IBM WebSphere Business Connection

Web Services Advanced Topics

Version 1.1.0

Note!
Before using this information and the product it supports, be sure to read the general information under "Notices" on page 21.

First Edition (September 2002)

This edition applies to Version 1, Release 1, Modification 0, 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

Advanced Topics

In the CrossWorlds(R) Sample document, you saw an example of how one IBM(R) CrossWorlds collaboration, using the SOAP connector, called a Web service that was enabled as another CrossWorlds collaboration. In this document, you will see how to install and use a Business Connection sample in which the SOAP Connector is used to invoke a Java(TM)-based, non-CrossWorlds Web service.

Because Java-based, non-CrossWorlds services are typically based on the remote procedure call (RPC) model of Web services, this sample is referred to as the CrossWorlds-to-RPC sample.

CrossWorlds-to-RPC sample

Some Web service and SOAP messaging concepts will be discussed, in this section, to give you an understanding of how the CrossWorlds-to-RPC sample works and how it differs from the CrossWorlds-to-CrossWorlds proxy service sample.

Full discussion of Web service and related technologies is beyond the scope of this document. For more depth, you should consult outside references that are available on the Web and in reference books. A good starting point on the Web is the **IBM DeveloperWorks(TM) Web Services Zone** found at http://www-106.ibm.com/developerworks/webservices/.

Calling general-purpose Java Web services

The sample you have worked with so far has dealt with a symmetric situation where a CrossWorlds-supplied service is called using the CrossWorlds SOAP Connector.

Figure: CrossWorlds artifacts are on both enterprises

In general, it is not possible to use the CrossWorlds SOAP connector to send a message to *any* Web service available on the Internet. There are limitations on the ability of the SOAP connector to form messages from business objects. These limitations are not apparent in the CrossWorlds-to-CrossWorlds case because CrossWorlds code handles both ends of the conversion needed: business-object-to-SOAP conversion at the sender, and SOAP-to-business-object conversion at the receiver.

However, it is useful to be able to call services that you have developed with tooling such as WebSphere(R) Application Developer. If you are the developer of these services, you can control their interfaces and modify their deployment in a way that will permit the SOAP connector to invoke them. Private Web services such as these allow you to connect from your collaborations to code that might not be reachable from the collaboration otherwise. For example, if you have an EJB whose services you want to use from within a collaboration, an option is to wrap the EJB services with a Web service and use the SOAP connector to call the service.

SOAP message styles

One of the differences between the CrossWorlds-to-CrossWorlds Web-service model and the general-purpose Web-service scenario is the style of SOAP messaging that is employed. There are two styles of SOAP messages:

- CrossWorlds-to-CrossWorlds uses the *document style* form of SOAP messaging.

  SOAP document-style messages are well suited to typical business-to-business interactions. A request message is sent as a single document contained in the SOAP message envelope. The receiver of the message is expected to know how to process the document and provide an appropriate response, possibly asynchronously.

  A SOAP-enabled application server receives document-style requests via the messagerouter URL.

- Most generally available Web services (including Web services created using WebSphere Application Developer version 4.1) use the RPC style of SOAP messaging.

  SOAP RPC-style messages mimic a method call in that message parts corresponding to method parameters are present in the SOAP message body. The receiver of the message expects a certain number, type, and ordering of message parts, just as a method call requires a certain parameter list when it is called. A synchronous response is provided by the service, and there may only be one message part in the response envelope.

  A SOAP-enabled application server receives RPC-style requests via the rpcrouter URL.

  Figure: SOAP message going to messagerouter (document-style) or rpcrouter (RPC-style)

A SOAP connector example

For direct CrossWorlds-to-CrossWorlds connectivity, the SOAP connector creates a document-style SOAP message. This message is sent to the messagerouter URL of the application server that hosts the CrossWorlds proxy class. An example document-style message sent from the SOAP connector is shown below:

