



**Stock Quote Sample**





**Stock Quote Sample**

**Note**

Before using this information and the product it supports, read the information in "Notices" on page 45.

---

# Contents

<b>Chapter 1. Introduction</b> . . . . .	<b>1</b>	Building the request flow . . . . .	21
<b>Chapter 2. Overview</b> . . . . .	<b>3</b>	Building the response flow . . . . .	28
<b>Chapter 3. Mediation design</b> . . . . .	<b>5</b>	<b>Chapter 5. Test</b> . . . . .	<b>33</b>
<b>Chapter 4. Build it yourself</b> . . . . .	<b>9</b>	Installing the runtime components. . . . .	33
Importing resources . . . . .	9	Testing the mediation flow . . . . .	34
Creating the StockQuoteService interface . . . . .	10	Debugging the mediation flow . . . . .	36
Setting up a business object for temporary data . . . . .	12	Changing the quality of service at runtime . . . . .	38
Creating the mediation module. . . . .	13	<b>Chapter 6. Deploy</b> . . . . .	<b>43</b>
Assembling the mediation module. . . . .	15	<b>Notices</b> . . . . .	<b>45</b>
Implementing the mediation. . . . .	20	<b>Terms of use</b> . . . . .	<b>49</b>
Defining the end points of the mediation . . . . .	20		



---

## Chapter 1. Introduction

The Stock Quote sample demonstrates how you can easily integrate different services and manipulate messages at runtime without affecting the client applications. This integration is achieved through the Service Component Architecture and mediation functionality provided by the WebSphere Enterprise Service Bus. The StockQuote mediation service is built with the business integration tools for building mediations: the assembly editor and the mediation flow editor. Mediation flows intercept and modify messages that are passed between existing services (providers) and clients (requesters) that want to use those services.

This sample runs on WebSphere Enterprise Service Bus v 6.2 or WebSphere Process Server v 6.2.

If you want to learn more about the WebSphere Enterprise Service Bus, see the documentation in the information center for IBM WebSphere Business Process Management.



---

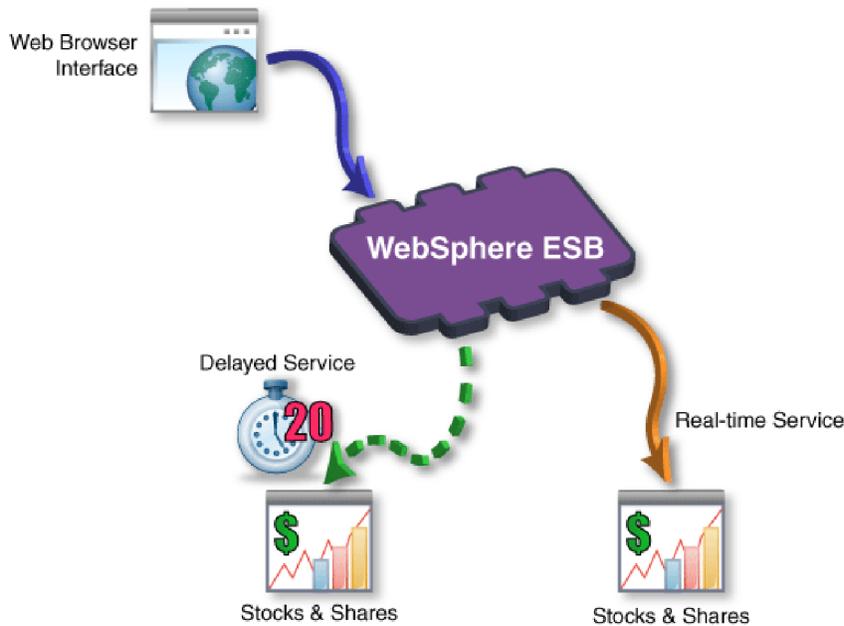
## Chapter 2. Overview

This Stock Quote sample addresses the business need of a financial services company that provides an interactive Web-based stock market service to its customers.

The company wants to differentiate itself from its competition by offering tiered levels of service. The company's goal is to offer delayed stock quotes to their *standard* customers and real-time quotes to their *premium* customers, that is, customers who pay a subscription.

