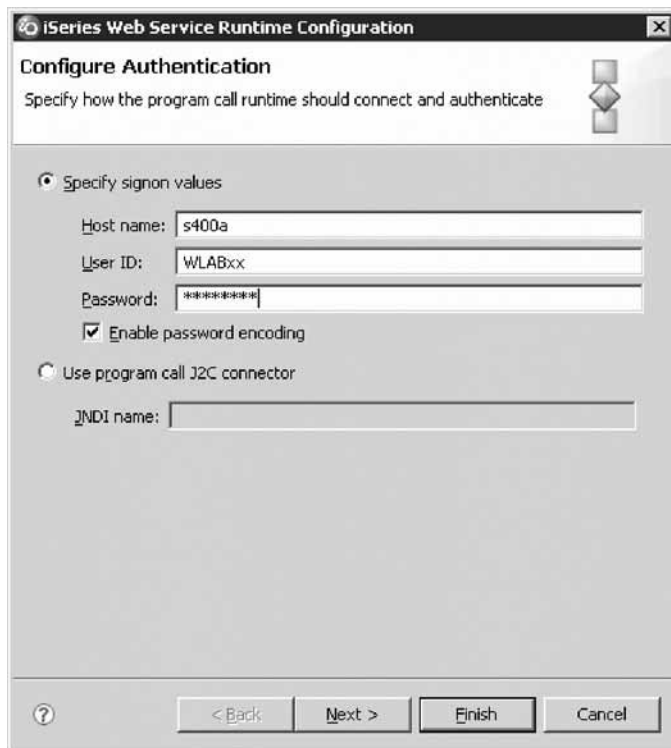
To complete the runtime configuration information:

1. Click **Edit** to open the iSeries Web Service Runtime Configuration dialog.



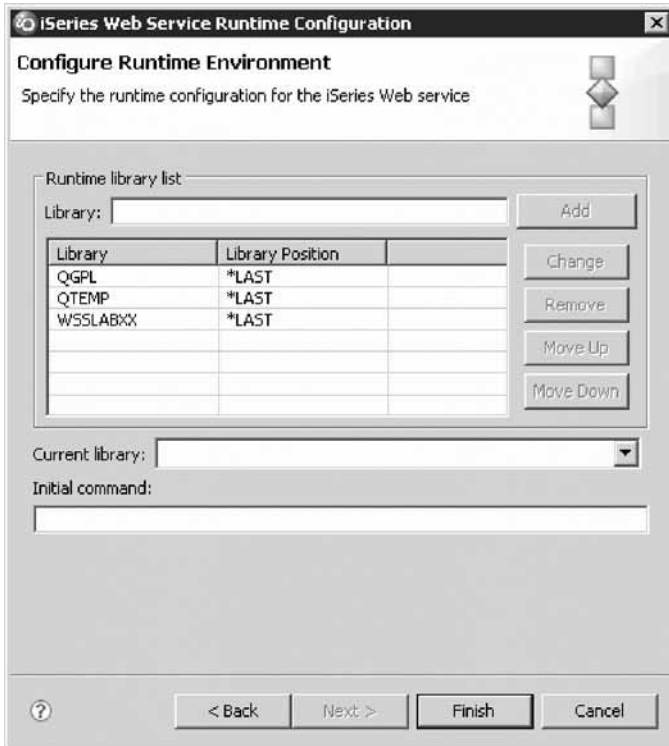
The default values for this dialog are taken from the connection associated with the source you used to launch the wizard. If the program object is somewhere else or a different authentication should be used, change these values accordingly.

2. Select **Specify signon values** radio button to enter the values for the host name, user ID and password. These values are stored in a runtime configuration file. When the Web service runs, these values are used when the iSeries host program is called.



3. Click **Next**.

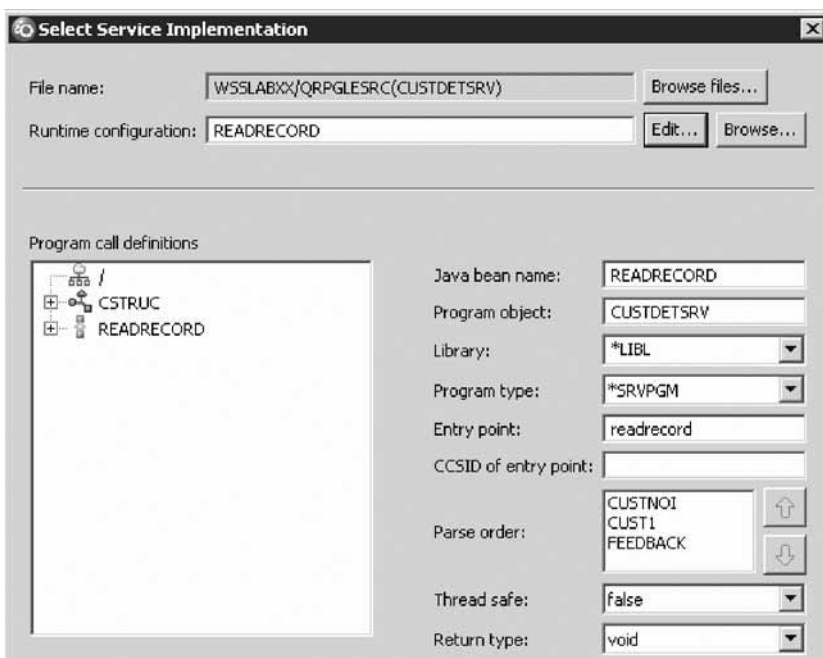
On the second page of the iSeries Web Service Runtime Configuration dialog, you define the runtime library list information for the Web service that will perform a program call to an iSeries host. Here you can add any runtime libraries that your program requires in the **Runtime library list** area. This results in the libraries being added to the library list on the iSeries server when the job is first created and before the iSeries program is invoked. The library you need (WSSLABxx) is already in your library list.



4. Click **Finish**.

Note: In addition to defining authentication and runtime values, you should ensure that host servers are running on your iSeries host. The servers are ***SRVMAP**, ***CENTRAL**, ***RMTCMD**, and ***SIGNON**. You start a host server with the STRHOSTSVR command. You can start all the servers by issuing the STRHOSTSVR *ALL command.

Next you select the program that will describe your Web service.

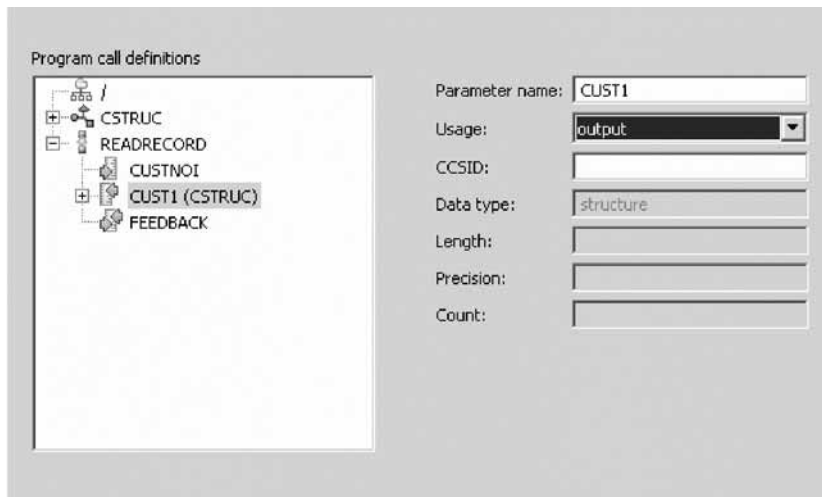


The program and entry point information are all correctly filled in, so you don't have to change these fields in this lesson.

Next you define parameters required by the Web service that you want to call. For a Web service, the default usage of input & output is often not ideal; you use the Select Service Implementation page to change the parameter's usage.

The program CUSTDETSRV requires one input parameter Customer number and it returns two output parameters: a data structure containing the detail customer information and a field containing process information.

5. Expand READRECORD in the left pane under **Program call definitions**.
6. Select the first parameter CUSTNOI.
7. In the right pane, change the **Usage** field from input & output to **input** to indicate that this parameter is used to pass a value from the Web Service Client to the program.
8. In the left pane select the CUST1 parameter.
9. In the right pane, change the **Usage** field from input & output to **output**, indicating that this parameter is used to pass a value from the program to the Web Service Client.

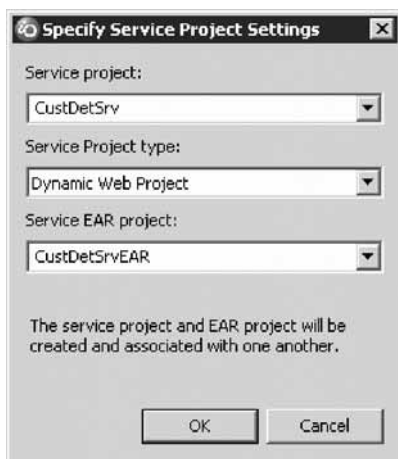


10. Do the same for the **FEEDBACK** parameter.
11. Click **OK**.

Next you change the names of the service project and client project. This is so you can easily identify these Web service projects from other Web service projects that you may have.

12. Click **Service project: WebServiceProject**.

The Specify Service Project Settings dialog opens.



13. Select the **Service project** field and enter a new name for this project, for example, CustDetSrv.

Note: The Service EAR project name is automatically updated.

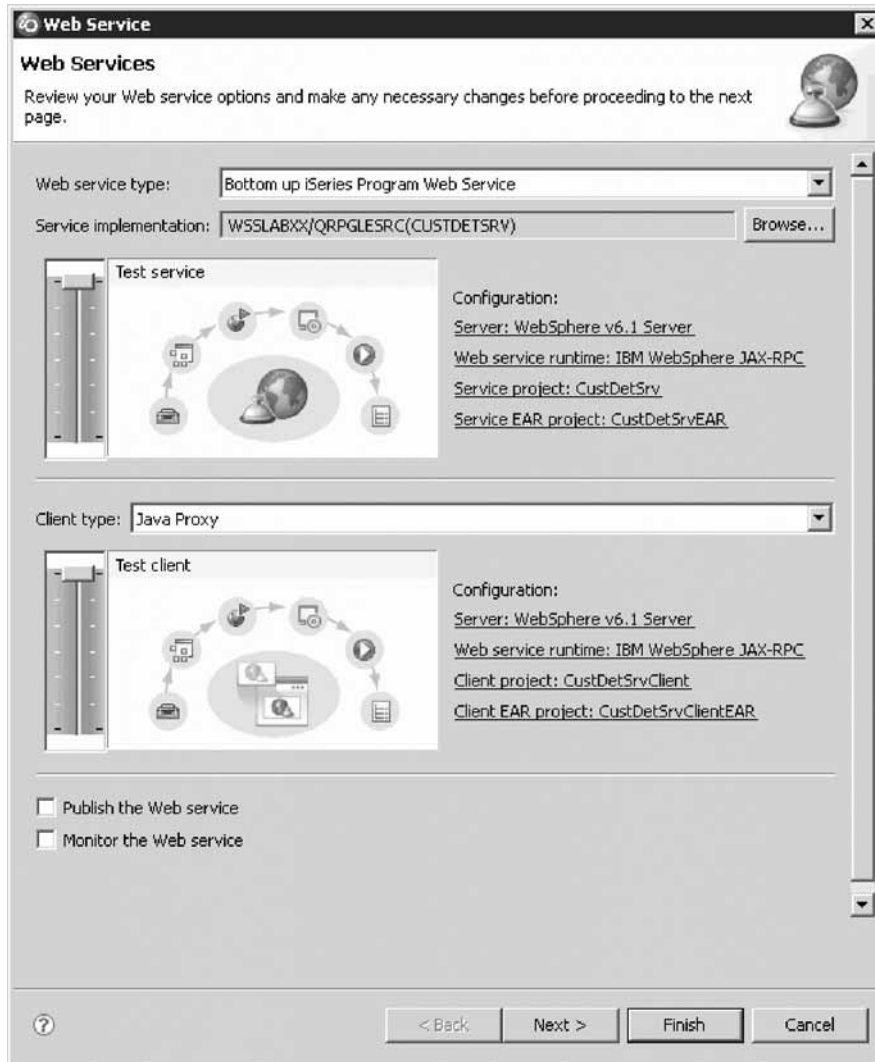
14. Click **OK**.
15. Click **Client project: WebServiceProjectClient**.
The Specify Client Project Settings dialog opens.