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<SOAP-ENV:Header>
<hns0:affiliate xmlns:hns0="http://www.ibm.com/wbc">
<DestIDIdentifier>stevesh</DestIDIdentifier>
<DestIDType>PartnerName</DestIDType>
</hns0:affiliate>
</SOAP-ENV:Header>
<SOAP-ENV:Body>
<ns1:m_BCT_DocStyleTest xmlns:ns1="urn:ibmwsgw#BCT_DocStyleTest_Retrieve">
<child>  <field1>true</field1>
<field2>1</field2>
<field3>1.1</field3>
<field4>1E2</field4>
<field5>string1</field5>
<field6>2002-05-14T19:26:00Z</field6>
<field7>long text 1</field7>
</child>
</ns1:m_BCT_DocStyleTest>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Notice the bold part of the message. This is where the CrossWorlds data handler places the data fields from the generic business object in the SOAP message. If you look at the data for the fields, you have some sense of their data type, but there is nothing to tell the service that handles the message anything about the data types.

This message will be read by the messagerouter servlet, and the contents of the SOAP-Env:Body will be passed to the service whose ID is named in the Body namespace of the message (urn:ibmwsgw#BCT_DocStyleTest_Retrieve). The method specified by the Body name of the message Body will be called for this service ID (that is, method name m_BCT_DocStyleTest will be called).

For document-style services, the service methods all have the same parameter list. One of the parameters is the SOAP Envelope. The messagerouter creates an instance of org.apache.soap.Envelope using the message contents. It passes the Envelope to the service method, which must include code to parse it appropriately and process the contents.

RPC message semantics

Recall that the RPC model mimics a method call using a SOAP message. The message encapsulates the parameters for the call in its Body. The order and type of the parameters must match the parameter list of the service Java method to invoke.

RPC messages are sent to the **rpcrouter** URL of the SOAP application server rather than the **messagerouter** URL. The **rpcrouter** takes the contents of the SOAP Envelope and creates an instance of a Java type for each message part. These Java objects are used as parameters to invoke a method on a Java class. Remember that the method name is the SOAP message Body part name, and the class name is found by the ID of the service, which is the SOAP message Body namespace.

For the **rpcrouter** to know the proper types of the parameters, the RPC-style message has to include type information along with each part in the message.

Using the SOAP Connector to send RPC-style messages

The CrossWorlds SOAP connector can be configured via the meta-objects it uses to send messages that use RPC-style semantics. This allows you to call Web services that expect RPC-style SOAP messages.

To configure the use of RPC-style semantics, you set the value of the TypeInfo field in the BOtoSOAP client meta-object for your SOAP request application-specific business object to the value **true** as shown below:

Figure: The attributes tab showing the TypeInfo field set to true

The resulting SOAP message issued by the SOAP connector now has RPC semantics, as you can see by the type information in each field in the following example message:

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<SOAP-ENV:Header>
<hns0:affiliate xmlns:hns0="http://www.ibm.com/wbc">
<DestIDIdentifier>stevesh</DestIDIdentifier>
<DestIDType>PartnerName</DestIDType>
</hns0:affiliate>
</SOAP-ENV:Header>
<SOAP-ENV:Body>
```

```
<ns1:m_BCT_RPCStyleTest xmlns:ns1="urn:ibmwsgw#BCT_RPCStyleTest_Retrieve">
<child xsi:type="ns1:BCT_RPCStyleTest" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
 <field1 xsi:type="xsd:boolean">true</field1>
<field2 xsi:type="xsd:int">1</field2>
<field3 xsi:type="xsd:float">1.1</field3>
<field4 xsi:type="xsd:double">1E2</field4>
<field5 xsi:type="xsd:string">string1</field5>
<field6 xsi:type="xsd:dateTime">2002-05-14T19:26:00Z</field6>
<field7 xsi:type="xsd:string">long text 1</field7>
</child>
</ns1:m_BCT_RPCStyleTest>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The bold part of the message text has type information that is used by the rpcrouter when it builds the parameter list.

Web Services Gateway messages

SOAP messages that are processed by the Web Services Gateway must use RPC semantics. The channels used by the Business Connection offering are based on the SOAP rpcrouter servlet, so for the Web Services Gateway to correctly process messages sent to a SOAP channel, the messages must include type information.

The CWGenUtility provided by the Business Connection offering to produce the CrossWorlds artifacts for a Web service produces meta-object definitions with the TypeInfo field set to **true** where it is appropriate. This means that the SOAP connector will produce messages that can be processed by the Web Services Gateway channels.