The company wants to:

- Offer the delayed and real-time stock quote services as a single service, which dynamically determines which external service to invoke based on the customer's subscription level.
- If the real-time service is unavailable, route requests to the delayed service without affecting the running application.
- Log all requests to the service to satisfy audit requirements.



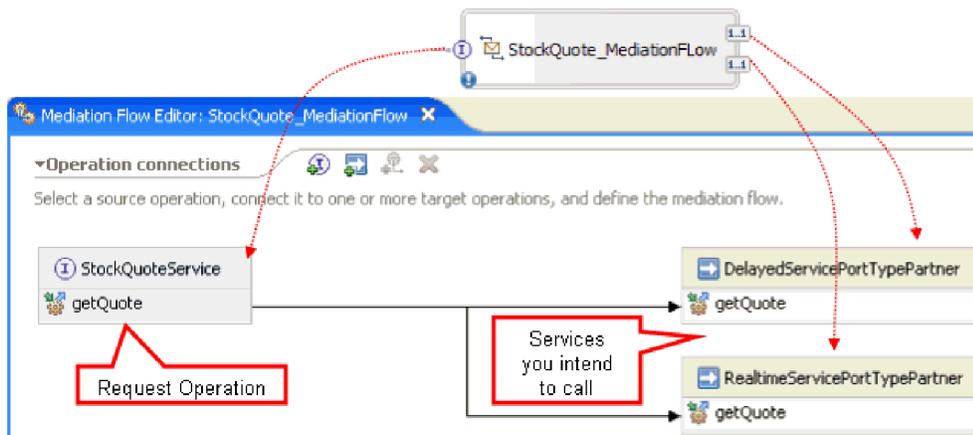
This sample should take approximately 60 minutes to build and run.



## Chapter 3. Mediation design

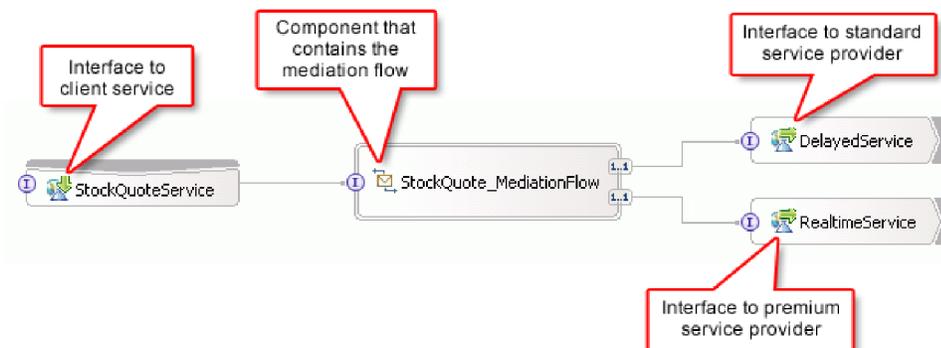
The mediation service that runs on the WebSphere ESB or WebSphere Process Service is contained in a single *mediation module* called StockQuote. The mediation module consists of: an *export* which provides an interface to enable the service to be called, *imports* that provide interfaces to the external web service providers, and a *mediation flow component* that defines the mediation implementation.

The mediation module, StockQuote, is built in the assembly editor, and the mediation flow component, StockQuote\_MediationFlow, is created in the mediation flow editor. The following figure shows the relationship between the interfaces and references in the assembly editor and the mediation flow editor.