16. Select the **Client project** field and enter a new name for the client project that is used to test this Web service, for example, CustDetSrvClient.

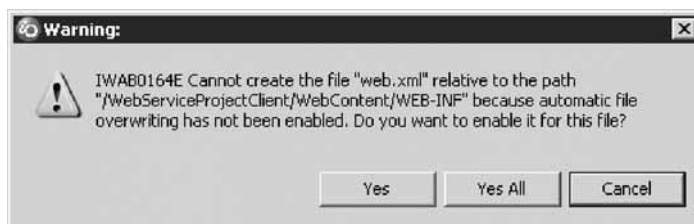
Note: The Client EAR project name is automatically updated.

17. Click **OK**.
Here you see the updated Web Services wizard page:



18. Click **Finish**.

If you see a message about overwriting files, click **Yes All**.



The wizard will close, and the Web service will be created.

Be patient as the two Web projects are created and the EAR projects are published to the WebSphere Test Environment.

Lesson checkpoint

You learned the following:

- How to set runtime information and select a program for your Web service

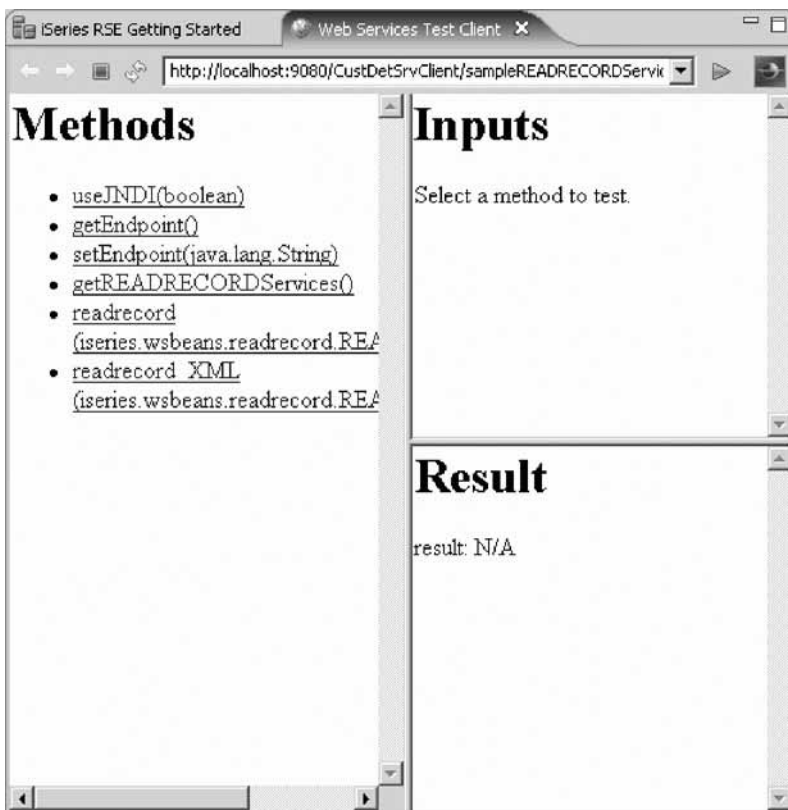
Testing the Web service

Since you specified that you want to test this Web Service the wizard launches a Web Services Client interface automatically in a Web browser.

The sample application is launched in the Web browser at the following URL: `http://localhost:9080/CustDetSrvClient/sampleREADRECORDServicesProxy/TestClient.jsp`.

Important: Sometimes the wizard tries to open the JSP above before it has been published to the Server, resulting in the Browser being opened with a message HTTP Error Code : 404 (for example, could not find the file). If that happens, ensure that generated resources have been published to the server (the progress monitor at the bottom-right shows a message while resources are being published to the server,

for example: `Publishing to WebSph.....: (0%)`). Then right-click in the Browser and select **Refresh** on the pop-up menu.



The sample application can be used to test the Web service by selecting a method, entering a value for the method, and clicking Invoke. The result of the method will display in the Result pane.

The wizard generates for you a simple test application; because the wizard already created the test application when it created the proxy, all you need to do now is launch the application. Since you selected the **Test the Web service** option in the Web Services wizard, the test application is already up and running.

To test the Web service:

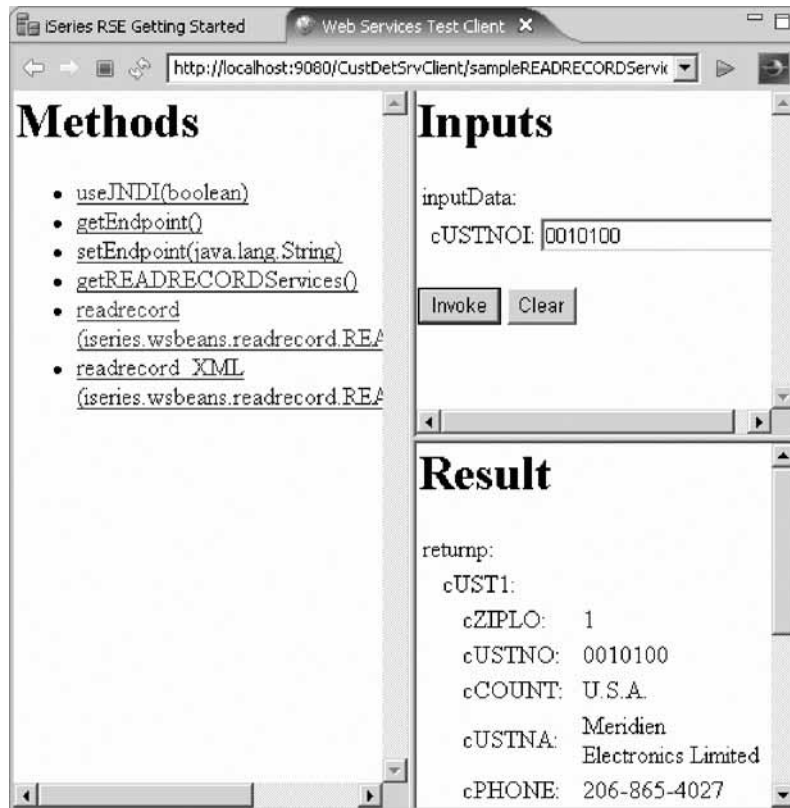
1. When the TestClient.jsp is launched, click the method `readrecord(iseries.wsbeans.readrecord.READRECORDInput)`.

The Inputs page contains an entry field for the input parameter data to be sent to the Web service.

2. Enter the value 0010100 for the CUSTNOI field and click **Invoke**.

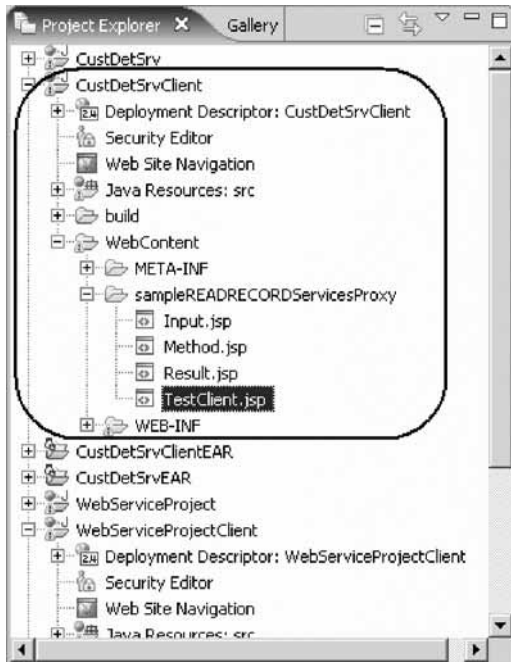
The Web service (your RPG program) is invoked and the data from the entry field gets passed to the program.

The results display in the Result pane.



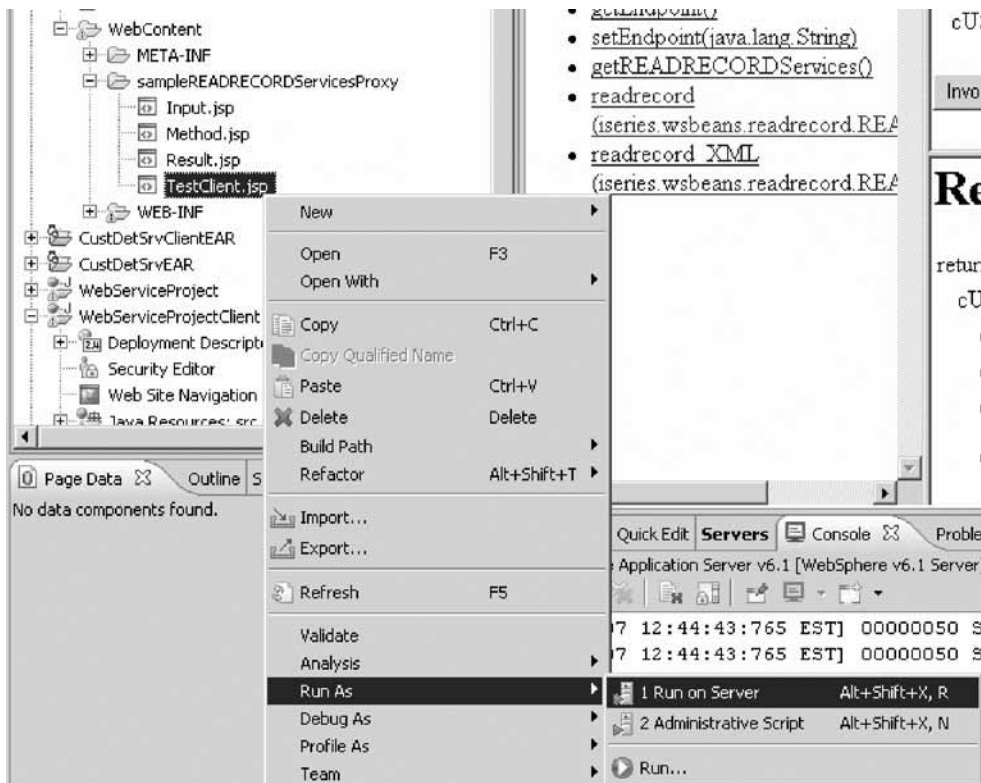
Note: If the output needs to be in XML format, then you can invoke the `getdata XML` method.