SOAP connector limitations

For an RPC-style Web service to be called by the SOAP Connector, these limitations must be accommodated:

- Only a single namespace can be used by the body of the message and all of its parts.
- Only some data types may be present in the Web-service interface. These include string, integer, float, double, and the other simple types supported by the CrossWorlds Business Object Designer. Commonly used Java classes such as Vector and Hashtable, however, may not be used in the interface to a Web service if it is to be called from the SOAP Connector.
- Complex types containing zero or more attributes of a simple type (as mentioned above) and zero or more attributes of a complex type are allowed.
- Arrays must be composed of complex types, not simple types.

These limitations are mainly derived from the fact that business objects are used to form the message that is sent by the SOAP connector. Any data in the message must have a CrossWorlds data type. Because CrossWorlds arrays (N-cardinality children) must be composed of business objects and not of simple types, any arrays in the Web-service interface must be arrays of complex types.

The RPC Sample Scenario

An example of a Java Web service developed using WebSphere Application Developer that is called from a collaboration via the SOAP connector is provided with the Business Connection offering. The theme of this sample is an expanded version of the common Stock Quote service.

The sample allows you to enter a list of one or more stock symbols as an array of business objects. This list is sent as a SOAP message via the SOAP connector to the rpcrouter URL of a SOAP-enabled application server deployed in WebSphere. The Web service that processes the request, which was created with WebSphere Application Developer, is deployed in the sample application server. The rpcrouter servlet forms a method call from the message and calls the service class. If the server has connectivity to http://www.xmltoday.com, a quote for each symbol in the request list will be obtained. If there is no connectivity, a list is returned with error values (-1.0) in the quote response fields. The response list is sent back synchronously to the caller.

Installing the RPC service in WebSphere

The RPC Stockquote service is provided in a SOAP-enabled EAR file.

To install the RPC service:

1. Bring up your WebSphere Admin Console.
2. Install bctwssamplesrpc.ear into an application server of your choice (the default server, for example). Do not install the EAR file into the Web Services Gateway App Server.
3. Regenerate the Web server plug-in, stop the application server (if it is running), and then restart (or start) the application server where you installed the EAR file.

Reviewing the Java code for the Web service

The source code for the classes used by the sample service is included at the end of this document. Look at the following classes that are used by the service:

- com.ibm.bct.ws.samples.rpc.QuoteRequest :

  Encapsulates a String field named **symbol** to carry a request for a single quote.

- com.ibm.bct.ws.samples.rpc.QuoteResponse :

  Encapsulates two String fields named **symbol** and **quote** to carry a single quotation back to the caller.

- com.ibm.bct.ws.samples.rpc.RequestList :

  Encapsulates a String field named **date** and an array of **QuoteRequest** objects. This is the request type for the **StockQuote getCompanyQuotes(RequestList)** method.

- com.ibm.bct.ws.samples.rpc.ResponseList :

  Encapsulates a String field named **date** and an array of **QuoteResponse** objects. This the return type for the **getCompanyQuotes(RequestList)** method.

- com.ibm.bct.ws.samples.rpc.Stockquote :

  This is the class which provides the **getCompanyQuotes(RequestList)** method. This is the Web service implementation.

Installing the sample CrossWorlds code

To install the code:

1. Refer to the CrossWorlds Samples section of the Web Services Overview and Samples and import the Business Connection sample CrossWorlds data if you have not already done so.
2. Before running the sample, you will need to compile the collaboration template named **StockQuoteServiceOutbound**.
3. After you have done this, stop and restart the ICS.
4. Check the system view to be sure that the collaboration object named **BCT_SampleConnector1_to_BCT_SampleSOAPConnector _StockQuoteServiceOutbound** has started.
5. Also make sure the **BCT_SampleSOAPConnector** has started. You can check that the connector controller is started by viewing the System View window. You also need to start the connector agent for this connector. Refer to the Web Services Overview and Samples document to review how this is done.
6. Look at the collaboration template and the collaboration object. They are similar to the previous **BCT_TestAllTypesOutbound** sample.