### StockQuote mediation module

The following diagram shows the assembled StockQuote mediation module:



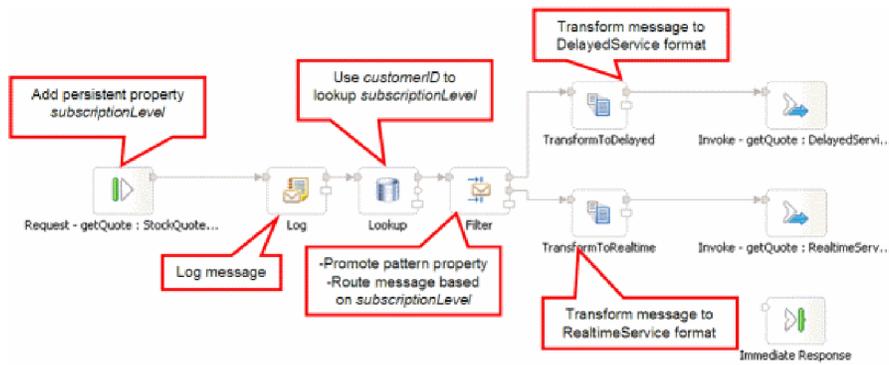
The StockQuote mediation module consists of the following elements:

- **StockQuoteService** has a WSDL interface, called StockQuoteService, and uses SOAP/JMS web service binding so that the servlet front end can connect to the mediation module by using JAX-RPC. In this sample, you will create the StockQuoteService interface and generate the WSDL file.
- **StockQuote\_MediationFlow** contains the mediation flow. In this sample, you will create and implement the StockQuote\_MediationFlow component.
- **RealtimeService** has a web service binding and an interface that matches the real-time (premium) service. In this sample, you will import the WSDL file RealtimeService.wsdl.

- **DelayedService** has a web service binding and an interface that matches the delayed (standard) service. In this sample, you will import the WSDL file DelayedService.wsdl.

### StockQuote\_MediationFlow

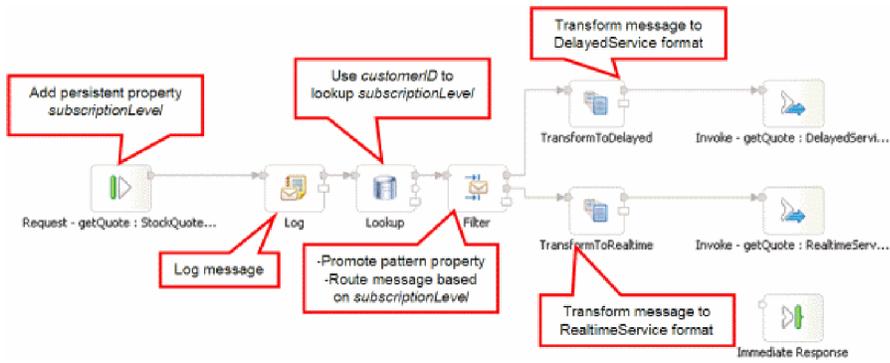
The diagram below shows the request flow that defines the mediation logic applied to the message as it flows through the StockQuote\_MediationFlow component to the target service providers.



The request flow is executed from left to right, in the following order:

1. The property `subscriptionLevel` is set in the correlation context of the message so that it will be available later in the response flow.
2. The request is logged using the Message Logger mediation primitive named `Log`.
3. A Database Lookup mediation primitive named `Lookup` uses the `customerID` element in the message body to determine whether the customer is entitled to the premium or standard service by looking this information up in the supplied `CustomerDatabase`. This information is added to the `subscriptionLevel` property in the correlation context of the message, for use later.
4. The request is routed by a Message Filter called `Filter`, based on the `subscriptionLevel` information in the correlation context, to either the real-time or delayed stock quote service. The Filter's pattern property is promoted so that it can be changed at runtime to redirect the stock quote request to the delayed service if the premium service is unavailable.
5. The message is transformed on the way to either service by XSLT primitives `TransformToDelayed` and `TransformToRealtime` so that it matches what the service expects.
6. The response from each service is passed through an XSLT mediation primitive (`DelayedToStockQuoteService` & `RealtimeToStockQuoteService`) to match the format that is required by `StockQuoteService`.

The diagram below shows the response flow that defines the mediation logic applied to the returning message as it flows through the `StockQuote_MediationFlow` component from the target service provider to the client. A Message Element Setter is used to copy the value of `subscriptionLevel` from the correlation context to the property `qualityOfService` in the message. The `qualityOfService` text indicates "Premium" to a response that is returned from the real-time service, and "Standard" to a response that is returned from the delayed service. The `qualityOfService` text is displayed in the client to indicate the service provider that was used.