You can also launch the `TestClient.jsp` from the Web perspective as the test application is a simple JavaServer Pages (JSP) file that uses the proxy. It is created within the `CustDetSRVClient` project under the `WebContent\sampleREADRECORDServicesProxy`.



To launch the TestClient.jsp from the Web perspective:

3. Open the Web perspective: **Window > Open perspective then Other**. Select **Web** from the list of perspectives. Click **OK**.
4. Expand Dynamic Web Projects.
5. Expand **CustDetSRVClient** then **WebContent**.
6. Expand **sampleREADRECORDServicesProxy**.
7. Right-click **TestClient.jsp** and select **Run As > Run on Server**.



8. Select the WebSphere Application Server V6.1 server, then click **Finish**.
9. In the left pane, select which Web service method you want to invoke – in this case, `readrecord(iseries.wsbeans.readrecord.READRECORDInput)`.
10. The top-right pane contains a form matching the input arguments to the service. Type in some input data, for example, 0010100, then click **Invoke**.
11. The lower right pane displays the output of the service – the customer information that was created.

Lesson checkpoint

You learned the following:

- How to test the Web service through the Web Service wizard
- How to test the Web service through the TestClient.jsp

Module summary

You have finished developing the Web service and have tested it to make sure that it works. The next step is to package your application so that you can deploy it to WebSphere Application Server for iSeries.

Lessons learned

You have learned how to:

- Start the product
- Set the default workspace
- Check the Remote System Explorer perspective is open
- Set up a new connection to an iSeries server
- Locate an RPG source file you want to create a Web service from
- Start the Web Service wizard
- Select Web Service options (create a Web project, generate a proxy, test the Web service)
- Specify runtime information and select the program object
- Test the service in the test client

Deploying a Web service (optional)

In this module, you learn how to deploy your Web service. You use the workbench to deploy Web services.

Learning objectives

- Change the address information in the WSDL file for the server project to reflect the new port number in the WebSphere Application Server that it gets installed on.
- Use the Remote System Explorer to move the EAR file from your local drive to the integrated file system on the iSeries server
- Install the EAR file on the server
- Specify the virtual host for the deployed Web module included in the EAR file
- Start the installed Web service application
- Change the address information of the server project to listen to the port number for the WebSphere Application Server.
- Test the deployed Web service application

This module should take approximately 10 minutes to complete.

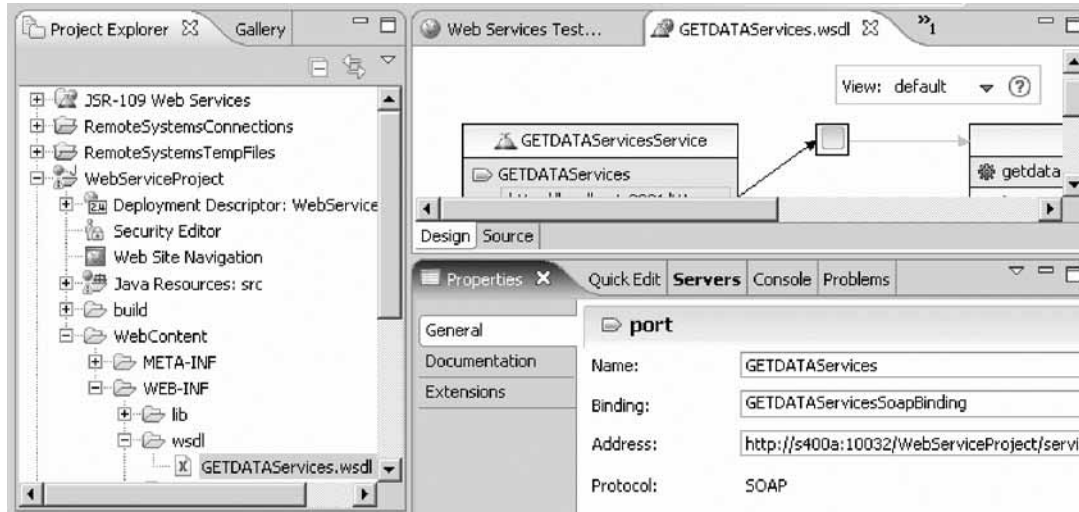
Modifying the service project WSDL

Since you want to move the Web service from running in your local WebSphere Test Environment instance to running on an iSeries WebSphere Application Server instance, you will need to change the project's service interface file, WSDL, to point your Web service users to the correct server.

This lesson will show you how to locate the WSDL file and how to change the server information.

In the Web perspective, to modify the address information:

1. In the **CustDetSrv\WebContent\WEB-INF\wsdl** folder, double-click the **READRECORDServices.wsdl** file, which opens the WSDL Editor.



2. Expand READRECORDServicesService from the Services section, if not already.
3. Select READRECORDServices.

The Properties view contains the detail information.

Change the address information to reflect the new WebSphere Application Server location and port.

As an example, change the URL from `http://localhost:9081/CustDetSrv/services/READRECORDServices` to `http://s400a:10032/CustDetSrv/services/READRECORDServices`

4. Select the address field in the properties editor.
Ask your administrator for the HTTP server port number, for example, 10032 and server name.
5. Close and save the WSDL file.

You have modified the service project READRECORDServices.wsdl file and it now contains the correct server information for the new location it will be installed at.

Lesson checkpoint

You learned the following:

- How to modify the service project WSDL

Building the EAR file

The last step is to package your application as an Enterprise Archive (EAR). The J2EE standard has a concept of an Enterprise Archive File (EAR). This file is a zip file that contains all information about your Web application. In the workbench, this file is built automatically for Web projects. Since your Web service project is a Web project, the CUSTDETSRVEAR file exists already. You only have to move it to the

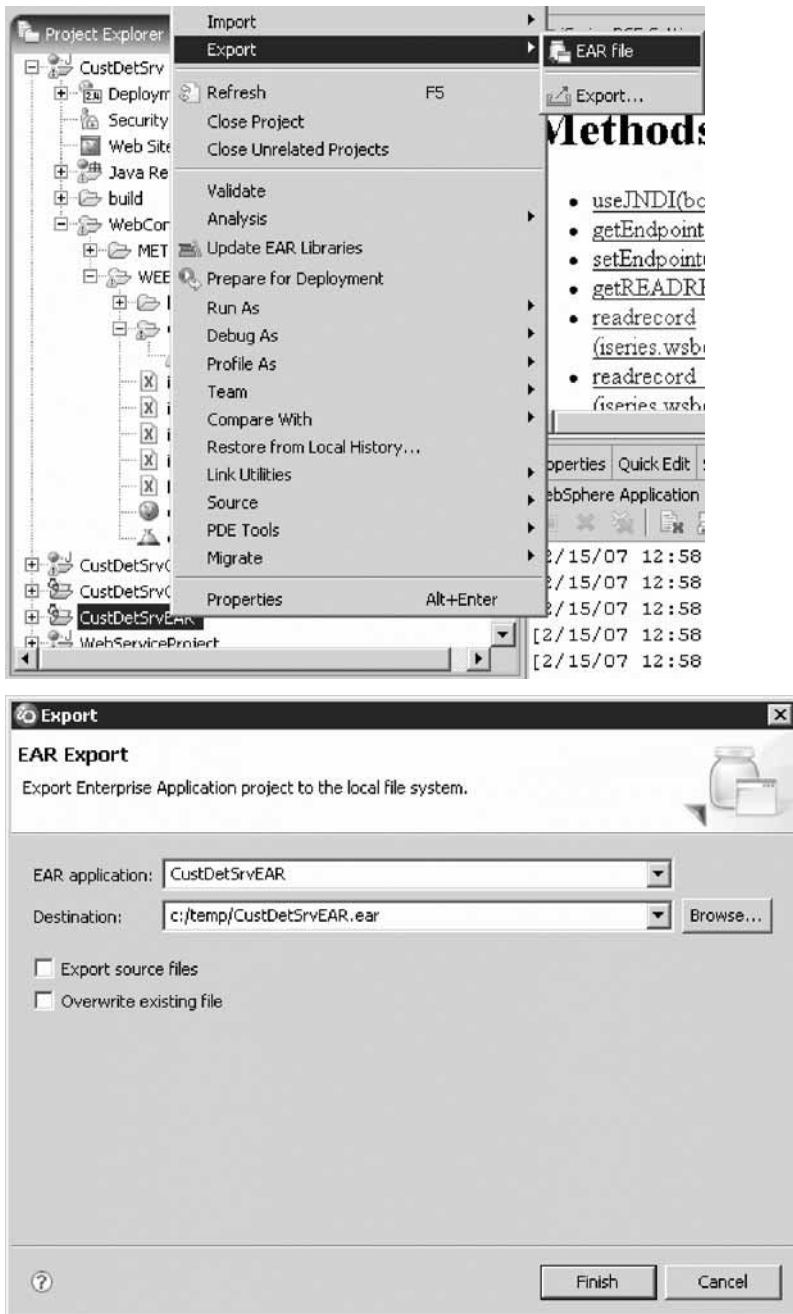
application server, point to it, and then install the application. There is no need to know about the structure of the application. Everything is handled in the workbench.

Note: Check with your system administrator to ensure that your user ID has the necessary authority to create and access IFS folders.

To build an EAR file:

1. Rebuild your project by selecting it, and then by selecting **Project > Clean** from the workbench menu. Clean discards all build problems and built states and rebuilds the projects from scratch.
2. In the Project Explorer, right-click CustDetSrvEAR and select **Export > EAR file**.


An EAR file is a compressed Enterprise Application Archive, and it contains all the files needed for the Web application.