Because this sample calls out from CrossWorlds to a Java RPC Web service, there is no inbound collaboration.

Reviewing the business objects

To review the business object:

1. Open the **StockquoteService** folder in the CrossWorlds **BCT_WS_Samples** project.

2. Look at the business object definitions for this service.

   In addition to the SOAP meta-objects for sending a message via the SOAP connector, there are also some business objects that have names resembling the four interface classes used by the StockQuote RPC Web service.

   If you open these business objects, you can see that they have attributes that correspond to the fields of the Java classes. In particular, notice that the arrays of the Java classes have corresponding arrays (cardinality N) fields in the CrossWorlds business objects.

   Figure: Expansion of the Business Objects for Stockquote Service

   Recall that one of the limitations of the SOAP Connector is that it cannot send messages with arrays of simple types. Because of that limitation, the **symbol** within the **QuoteRequest** was wrapped so that it could be passed as an array of symbols in the message. An array of Strings cannot be represented as a business-object attribute, but an array of **com.ibm.bct.ws.samples.QuoteRequest** can.

3. Open the BO definition for **MO_Client_BOtoSOAP_BCT_SOAP_StockQuoteService_getCompanyQuotes _Request_Retrieve**.

   Figure: Attributes tab highlighting the BodyNS entry

   Notice the **BodyNS** value. This is the single namespace that will be used in the SOAP message. This is important to note, because when WebSphere Application Developer is used to produce a Web service, it uses multiple

namespaces in the service deployment descriptor and WSDL that it generates. For this sample, it used two namespaces, and only one can be used.

In the sample, the namespace used is (http://tempuri.org/com.ibm.bct.ws.samples.rpc.Stockquote). The namespace is placed into the meta-object definition and has to be changed on the service side as well. This is done by changing the deployment descriptor for the service.

Reviewing the deployment descriptor file

Look at dds.xml under the Web application directory for the deployed Web service (look under *<was_root>*\installedapps\...) You see that there is a comment at the top of this file describing what was done to the deployment descriptor that WebSphere Application Developer originally generated. Below are excerpts from the dds.xml file showing what was done.

```
<isd:service xmlns:isd="http://xml.apache.org/xml-soap/deployment"
    id="http://tempuri.org/com.ibm.bct.ws.samples.rpc.Stockquote"
    I figucheckMustUnderstands="false">
 .
 .
 .
 <!-- Changed the namespace xnlns:x="http://www.stockquote.com
   /schemas/StockquoteRemoteInterface" -->
 <!-- to be the same as the above "id" value in the following 4 lines
 --> . .
```

See the actual dds.xml file for all of the changed lines.

This provides the service with only a single namespace, which is what is required if it is to be invoked from the SOAP connector.

Running the RPC sample

Be sure that the WebSphere application server that hosts the RPC service is started. Also the BCTSampleSOAPConnector and its agent process, BCTSampleConnector1, and the RPC collaboration object must all be running.

To run the sample service:

1. Bring up one Test Connector window.
2. Open the **BCTSampleConnector1** profile.
3. Connect to the agent process.
4. Load the saved business object named **BCT_SOAP_StockquoteService_getCompanyQuotes1.**
5. Examine the business object:

   Figure: Picture of business object with two requests entered

   It is preloaded with two requests, one for symbol **IBM** and one for **GE.** The date is pre-filled, but this will be overwritten by the collaboration just before sending the request to the SOAP connector. The URL is pre-filled with the URL of the RPC service of the SOAP server that you deployed for this sample.

6. If you want, you can change the symbols or add more symbols by right-clicking the **requestQuoteList**.
7. Click **OK**.
8. Highlight the preloaded business object definition and send it.
   You will see activity in the agent window, followed by a short pause, followed by activity as the response comes back.

9. Accept the response in the Test Connector window.
   The response will appear in the right-hand pane of the Test Connector window.
10. Double-click the response to examine it.
11. Expand the Response to see the details:

    Figure: The business object with the results of the two requests (the stock prices)

Compare the values to the earlier ones (shown in step 5). Notice that:

- The date has been updated. This was done by the calling collaboration before the SOAP message was sent to the Web service.
- The quote values have been updated by the Web service.