---

## Chapter 4. Build it yourself

Build the sample, and then test it.

Start building the Stock Quote sample by creating the resources library and mediation module for your sample. Resources such as interfaces and business objects can be contained in libraries so that they can be easily shared by modules. Mediation modules contain mediation flow and Java components, exports to allow the target service in the mediation module to be called by other modules or clients, and imports which allow services external to the module to be invoked. The assembly diagram of the mediation module is used to wire the exports, components, and imports together to form the integrated service application.

Create and assemble the library and mediation module in the following order:

1. Create a resource library, and import the service providers' WSDL files into it.
2. Create an interface in the resources library. This interface allows the client to access the mediation flow component.
3. Create a mediation module.
4. Assemble the mediation module.
5. Implement this mediation flow component.

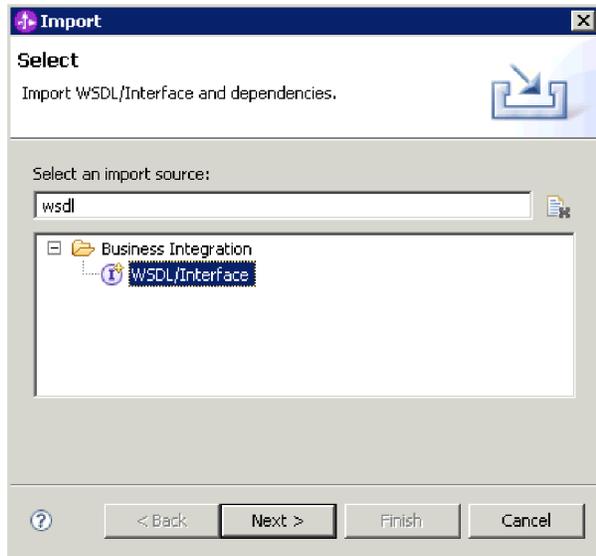
---

### Importing resources

Create a library, and import the ready-made WSDL files into it. These files are the web service interfaces that you will use to connect to the delayed and real-time services.

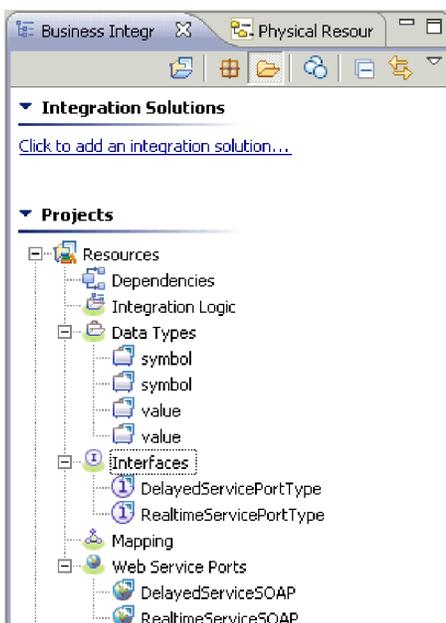
To create the **Resources** library and import the ready-made WSDL files into it, complete the following steps.

1. In the Business Integration view, under Projects select **Click to add a business integration project**. For the Business Integration Project, select **Create a library** and select **Next**. For the Library name specify Resources. Click **Finish**. If you already have Projects then right-click and select **New > Project > Library**. Name it Resources. Click **Finish**.
2. Right-click and select **Import**. In the Select an import source text box, type *wsdl* and select *WSDL/Interface*. Click **Next**.



3. In the **Import From** directory field, browse to the directory Install Shared Resources directory /SDPShared/plugins/com.ibm.wbit.samples.content\_6.2.0.qualifier/artifacts/stockquote/wsdl. Click **OK**.
4. In the Import wizard, select both WSDL files: **DelayedService.wsdl** and **RealtimeService.wsdl**.
5. The target module is **Resources**.
6. Click **Finish** to import the WSDL files.

The available port types and ports based on the imported WSDL files are created under the Interfaces and the Web Service Ports categories in the navigation tree respectively.