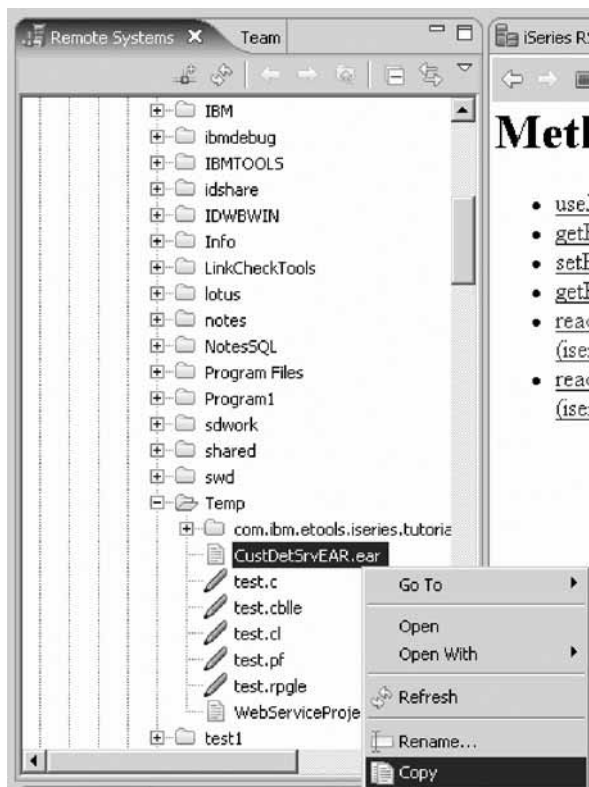


- In the **Destination** field, specify where you want to put the EAR file by entering the drive and directory structure on the local file system. For example, C:/temp/CustDetSrvEAR.ear.
- Click **Finish**.

You have exported the EAR file to the local file system.

Now that you have exported the EAR file to your local hard drive, you need to move it to a location where your WebSphere Application Server is running. In this tutorial, you use a server on an iSeries system so you have to copy the EAR file to the IFS of the iSeries server. You have two choices: you can map the IFS as a drive or you can use Remote System Explorer to move the EAR file from your local drive to the IFS.

- Open the Remote System Explorer. Look for the RSE icon  in the taskbar of the workbench and click it.
- In the Remote System Explorer perspective, locate the directory where you placed the EAR file. In the Remote Systems view, expand **Local** then **Local Files** and then **Drives**.
- Right-click the EAR file and click **Copy** on the pop-up menu.



- Expand **IFS Files** in the Remote Systems view for your iSeries connection.
- Right-click a directory, for example, **Home\wservxx** directory and click **Paste** on the pop-up menu. If you do this on your System i™, select any directory on your IFS you have authority to.

You have exported the EAR file for the server project and imported into an IFS directory.

Lesson checkpoint

You learned the following:

- How to build an EAR file

Installing the enterprise application

At this point, you already need to have set up and started the WebSphere Application Server and an associated HTTP server on your iSeries server. You also need to find the port number of the HTTP servers Administration Console (referred to as port#a).

To install the enterprise application:

1. Open any Web browser and type `http://s400a:port#a/` in the **Address** field.

Tip: The default administrative port number for the HTTP server is 2001. The Logon dialog appears.

2. Type your iSeries user ID in the **user ID** field.
3. Type your iSeries password in the **password** field.
4. Click **OK**. The browser opens.



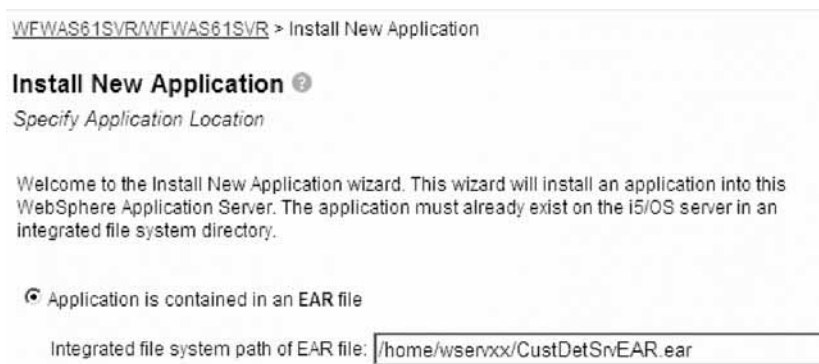
5. Click the **IBM Web Administration for i5/OS** from the i5/OS Tasks menu. The Administration pages for the HTTP and application servers display.
6. Select the **Manage** tab then the **Application Servers** tab. The Manage WebSphere Application Server page opens.



7. Select your WebSphere Application Server instance from the **Server** list and start this application server instance by clicking the start button if it is not started already.



8. Click **Install New Application** in the right pane. The Install New Application page opens.



9. Click the **Application is contained in an EAR file** radio button.
10. Specify the location of your EAR file; (the file that you exported in the previous step). You can use the Browse button to locate the file.
11. Click **Next**. A page appears with options for application installation.

WFWAS61SVR/WFWAS61SVR > Install New Application

Install New Application

Provide Options to Perform the Install

Specify application deployment options

Application name: 

Directory application installed to: /QIBM/UserData/WebSphere/AppServer/V61/Base/profiles/M

Pre-compile JSPs 

Note: Pre-compiling JSPs can have significant performance impacts. When enabled, the JSP compiled at installation time, causing the application install to take longer. When disabled, th

12. In the **Application name** field, type CustDetSrv.
13. Click **Next**. The Map Virtual Hosts Page appears.

WFWAS61SVR/WFWAS61SVR > Install New Application

Install New Application

Map Virtual Hosts for Web Modules

Specify the virtual host you want to associate with the Web modules contained in this application. Web modules can be installed on the same virtual host or dispersed among several virtual hosts.

Map Web modules to virtual hosts: 

Web module	Virtual host	Web server
CustDetSrv	<input type="text" value="default_host"/>	<input checked="" type="checkbox"/> WFWBSRVE61

Because the EAR includes a Web module, you need to specify which virtual host it should be deployed onto. The next window in the Application Installation wizard allows you to override existing bindings in the EAR file and specify the virtual host.

14. Leave the default value for **virtual host** which will be associated with the Web module.
15. Click **Next**. The application summary page displays.

Summary

When you click **Finish** the installation will be started for the following application.

Integrated file system path of EAR file: /home/wserrxx/CustDetSrvEAR.ear

Application name: CustDetSrv

Directory application installed to: /QIBM/UserData/WebSphere/AppServer/V61/Base/profile

Pre-compile JSPs: Enabled

Map Web modules to virtual hosts:

Web module	Virtual host	Web server
CustDetSrv	default_host	<input checked="" type="checkbox"/> WFWBSRVE61

16. Click **Finish**. After the successful installation you see a page where you can manage the installed applications.

Current Configuration	Profile: "WFWAS61SVR"	Server: "WFWAS61SVR"
Manage Virtual Hosts <input checked="" type="radio"/> default_host <input checked="" type="radio"/> admin_host	Manage Installed Applications <input checked="" type="radio"/> V6013-ENU-Web-061122 <input checked="" type="radio"/> CustDetSrv <input checked="" type="radio"/> isclite	Manage Data Sources <input checked="" type="radio"/> DefaultEJBTimerDataSource <input checked="" type="radio"/> Default Datasource

You have completed the Web application server installation.

Lesson checkpoint

You learned the following:

- How to install the EAR file
- How to deploy the Web module and complete the installation

Manage the installed application

To manage the installed application:

1. Click **Manage Installed Applications** in the Manage Application Server page. A page with the installed applications opens. Wait for the status to change. Click **Refresh** to see the status change to stopped.

[WFWAS61SVR/WFWAS61SVR](#) > Manage Installed Applications

Manage Installed Applications ?

Data current as of Feb 21, 2007 6:20:09 AM.

Installed applications: ?

	Application name	Status	Enablement
<input type="radio"/>	V6013-ENU-Web-061122	<input checked="" type="radio"/> Running	Enabled
<input type="radio"/>	CustDetSrv	<input checked="" type="radio"/> Stopped	Enabled
<input type="radio"/>	isclite	<input checked="" type="radio"/> Running	Enabled
<input type="radio"/>	hats1109_wfhats1109	<input checked="" type="radio"/> Running	Enabled

Close

2. Click the **wservxx** radio button.

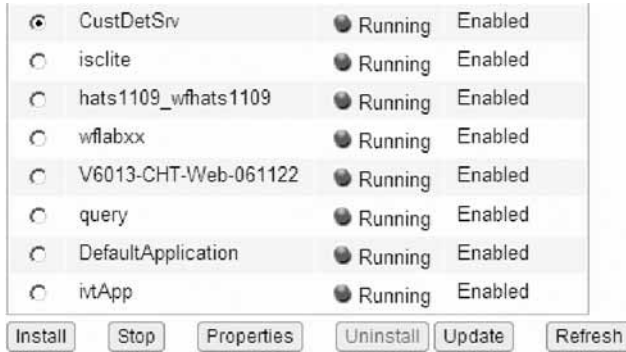
<input checked="" type="radio"/>	CustDetSrv	<input checked="" type="radio"/> Stopped	Enabled
<input type="radio"/>	isclite	<input checked="" type="radio"/> Running	Enabled
<input type="radio"/>	hats1109_wfhats1109	<input checked="" type="radio"/> Running	Enabled
<input type="radio"/>	wflabxx	<input checked="" type="radio"/> Running	Enabled
<input type="radio"/>	V6013-CHT-Web-061122	<input checked="" type="radio"/> Running	Enabled
<input type="radio"/>	query	<input checked="" type="radio"/> Running	Enabled
<input type="radio"/>	DefaultApplication	<input checked="" type="radio"/> Running	Enabled
<input type="radio"/>	itApp	<input checked="" type="radio"/> Running	Enabled

Install Start Properties Uninstall Update Refresh

3. Click **Start**. If the Start button is not enabled, start the server. Click the start server button to the left of the server name.



Here you see the application is started.



You have started the application.

Lesson checkpoint

You learned the following:

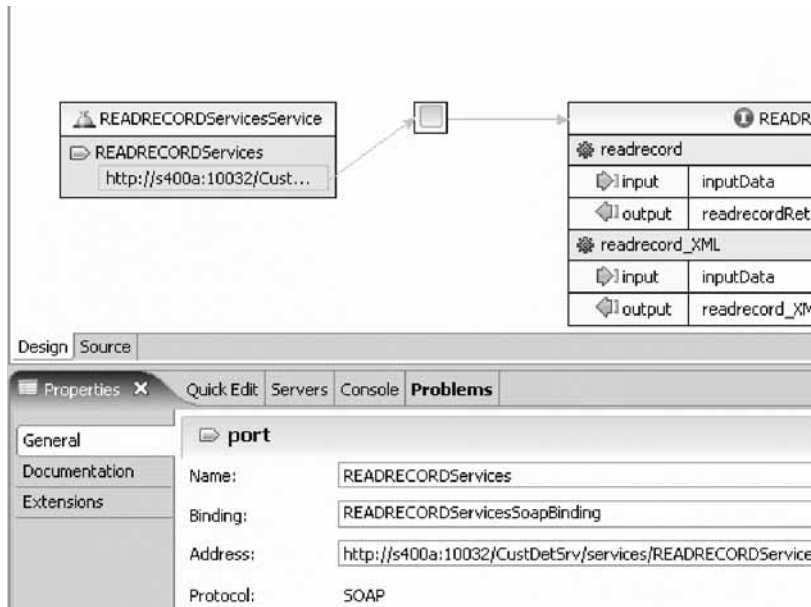
- How to start the application

Modifying the client project WSDL

The Web service client is still setup to work with the Web Service on your local workstation. In order to redirect it to the Web Service on the System i for testing purposes, you need to change the server name and port number.