This document described how you can use the SOAP connector to send RPC messages to a non-CrossWorlds Web service. To learn more about CrossWorlds-to-RPC flows, including how to develop them, see the Web Services Technical Reference.

Routing filters and audit logs

This section explains concepts that you can apply to enhance the Business Connection offering. After reading the document, you can update the samples shipped with Business Connection to apply the concepts.

Using a routing filter

Recall that a Web Services Gateway service can have more than one target service associated with it. To uniquely identify each target service, the Gateway Administrator assigns a string value, which can be a trading partner ID, to each target service when more than one is associated with a gateway service.

The following screen shows how a target service with an identifier of **TradingPartner-1068** is added to a gateway service. You get to this screen by listing the gateway services and clicking the link for one of them.

Figure: Target Services screen with fields filled out for a new target

After adding two more target services to the gateway service, the gateway service details screen shows the following:

Figure: Target Services screen after new target was added

To use routing in the Web Services Gateway, the message coming to the outbound channel intended for a particular trading partner must carry the destination trading partner ID in it to specify the correct target service to invoke. The Business Connection offering uses the SOAP message header to carry the target trading partner ID.

The CrossWorlds collaboration or a mapping can be used to fill in a SOAP header business object with the destination trading partner ID before the message is sent out through the SOAP connector. Business object application-specific data is used to specify that data carried by the generic business object in a child business object will be placed in the header of the SOAP message. The development procedures describe how you do this.

The sample outbound collaboration object uses the SOAP header. Look at the BCT_SOAP_BCT_TestAllTypes_Wrapper business object definition. This is the type that is actually used for the request by the SOAP connector. It includes a child business object named child that is of the type used by the generic business object BCT_TestAllTypes. The map from the collaboration **To** port to the BCTSampleSOAPConnector fills in this child business object. The other child of the business object is named requestHeader. It has a child also, named affiliate. These names are arbitrary, but the application-specific data for these two attributes determines that the SOAP data handler will place the affiliate fields into the header of the SOAP message rather than the body.

For your information, the requestHeader attribute has application-specific data of:

```
soap_location=SOAPHeader;type=BCT_SOAP_BCT_TestAllTypes_HDR
```

The affiliate attribute has application-specific data of:

```
elem_ns=SOAPHeader;type=BCT_SOAP_BCT_TestAllTypes_HDR
```

Refer to the CrossWorlds Guide to using Web Services for detailed discussion of the header usage. For the purposes of this document, it is enough to understand that the affiliate fields are placed into the header for use by the Web Services Gateway.

Figure: The Attributes tab for requestHeader

The figure below shows the flow of a SOAP message that is routed by the Web Services Gateway based on the SOAP header Destination ID:

Figure: Flow of a SOAP message with header through the routing filter being delivered to three targets

The Web Services Gateway allows Request and Response Filters to be deployed for each gateway service. For the specific purpose of routing in this sample, the Routing Filter used by WebSphere Business Connection has been written to extract the destination trading partner ID value from the SOAP message header and is deployed in the Request Filter.

This filter uses a Gateway API to select the correct target service to receive the call.

Note that the Routing Filter is provided in bctwswsgwroutingfiltersoap.ear. If you deploy it and then configure a gateway service to use it as a Request Filter, it will read the SOAP header, look for the affiliate element, and then read the name in the DestIDIdentifier field. It will use this value to select the target service that has that identifier value.

Deploying and configuring a filter

Here are the steps to manually deploy and configure this filter. Note that when you install WebSphere Business Connection using the instructions in the Installation and Configuration Guide, this filter is automatically deployed and configured for your use.

1. Display the Web Services Gateway Admin screen and select **Filters**.
2. Click **Deploy**.
3. In the Filter name field, type:
   ```
   RoutingFilter
   ```
4. In the Home location field, type:
   ```
   BCTWSRoutingFilterSOAP
   ```
5. Click **OK**.

<o> <o> After deployment, the filter will be available as a choice on the gateway service screen for either a Request Filter or a Response Filter. If you want to use routing of gateway service requests to one of many target requests, you must add the Routing Filter as a Request Filter for the gateway service.