## Creating the StockQuoteService interface

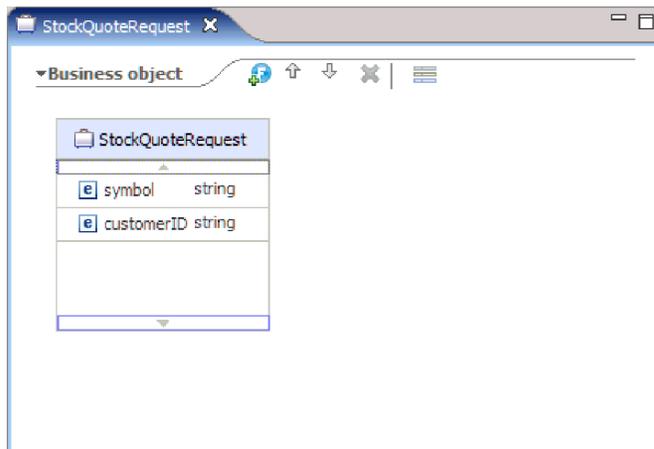
Create the StockQuoteService interface. This is the interface that you can use to connect to your web client to the mediation module.

The StockQuoteService interface will have a getQuote operation. The data that is sent and received by the getQuote operation will be contained in business objects. We will create the interface and business objects in the resources library, so that they can be used by other modules in the future.

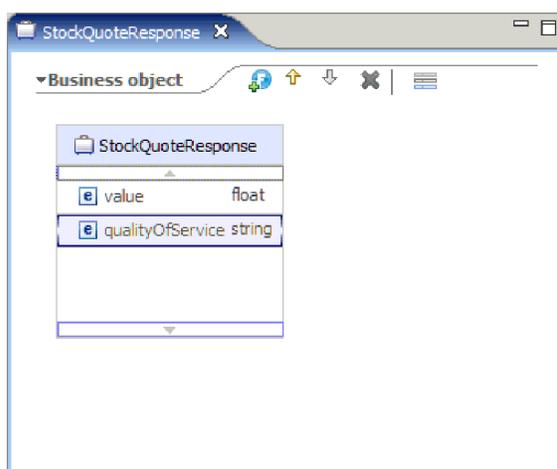
The getQuote operation will be used to send the request for a stock quote. The operation will send the request data as a business object named StockQuoteRequest that contains the fields symbol and customerID. The operation will receive the response data as business object named StockQuoteResponse that contains the fields value and qualityOfService.

Follow these instructions to first create the business objects, and then the service interface:

1. In the Business Integration view, select **Resources**, right-click and choose **New > Business Object**.
2. In the New Business Object wizard, type StockQuoteRequest in the **Name** field. Click **Finish**.
3. In the Business Object Editor that is now opened, click the **Add a field to a business object** button . A field named *field1* is created.
4. Rename field1 to symbol by clicking on the name to enter input mode.
5. Click the **Add a field to a business object** button  to add another field. Rename the field to customerID. Save the new business object.