Back in the workbench and the Web perspective, to modify the address information:

1. Expand the CustDetSrvClient project that was created from the Web Services wizard.
2. Expand the WebContent directory.
3. Expand the WEB-INF directory.
4. Expand the wsdl directory.
5. Double-click on the .wsdl file. The editor opens.
6. In the Service pane, expand READRECORDServicesService
7. Expand the READRECORDServices.
8. Select wsdlsoap address.
9. In the Properties view, change the first part of the value of the location property to i5:YYYY where i5 is the name of the server and YYYY is the port number. This is the port number that you used in the server WSDL as well. The example name and number used there was s400a:10032. Don't change the rest of the URL as it is still valid. Close and save the WSDL file.



You have modified the client project WSDL file.

Lesson checkpoint

You learned the following:

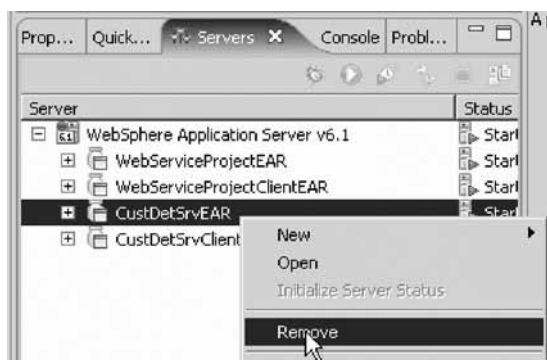
- How to change the address of the WSDL client project

Testing the Web service deployed and running on iSeries

Now after you modified the WSDL file earlier to point to the iSeries server and you are ready to test the service that is now deployed and running on the iSeries.

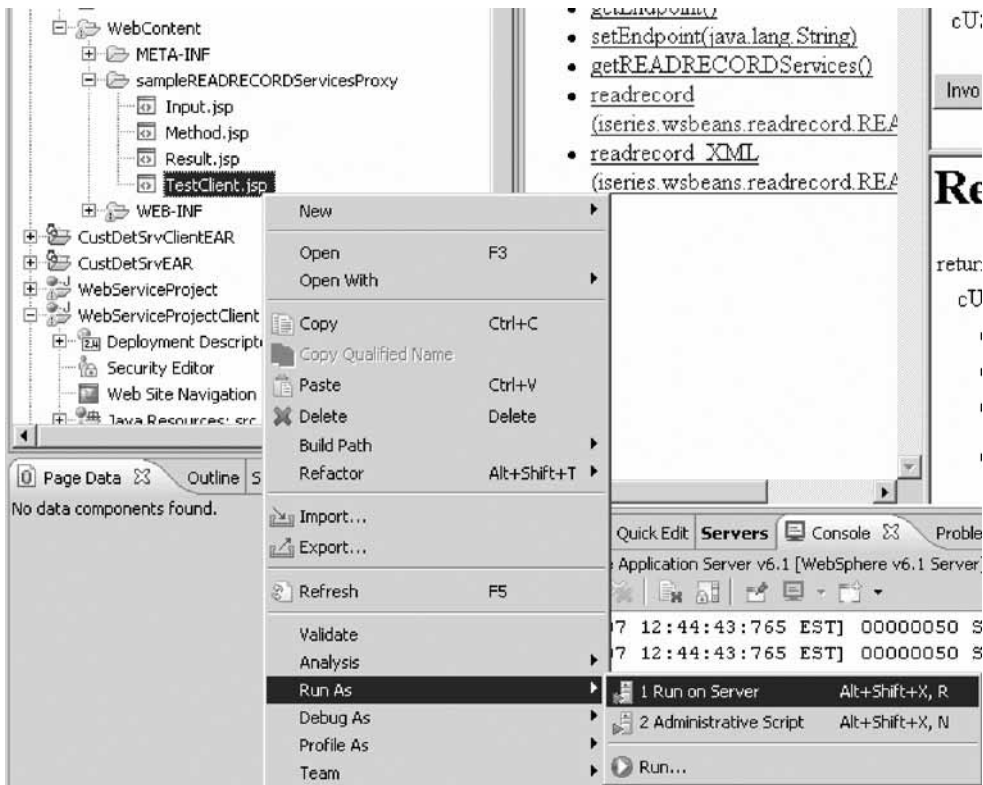
To test the Web service deployed and running on iSeries:

1. In the Servers view, expand the server WebSphere Application Server v6.1
2. Right-click CustDetSrvEAR and select **Remove** from the pop-up menu.



You remove this EAR file in order that you don't invoke the local service.

3. In the Project Explorer view, expand CustDetSrvClient then WebContent.
4. Expand sampleREADRECORDServicesProxy.
5. Right-click TestClient.jsp and select **Run As > Run on Server**.



6. Select the existing server, then click **Finish**.
7. In the left pane, select which Web service method you want to invoke -- in this case, **readrecord(iseries.wsbeans.readrecord.READRECORDInput)**.
8. The top-right pane contains a form matching the input arguments to the service. Type in some input data, for example, **0010100**, then click **Invoke**.
9. The lower right pane displays the output of the service -- the customer information that was created.

The deployed Web service opens.

Lesson checkpoint

You learned the following:

- How to test the deployed Web service

Module summary

In this module, you learned how to deploy a Web service.

Lessons learned

By completing this module, you learned about the following concepts and tasks:

- Change the address information of the server project to listen to the port number for the WebSphere Application Server.
- Use the Remote System Explorer to move the EAR file from your local drive to the integrated file system on the iSeries server
- Install the EAR file on the server
- Specify the virtual host for the deployed Web module included in the EAR file
- Start the installed Web service application

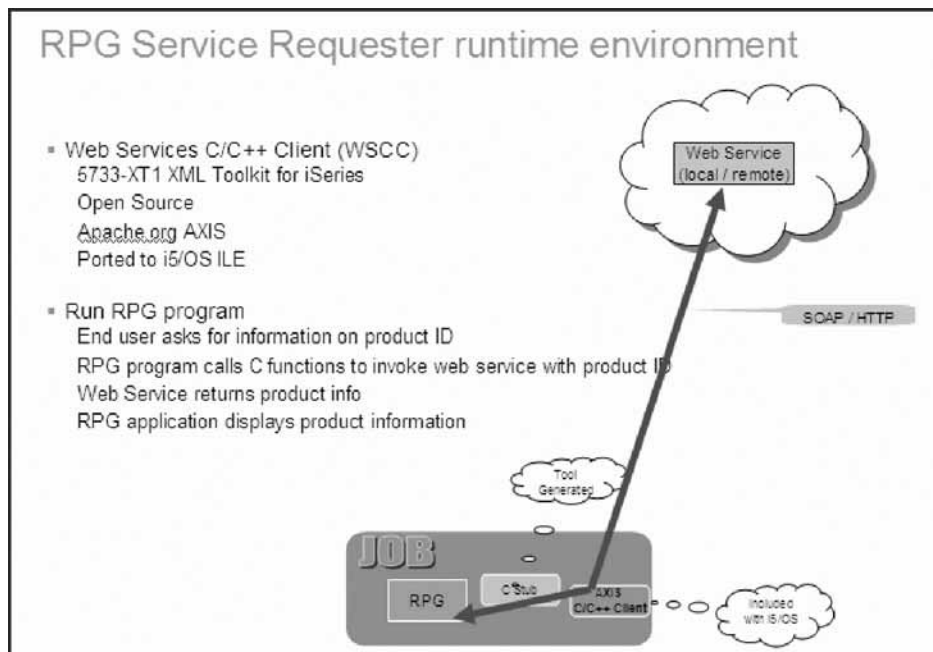
- Change the address information of the server project to listen to the port number for the WebSphere Application Server.
- Test the deployed Web service application

If you want to give the Web service information to other people, you can copy the WSDL file and send it to them.

Calling a Web service from RPG

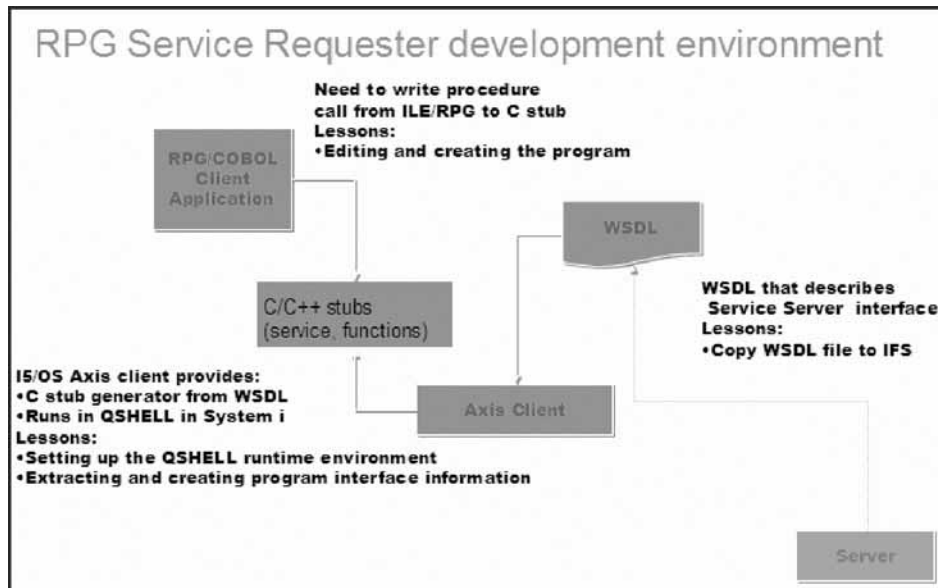
So far we have discussed invoking an RPG program or service program as a Web service. But what about calling a Web service from RPG. You don't need to know how the Web service is implemented or what platform it is running on. All you need is a WSDL file that describes the service. Then you have to create a way to invoke the Web service from RPG. In this module you will do this by using the AXIS client on i5/OS that is part of the XML Toolkit.

Note: Check with your system administrator to ensure that your user ID has the necessary authority to create and access IFS folders.



The RPG program that is using the Web service is a simple 5250 program that has a screen with an input field for a product ID, this product ID gets passed to the Web service, the Web service when successful returns a XML data stream with product information.

The RPG program parses the data from the XML stream and displays the product information received from the web service in the 5250 panel. What you will do in this tutorial to accomplish this task, is described in the next picture.



1. You will prepare the i5/OS QSHELL environment to be set up to run the C stub generator, that creates a C interface from a given WSDL file.
2. You copy the WSDL from your local desktop hard drive to the IFS file system on your System i.
3. You run the Stub generator to create the C functions from the WSDL
4. You change the existing RPG source to add the procedure calls for the C stubs to the program
5. You create the RPG program and bind the C functions
6. Done, you can try it out

Learning objectives

After completing the lessons in this module you will understand the concepts and know how to do the following:

- Set up an environment variable to a directory in the XML Toolkit
- Run the interface extraction utility in QSHELL
- Add a prototype to describe the program interface
- Add a call command to invoke the program
- Run the program

This module should take approximately 20 minutes to complete.