For the sample, this is done as shown below in the screen below. Note that you must highlight the Routing Filter selection before pressing the **add** button.

Figure: The Request Filters and Response Filters with add and remove buttons shown

After adding it, you can remove it by using the same screen and pressing the **remove** button.

Using the Business Connection Solution Manager Audit Log

This section discusses how you can log incoming requests and responses for a Web service.

If you check the **Audit Policy** check box labeled **Log requests for this service** for a gateway service, the Business Connection offering can use its Solution Manager component to save a copy of the incoming requests and responses for the service. The Solution Manager Audit Log is used for this purpose. A message queue is used to send messages from the Web Services Gateway machine to the Solution Manager Audit Log.

Figure: Gateway Service Properties screen with Log requests to this service checked

Note that there are two other logs (the Business Log and the Exception Log) maintained by the Solution Manager component. They are not used by the Web Services Gateway component, so they are not discussed in this document.

For more information on using the Logging Client of the Solution Manager, refer to **Using Business Connection APIs**.

Source Code for the RPC Sample Service

The source code for the classes that comprise the RPC sample service is listed below for your reference.

QuoteRequest.java

```
package com.ibm.bct.ws.samples.rpc;
 public class QuoteRequest
 {
         private String symbol;
         /**
          * Gets the symbol
          * @return Returns a String
          */
         public String getSymbol() {
                 return symbol;
         }
         /**
          * Sets the symbol
          * @param symbol The symbol to set
          */
         public void setSymbol(String symbol)
         {
                 this.symbol = symbol;
         }

 }
```

QuoteResponse.java

```
package com.ibm.bct.ws.samples.rpc;
 public class QuoteResponse
 {
         private float quote;
         private String symbol;

         /**
          * Gets the quote
          * @return Returns a float
          */
         public float getQuote() {
                 return quote;
         }
         /**
          * Sets the quote
          * @param quote The quote to set
          */
         public void setQuote(float quote)
         {
                 this.quote = quote;
         }


         /**
          * Gets the symbol
          * @return Returns a String
          */
         public String getSymbol() {
                 return symbol;
         }
         /**
```

```
         * Sets the symbol
         * @param symbol The symbol to set
         */
        public void setSymbol(String symbol)
        {
                this.symbol = symbol;
        }

 }
```

RequestList.java

```
package com.ibm.bct.ws.samples.rpc;
 public class RequestList
 {
        private String date;
        private QuoteRequest[] requestQuoteList =
  new QuoteRequest[0];

        /**
         * Gets the date
         * @return Returns a String
         */
        public String getDate() {
                return date;
        }
        /**
         * Sets the date
         * @param date The date to set
         */
        public void setDate(String date)
        {
                this.date = date;
        }


        /**
         * Gets the requestQuoteList
         * @return Returns a QuoteRequest[]
         */
        public QuoteRequest[] getRequestQuoteList() {
                return requestQuoteList;
        }
        /**
         * Sets the requestQuoteList
         * @param requestQuoteList The requestQuoteList to set
         */
        public void setRequestQuoteList(QuoteRequest[]
  requestQuoteList)
        {
                this.requestQuoteList = requestQuoteList;
        }


 }
```

ResponseList.java

```
package com.ibm.bct.ws.samples.rpc;
 public class ResponseList
 {
        private String date;
        private QuoteResponse[] responseQuoteList = new
  QuoteResponse[0];
        /**
         * Gets the date
```

```
         * @return Returns a String
         */
        public String getDate() {
                return date;
        }
        /**
         * Sets the date
         * @param date The date to set
         */
        public void setDate(String date)
        {
                this.date = date;
        }


        /**
         * Gets the responseQuoteList
         * @return Returns a QuoteResponse[]
         */
        public QuoteResponse[] getResponseQuoteList() {
                return responseQuoteList;
        }
        /**
         * Sets the responseQuoteList
         * @param responseQuoteList The responseQuoteList to set
         */
        public void setResponseQuoteList(QuoteResponse[]
 responseQuoteList)
        {
                this.responseQuoteList = responseQuoteList;
        }


 }
```