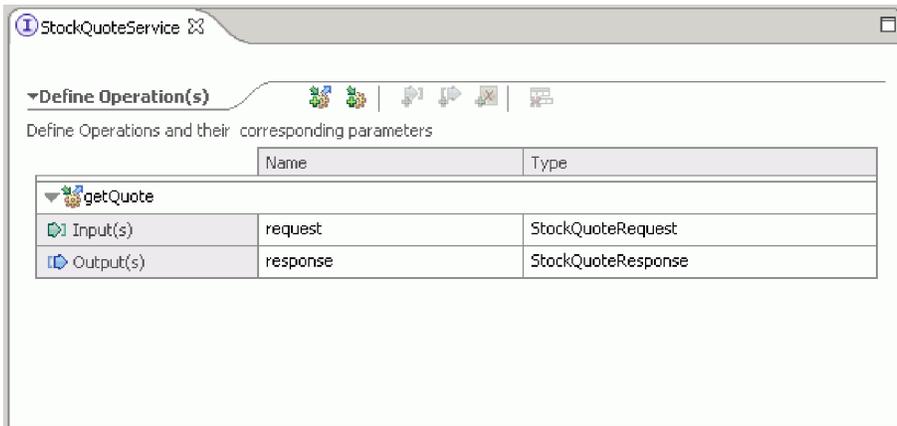


6. Right-click **Data Types**, and choose **New > Business Object** to create another business object.
7. In the New Business Object Wizard, type StockQuoteResponse. Click **Finish**.
8. In the Business Object Editor, click the **Add a field to a business object** button . An attribute named *field1* is created. Rename field1 to value. Click string and select float from the data type list.
9. Add another field. Rename the field to qualityOfService. Save the new business object.



10. In the Business Integration view, select **Resources**, right-click and choose **New > Interface**.
11. In the New Interface wizard, type StockQuoteService in the **Name** field. Click **Finish**.
12. In the Interface Editor that is now opened, click the **Add Request Response Operation** icon. An operation named *operation1* is created, with an input and an output. Rename operation1 to getQuote.
13. Rename input1 to request. Click input1, select it, and type request.
14. Click on the request input's type ("string") and in the resulting Data Type selection dialog, select **Browse**. Select **StockQuoteRequest** as the type and click **OK**.
15. Rename output1 to response. Click output1, select it, and type response.
16. Change the output's type to **StockQuoteResponse** . Save the interface.

This is what the interface that you created should look like in the interface editor.

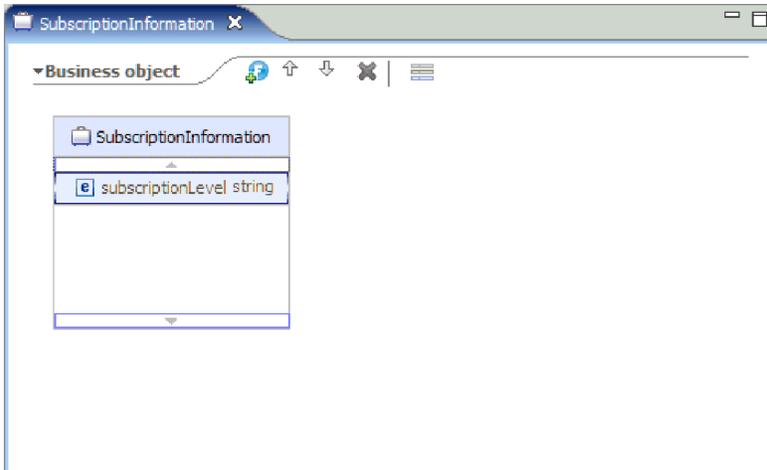


## Setting up a business object for temporary data

The request message is routed based on the value of the subscription level. This value needs to be passed with the message from one primitive to another. We will create a business object to contain the subscription level, and we will set the value of the subscription level when we build the request flow.

The SubscriptionInformation business object will contain a subscriptionLevel field. To create the business object, complete the following steps:

1. In the Business Integration view, select Resources. Right click and choose **New > Business Object**. Type SubscriptionInformation in the Name field, and click **Finish**.
2. In the Business Object Editor, click the **Add a field to a business object** button. An attribute named field1 of type string is created.
3. Rename field1 to subscriptionLevel and keep the type of string.



When you build your request flow, you will add this business object to the input node so that the `subscriptionLevel` property is available throughout the request and response flows. Close all open editors.