Setting up the QShell runtime environment

In this lesson, you learn how to setup the runtime environment by configuring environment variables, environment paths, and implementing the WSDL file.

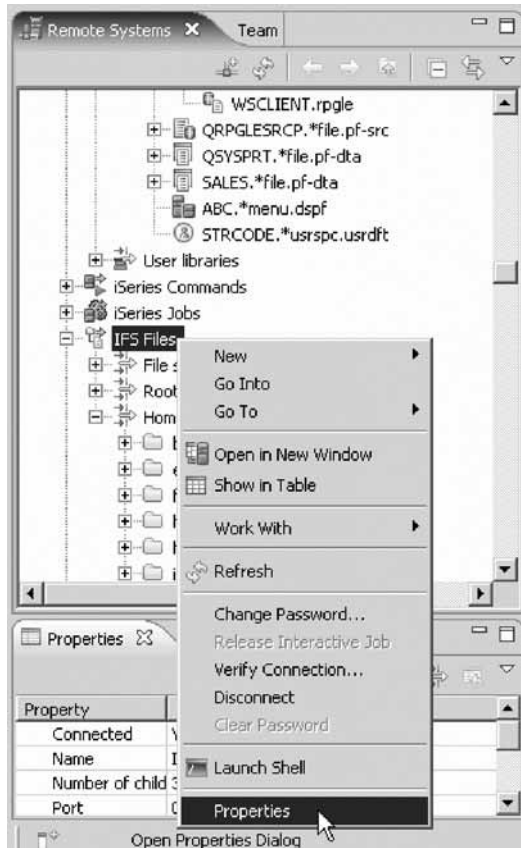
To run an existing Web service with the AXIS client on an iSeries, the XML toolkit must be installed on your iSeries system. In addition you must set up QSHELL by adding an environment variable entry to include a specific directory in this toolkit. This directory contains a shell script tool in the XML toolkit.

To add an environment variable:

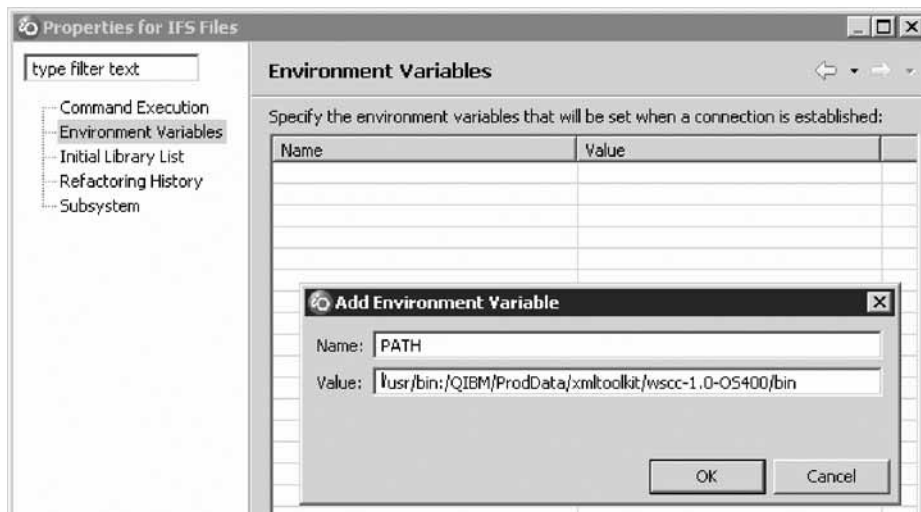
1. In the Remote System Explorer right-click a subsystem, for example, iSeries Objects, under your iSeries connection.
2. From the pop-up menu, select **Properties**.

The Properties dialog opens.

Now, you set the environment variable for the job that gets started when the RSE connection gets activated.



3. Select Environment Variables from the left pane of the window.



4. Click **Add(B)**

5. On the Add Environment Variable dialog, type PATH for the name and /usr/bin:/QIBM/ProdData/xmltoolkit/wsc-1.0-05400/bin for the value.

Tip: Make sure you use the correct case for the directory names and the name PATH.

6. Click **OK**.

The Environment variable is added to the Properties dialog.

7. Click **OK** to close the Properties dialog.

Next, you copy the WSDL file.

Lesson checkpoint

You learned the following:

- How to set up an environment variable to point to a directory in the XML toolkit

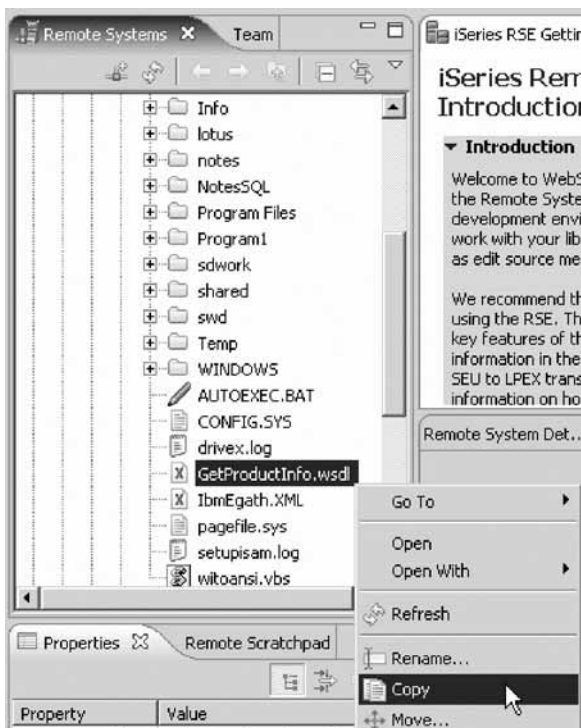
Copying the WSDL file to IFS

This WSDL file describes the interface of the Web service that you want to access from your RPG program. The WSDL file is on the C: drive of your workstation.

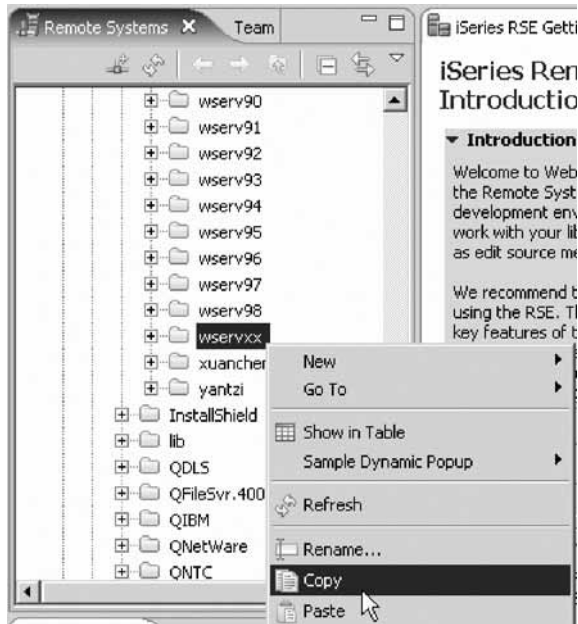
Note: Check with your system administrator to ensure that your user ID has the necessary authority to create and access IFS folders.

To copy the WSDL file:

1. In the Remote Systems view, expand **Local Files > Drives > C:**.
2. Locate the GetProductInfo.wsdl file. Right-click and select **Copy** on the pop-up menu.



3. In the Remote Systems view at the iSeries connection, expand IFS.
4. Expand the **Root file system > home**.
5. Locate the wservxx directory. Right-click and select **Paste** on the pop-up menu.



Now you have the WSDL file on the IFS and you can run a tool in the XML Toolkit to extract the program interface information from the WSDL file and provide a C interface for this Web service that you can then call from your RPG program.

You have also copied the WSDL file to an IFS directory on the iSeries. Now you are ready to run the XML Toolkit extraction utility in QSHELL.

Lesson checkpoint

You learned the following:

- How to copy the WSDL file to an IFS directory

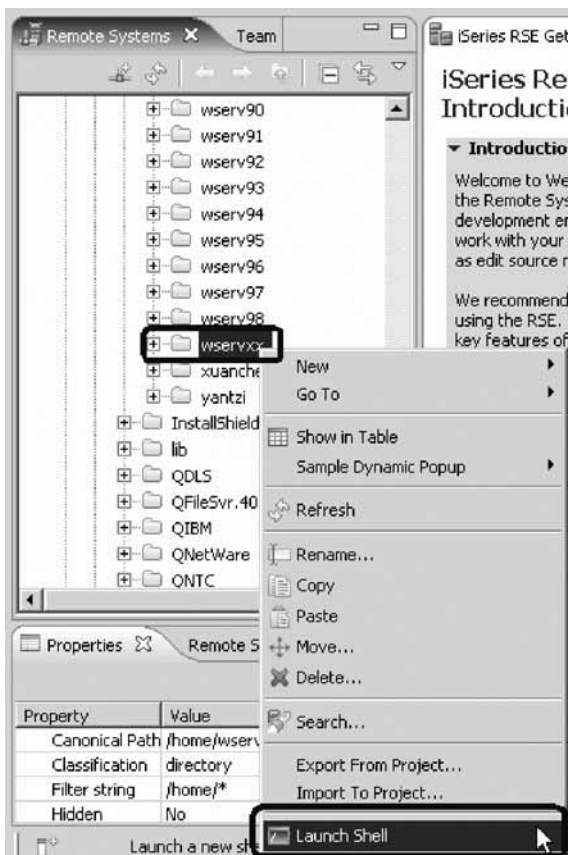
Extracting the Web service interface information and creating the program interface

In this lesson, you'll learn how to use the Remote System Explorer to extract the Web service interface information and create the program interface.

To extract the program interface information from the WSDL file, you need to run the interface extraction utility in QSHELL.

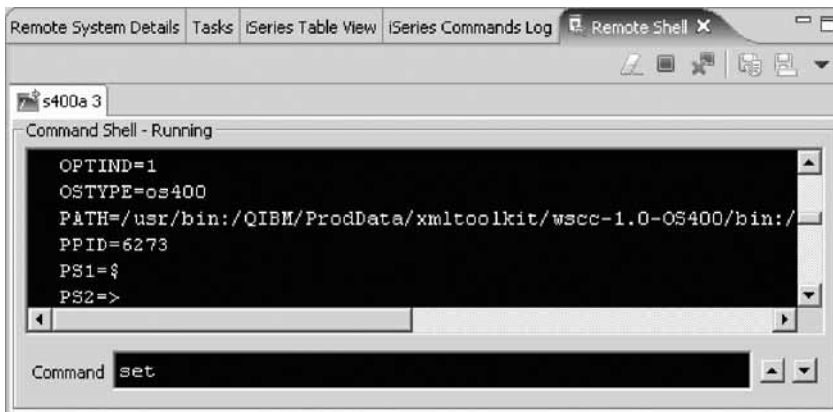
To extract the program interface information:

1. In the Remote Systems view, right-click the IFS subsystem to home\wservxx.
2. Select wservxx directory, and right-click it.
3. Select **Launch Shell**.