Stockquote.java

```
package com.ibm.bct.ws.samples.rpc;

 import java.net.URL;
 import java.io.*;
 import org.w3c.dom.*;
 import org.xml.sax.*;
 import javax.xml.parsers.*;
 //import org.apache.soap.util.xml.*;

 import com.ibm.ras.*;
 import com.ibm.bct.ws.trace.*;


 /**
  * This class is used to test the ability to use complex
  * types and arrays of complex types for an rpc-style web
  * service that is to be called using the CrossWorlds
  * SOAPConnector. Method getCompanyQuotes(RequestList)
  * exercises this capability by involving complex types
  * in the interface, which include arrays of other complex
  * types.
  *
  */
 public class Stockquote
 {
        public static final float CANNOT_OBTAIN_QUOTE = -1.0f;
        public static final float SYMBOL_LOOKUP_FAILED = -2.0f;

        final static private String CLASSNAME =
```

```
    Stockquote.class.getName();

       private static final RASTraceLogger m_traceLogger =
               BCTWSTrace.createRASTraceLogger(
                   BCTWSTraceConstants.BCT_ORG,
                   BCTWSTraceConstants.BCT_PROD,
                   BCTWSTraceConstants.BCT_COMP,
                   Stockquote.class,
                   BCTWSTraceConstants.BCT_GROUPNAME);
                   //TraceConstants.m_strGroupName);
       private static final RASMessageLogger m_msgLogger =
               BCTWSTrace.createRASMessageLogger(
                   BCTWSTraceConstants.BCT_ORG,
                   BCTWSTraceConstants.BCT_PROD,
                   BCTWSTraceConstants.BCT_COMP,
                   Stockquote.class,
                   BCTWSTraceConstants.BCTMESSAGEFILE);

       private boolean islogging = m_traceLogger.isLogging;

       //------
       // Obtain quotations for a list of symbols passed
// in a RequestList. Respond with a list of quotations
// encapsulated in a ResponseList.
       // Demonstrate some WAS tracing while we are at it.
       //------
       public ResponseList getCompanyQuotes(RequestList reqList)
       {
               if (m_traceLogger.isLogging)
                   m_traceLogger.entry(
                           RASITraceEvent.TYPE_ENTRY_EXIT,
                           CLASSNAME,
                           "getCompanyQuotes",
                           new Object[]{});

               ResponseList respList = new ResponseList();
               respList.setDate(new java.util.Date().toString());

               QuoteRequest[] qreqlist =
   reqList.getRequestQuoteList();

               QuoteResponse [] qresplist = new
   QuoteResponse[qreqlist.length];

               for (int i=0; i<qreqlist.length; i++)
               {

                   // Prepare the response array
                   qresplist[i] = new QuoteResponse();
                   qresplist[i].setSymbol(qreqlist[i].getSymbol());

                   float quotation = SYMBOL_LOOKUP_FAILED;
                   try
                   {
                           quotation = getQuote(qreqlist[i].getSymbol());
                           m_traceLogger.trace(
                               RASITraceEvent.TYPE_API,
                               CLASSNAME,
                               "getCompanyQuotes",
                               "Obtained quote for symbol
      <" +qreqlist[i].getSymbol()+ "> :
   quotation = "+quotation);
                   }
                   catch (Throwable t)
                   {
                           quotation = CANNOT_OBTAIN_QUOTE;
                           m_traceLogger.trace(
```

```
                                RASITraceEvent.TYPE_API,
                                CLASSNAME,
                                "getCompanyQuotes",
                                "Caught exception looking up symbol
    <" +qreqlist[i].getSymbol()+ "> :
 setting quotation = "+quotation);
                    }
                    qresplist[i].setQuote(quotation);

                }

                // Put the response array into the returning ResponseList
                respList.setResponseQuoteList(qresplist);

                if (m_traceLogger.isLogging)
                            m_traceLogger.exit(
                                RASITraceEvent.TYPE_ENTRY_EXIT,
                                CLASSNAME,
                                "getCompanyQuotes",
                                new Object[]{});

                return respList;
        }

        //------
        // Obtain a single quotation for a symbol passed within a
        // wrapper class. Demonstrate some WAS tracing while we are at it.
        //------
        public QuoteResponse getCompanyQuote(QuoteRequest qr)
        {
                if (m_traceLogger.isLogging)
                    m_traceLogger.entry(
                            RASITraceEvent.TYPE_ENTRY_EXIT,