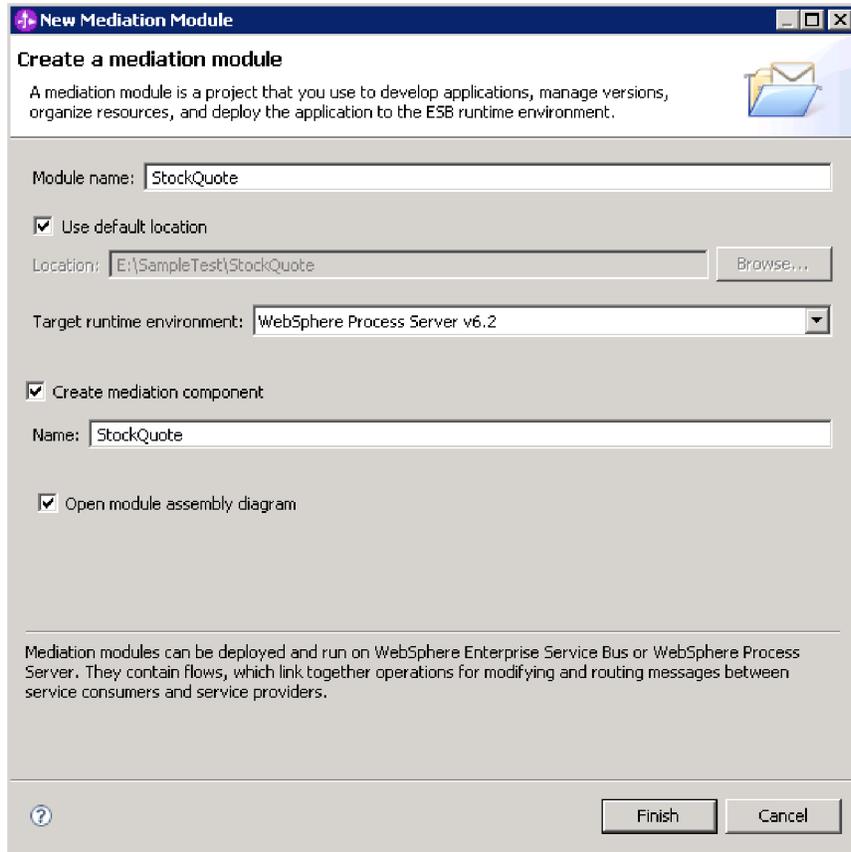
---

## Creating the mediation module

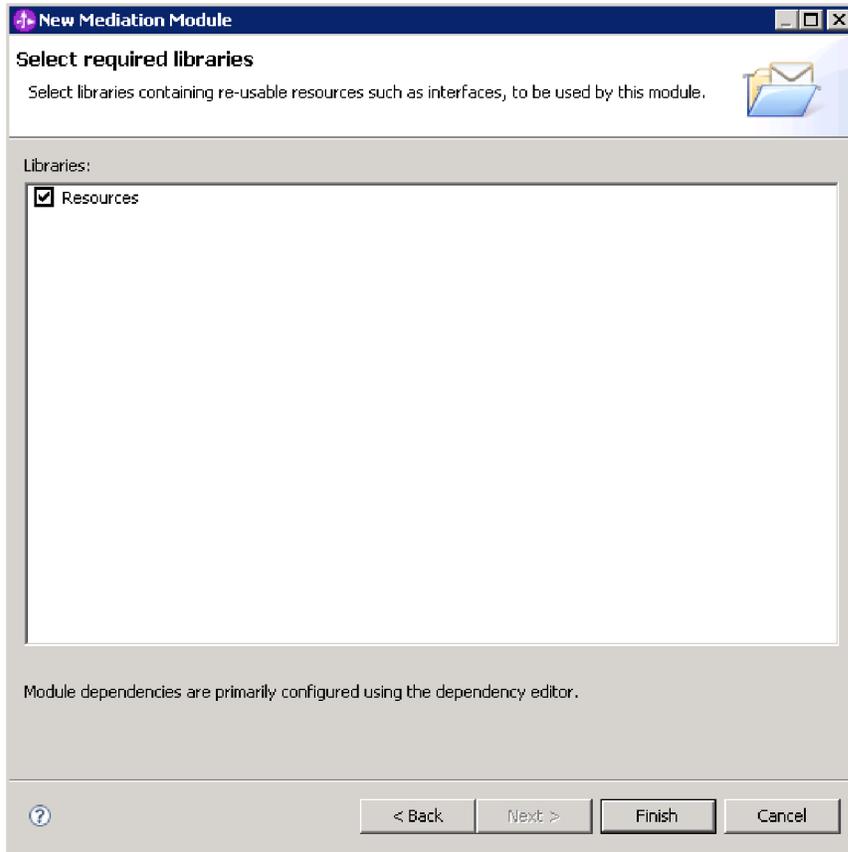
Create the mediation module named `StockQuote` that will contain your export, imports and mediation flow component.