4. In the command line of the QSHELL command window, type the set command.

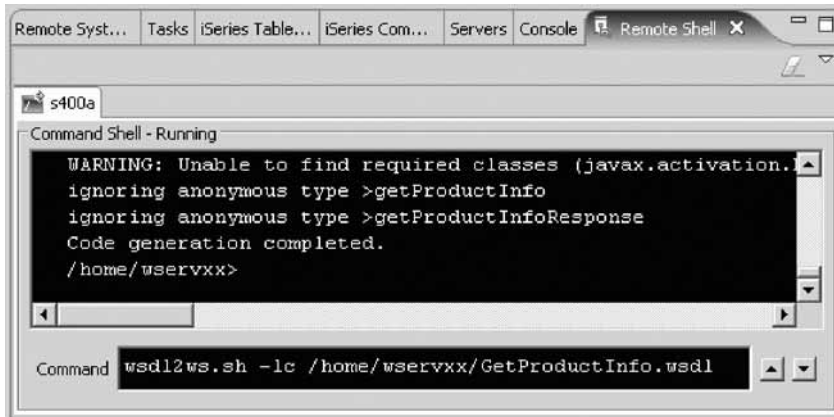
The set command shows you the current QSHELL environment setup. In the output window scroll up until you see the PATH variable to make sure it contains the directory information that you typed in previously when changing the Remote System Explorer subsystem properties.



5. In the command line, type the name of the interface extraction utility with its parameters: `wsdl2ws.sh -lc /home/wservxx/GetProductInfo.wsdl`

Tip: This is case sensitive and the `-lc` parameter has a lowercase letter 'l' not a number one '1'.

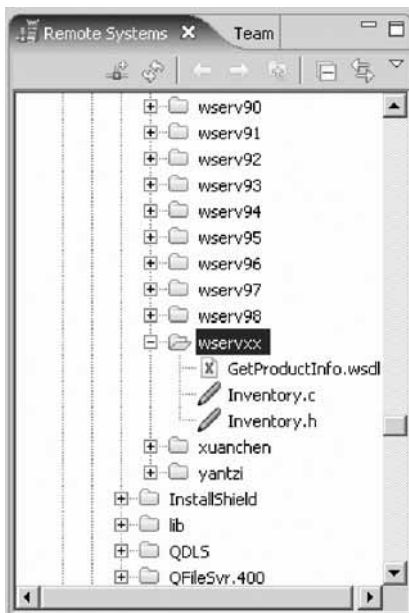
If you have setup everything correct, the output window shows that the code generation is completed. You can ignore the warning message that begins "Unable to find required classes...".



The `-lc` parameter tells the utility to generate C stubs instead of C++ stubs.

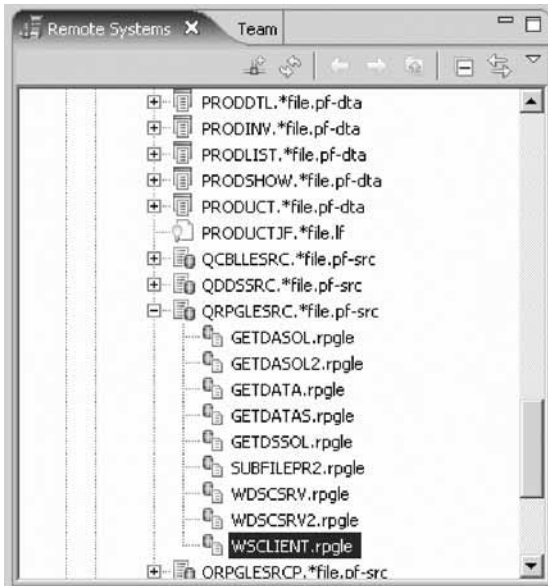
6. Expand your home directory and check that the files are really included.

The files containing the C stubs are generated into your home directory on the IFS which is `/home/wservxx`.



You should see the `Inventory.c` file and the `Inventory.h` file.

7. Double-click the `Inventory.c` file to see what was generated from the `wsdl` file. The interesting part is in the section labeled `methods` corresponding to the Web services methods. Here you can see the function `getProductInfo`:

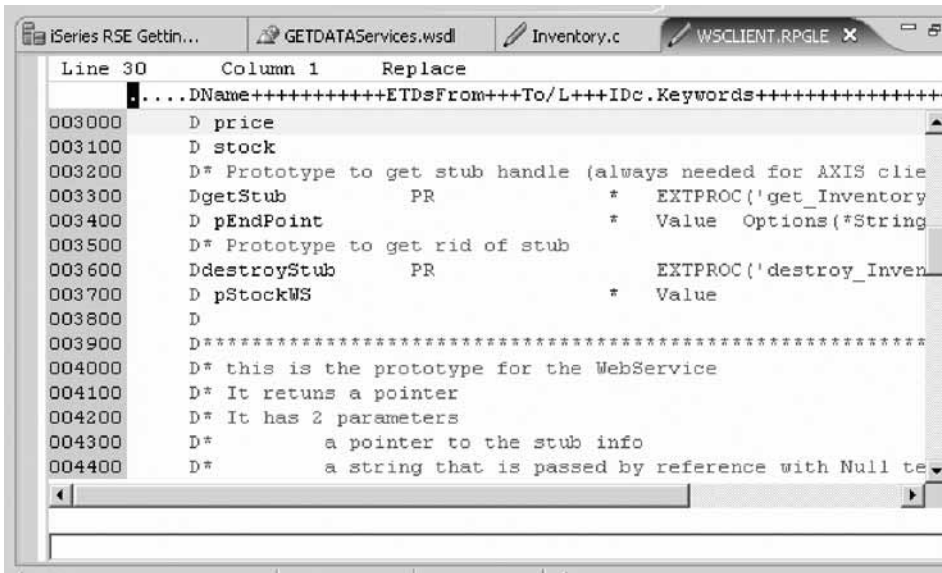


4. Double-click WSCLIENT.rpgle file.

The source editor opens.

This is an RPG program that is created for you so you don't have to key in the source. You need to change one line and check the logic relevant to the Web service invocation.

Let's look at the prototype first:



The prototype specifies that this function passes two parameters, the first for the handle to the Web service and the second for the product information which is Null terminated.

It returns a pointer, which actually points to the result string.

5. The first function you need to invoke sets up the Web service location and returns a handle for this Web service. You need to change the location information. Scroll down to the calculation specs to this area:


```

005200 FREE
005300 // this line tells the AXIS where the WebService is located
005400 // you will need to adjust this line
005500 // the return is a handle that will be used to the real WebService invocation
005600 // ----->
005700 // ----->
005800 stockWS = getStub('http://torasbcc:10032/intelWS/services/Inventory');
005900 // Loop to show screen until command key 3 is hit

```

Change the information to the location information which is the name of the system and the port number where the service is deployed.

- Next, you want to look at the actual invocation of the Web service. Scroll down until you see these lines:

```

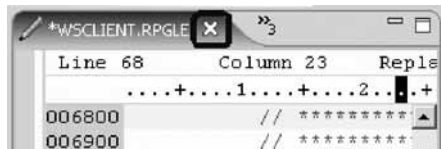
006900 // *****
007000 // invoke WebService
007100 // pass stub and prodid
007200 // returns pointer to string
007300
007400 xmldoc = getProductInfo(stockWS : %trim(prodid));
007500

```

The invocation of the getProductInfo Web service specifies that you pass a handle to the Web service information and you pass the Product identification that is the input from the 5250 screen. The return value is a pointer, that gets stored in variable xmldoc and the receiving string xmlstr is based on this pointer. This allows you to work with the data in the RPG program.

The remaining logic in the program just parses the data from the XML document in xmlstr and deals with exception handling.

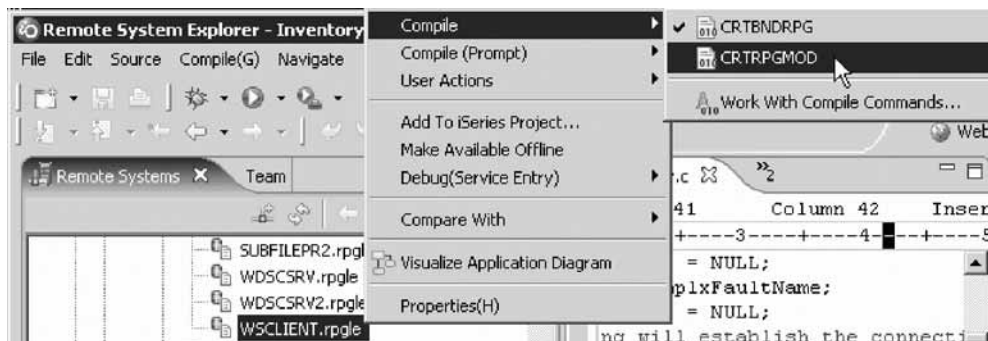
- You are ready to save your source. Click the X on the editor tab.



- Specify **Yes** when asked whether you want to save the source.

Next you create a module from this source. Since you will also have to bind the C functions to the program, you can't just create a program.

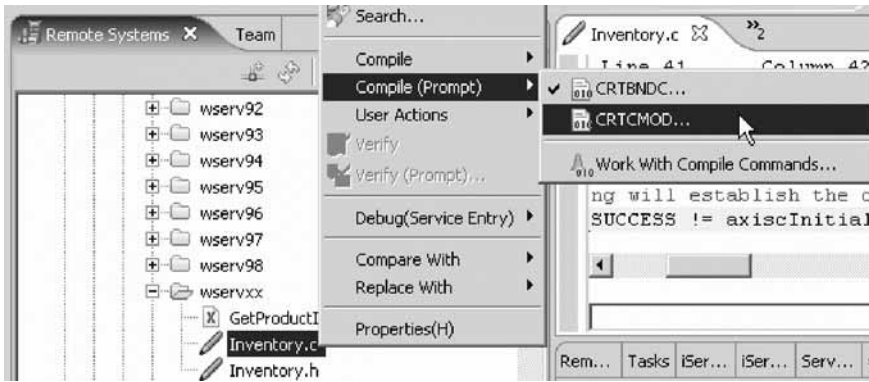
- Right click WSCLIENT.rpgle and select **Compile > CRTRPGMOD** on the pop-up menu.



It should compile without error, so don't worry about warning messages.

Now you need to create the C module from the c source file that has been created by the XML Toolkit.

- Scroll down to the IFS drive and your directory. Right-click the Inventory.c file and select **Compile (Prompt) > CRTCMOD** on the pop-up menu.



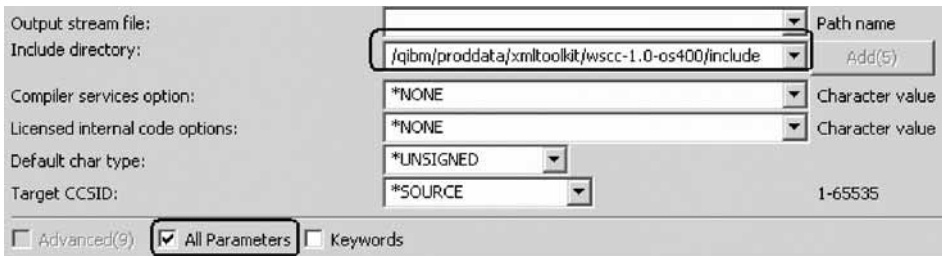
The Create C Module dialog opens.