                            CLASSNAME,
                            "getCompanyQuote",
                            new Object[]{});

                QuoteResponse qresp = new QuoteResponse();

                qresp.setSymbol(qr.getSymbol());
                float quotation = SYMBOL_LOOKUP_FAILED;
                try
                {
                    quotation = getQuote(qr.getSymbol());
                    m_traceLogger.trace(
                            RASITraceEvent.TYPE_API,
                            CLASSNAME,
                            "getCompanyQuote",
                            "Obtained quote for symbol
<" +qr.getSymbol()+ "> :
quotation = "+quotation);
                }
                catch (Throwable t)
                {
                    quotation = CANNOT_OBTAIN_QUOTE;
                    m_traceLogger.trace(
                            RASITraceEvent.TYPE_API,
                            CLASSNAME,
                            "getCompanyQuotes",
                            "Caught exception looking up symbol
<" +qr.getSymbol()+ "> :
setting quotation = "+quotation);
                }

                qresp.setQuote(quotation);

                if (m_traceLogger.isLogging)
```

```
                            m_traceLogger.exit(
                                RASITraceEvent.TYPE_ENTRY_EXIT,
                                CLASSNAME,
                                "getCompanyQuote",
                                new Object[]{});

                return qresp;
        }

        //------
        // Obtain quotations for an array of symbols, returning an
// array of floats.
        //------
        public float[] getQuotes (String [] symbols)
        {
                System.out.println ("Stockquote:getQuotes(String[])
 entry point called");

                float[] retArray = new float[symbols.length];

                System.out.println ("Stockquote:getQuotes(String[])
 response: ");
                for (int i=0; i<symbols.length; i++)
                {
                    float quotation = SYMBOL_LOOKUP_FAILED;
                    try
                    {
                            quotation = getQuote(symbols[i]);
                    }
                    catch (Throwable t)
                    {
                            quotation = CANNOT_OBTAIN_QUOTE;
                    }

                    retArray[i] = quotation;

                    System.out.println("\t"+symbols[i]+": "+quotation);
                }


                System.out.println ("Stockquote:getQuotes(String[])
 normal exit");


                return retArray;
        }

        //------
        // Obtain a quotation from a public url service for a given
// symbol. If anything bad happens, just throw it back to
// the caller.
        //------
        public float getQuote(String symbol) throws Exception
        {
                System.out.println("Stockquote:getQuote(String)
 entry point called");


                DocumentBuilderFactory factory =
 DocumentBuilderFactory.newInstance();
                DocumentBuilder xdb = factory.newDocumentBuilder();


                // get a real (delayed by 20min) stockquote
                URL url =
                    new URL(
                            "http://www.xmltoday.com/examples/stockquote/
```

```
 getxmlquote.vep?s=" + symbol);

              System.out.println(
                  "Stockquote:getQuote(String) opening input stream
  with URL: " + url);

              InputStream is = url.openStream();

              System.out.println(
                  "Stockquote:getQuote(String) input stream has been
  obtained, parsing for price beginning");

              Document d = xdb.parse(is);
              Element e = d.getDocumentElement();
              NodeList nl = e.getElementsByTagName("price");
              e = (Element) nl.item(0);
              String quoteStr = e.getAttribute("value");
              try
              {
              System.out.println("Stockquote:getQuote(String) normal
  exit 1");
                  return Float.valueOf(quoteStr).floatValue();
              }
              catch (NumberFormatException e1)
              {
                  // Could it be an int?
                  try
                  {
                      System.out.println("Stockquote:getQuote(String)
   normal exit 2");
                          return Integer.valueOf(quoteStr).intValue()
   * 1.0F;
                  }
                  catch (NumberFormatException e2)
                  {
                      System.out.println("Stockquote:getQuote(String)
   error exit 3");
                          return -1.0F;
                  }

              }

       } // end of method getQuote


}
```

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.