- Change the library parameter to wsslabxx.

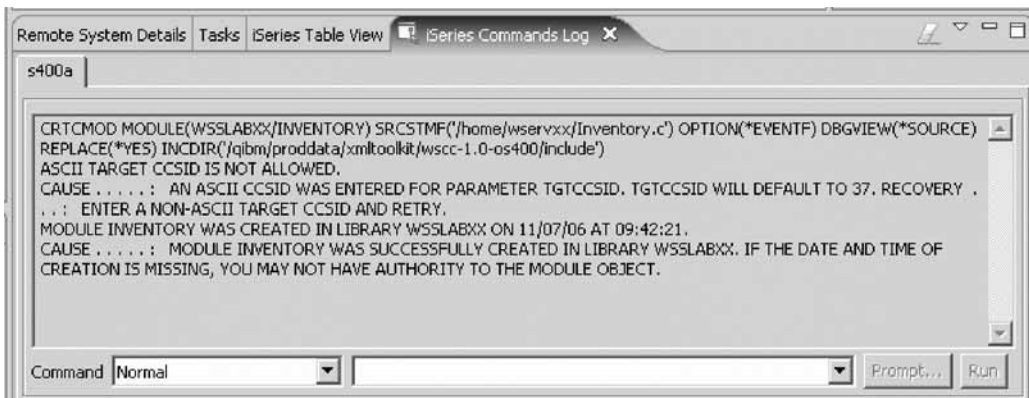


Next you need to add the include directory of the XML Toolkit which is /qibm/proddata/xmltoolkit/wsc-1.0-os400/include

- Select the **All Parameters** check box.
- Scroll down the dialog until you see the include directory. Add the include directory shown.

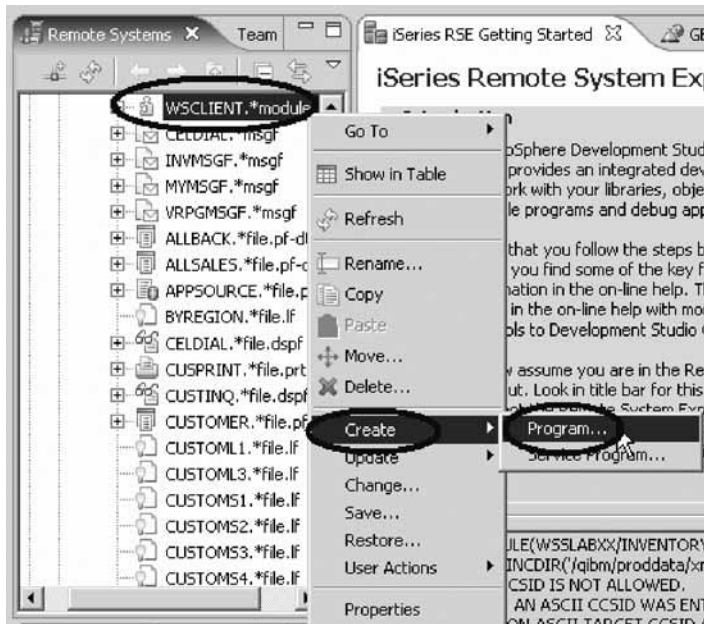


- Click **OK** to compile this source.
Check the iSeries Commands log for a successful compile message.

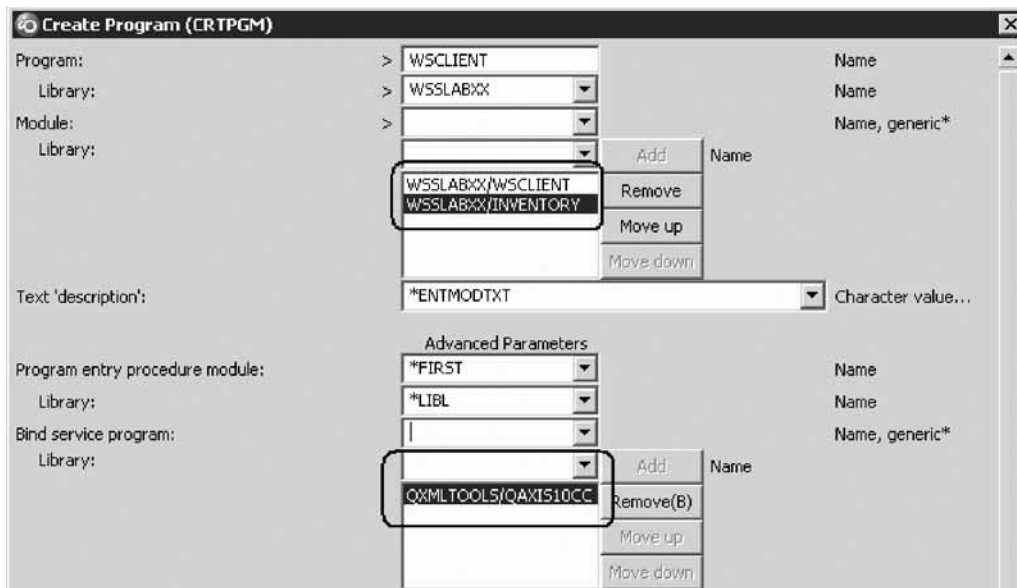


Now all that is left to do is to create a program from these two modules. Then you are ready to test the Web service invocation.

15. Select the WSCLIENT module that you just created. Right-click and select **Create > Program** on the pop-up menu.



16. In the Create Program dialog, add the WSSLABxx/Inventory module to the module list. Also add the Service program QXMLTOOLS/QAXIS10CC to the Bind Service program parameter. Select the **All Parameters** check box to see this parameter.



17. Click **OK** to create the program.
Check the iSeries Commands log for a successful compile message.



Now all that is left to do is to run the program you just created to invoke the Web service.

Lesson checkpoint

You learned the following:

- How to edit the RPG program source and create the program

Running the program

In this lesson, you learn how to run the program that you've configured and created.

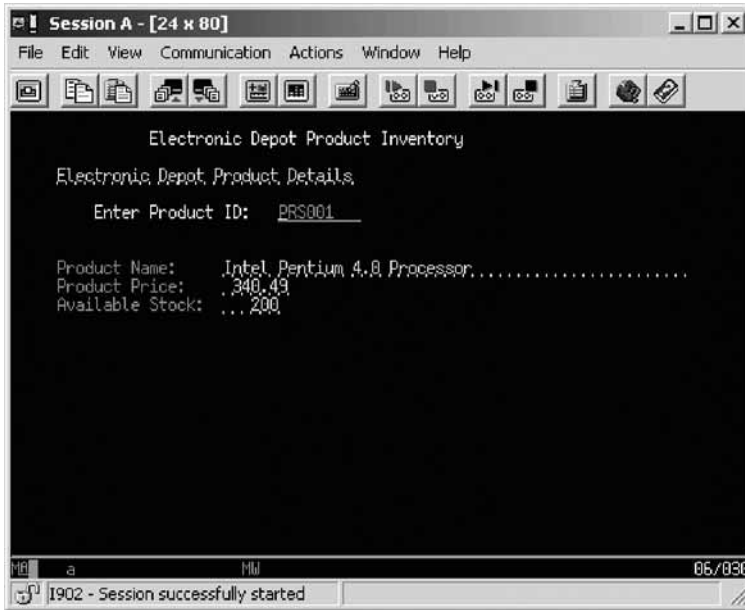
Note: The IntelWSEAR.ear file is for the Web service application that you need to install onto a WebSphere Application Server. This EAR file contains the server application that will access the RPG client that you just created. You will need the http port number that this WebSphere Application Server instance is using to allow the RPG client program to access this Web service.

To run the program:

1. Start a 5250 emulation session and sign-on to that system with the provided user ID and password. Make sure WSSLABxx library is in your library list.
2. Run the program by typing this command: Call WSCLIENT
3. Enter a product ID value such as PRS001, PRS002 or PRS003.



The information is returned back from the Web service.



You have successfully created an RPG program that can invoke a Web service. From a given WSDL file you have created the C functions to that interface with the Web service using the AXIS client on iSeries.

Lesson checkpoint

You learned the following:

- How to run the program that you configured and created

Module summary

In this module, you learned how to call a Web service from an RPG program. You have successfully created an RPG program that can invoke a Web service. From a given WSDL file you have created the C functions to that interface with the Web service using the AXIS client on iSeries. The AXIS client is part of the XML Toolkit 5733-XT1 in iSeries. Keep in mind that this is just the first iteration of supporting C functions to allow RPG and COBOL programs to directly invoke a Web service using the AXIS client. As Web services become more pervasive, the capabilities of the generated C functions are expected to increase over time.

Lessons learned

By completing this module, you learned about the following concepts and tasks:

- Set up an environment variable to a directory in the XML Toolkit
- Run the XML Toolkit extraction utility named QSHELL
- Add a prototype to describe the program interface
- Add a call command to invoke the program
- Run the program

Additional resources

For a comparison of C and RPG prototypes use this document: <http://www.opensource400.org/callc.html>.

Summary

In this tutorial, you learned how to develop a Web service using the Web Service wizard. You took a customer inquiry application written in RPG and wrapped it as a Web service. You used the Web Service wizard to build the Web service, setting all the properties and letting the wizard generate the code for you. You then learned how to package your application and Web service in an enterprise archive so that you can deploy it on a machine running Application Server.

Next you learned how to install the EAR containing the Web service and the implementation code onto a runtime instance of the WebSphere Application Server. Then you learned how to test the deployed Web service.

Finally you learned how to take an existing Web service and modify an existing RPG program to call this Web service.

Lessons learned

By completing this tutorial, you learned about the following concepts and tasks:

- Create an iSeries Web service using the Web Service wizard
- Test an iSeries Web service on a local test client
- Deploy the iSeries Web service to WebSphere Application Server for iSeries
- Test the deployed iSeries Web service
- Call a Web service from an RPG program

Additional resources

For more information on the product and the iSeries Web Tools, see ibm.com/software/awdtools/iseriess.

Notices

© Copyright IBM Corporation 1992, 2007. All rights reserved.

U.S. Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this documentation in other 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 documentation. The furnishing of this documentation 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.*

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

*IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan*

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 OR CONDITIONS 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 the 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.

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:

*Intellectual Property Dept. for WebSphere Software
IBM Corporation
3600 Steeles Ave. East
Markham, Ontario
Canada L3R 9Z7*

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 documentation 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 measurements 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.

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.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. 1992, 2007. All rights reserved

Trademarks and service marks

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

- IBM
- i5/OS
- iSeries
- Rational®
- System i
- WebSphere

Intel® and Pentium® are trademarks of Intel Corporation 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.

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

UNIX® is a registered trademark of The Open Group in the United States and other countries.

Linux® is a registered trademark of Linus Torvalds in the United States, other countries, or both.

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