To create the mediation module, complete the following steps:

1. In the Business Integration view, right-click to see the context menu and select **New > Mediation Module**. The New Mediation Module window opens.
2. In the **Module Name** field, type `StockQuote`.
3. Depending on which server profile is installed, keep the default target runtime WebSphere ESB Server v 6.2, or change it to WebSphere Process Server v 6.2.
4. Keep the **Create mediation component** box checked.



5. Click **Next**.
6. In the **Select Required Libraries** wizard, select the **Resources** library and click **Finish**. This will allow the artifacts in the library to be used by the mediation module.

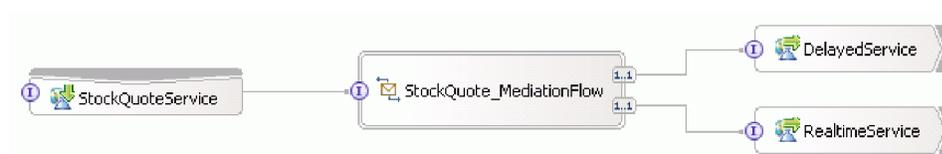


A mediation module called StockQuote is created, with a dependency on the Resources library. A mediation flow component called StockQuote is created in the module's assembly diagram.

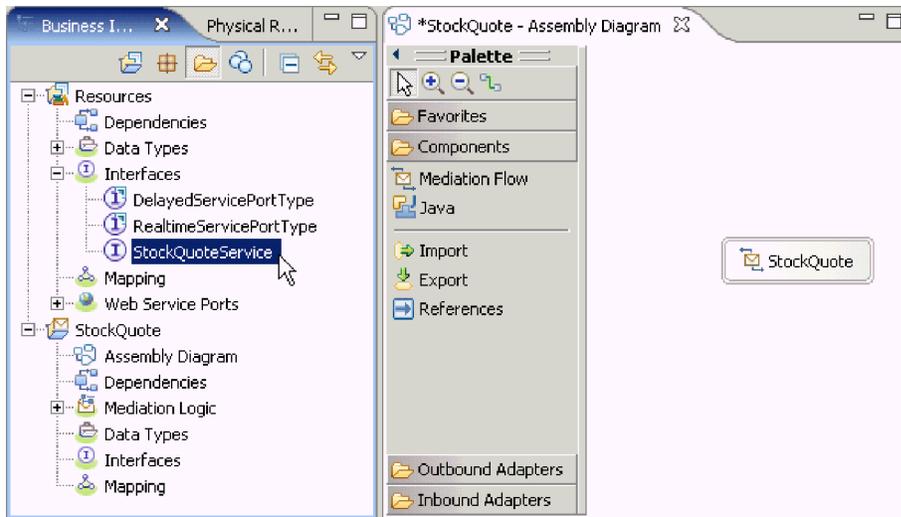
## Assembling the mediation module

Assemble the client and provider services of the StockQuote sample and wire them to the mediation flow component.

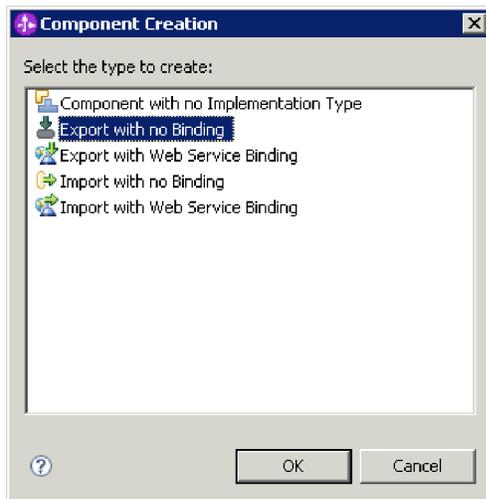
To build the StockQuote mediation module assembly diagram, as shown in the following diagram, complete the following steps:



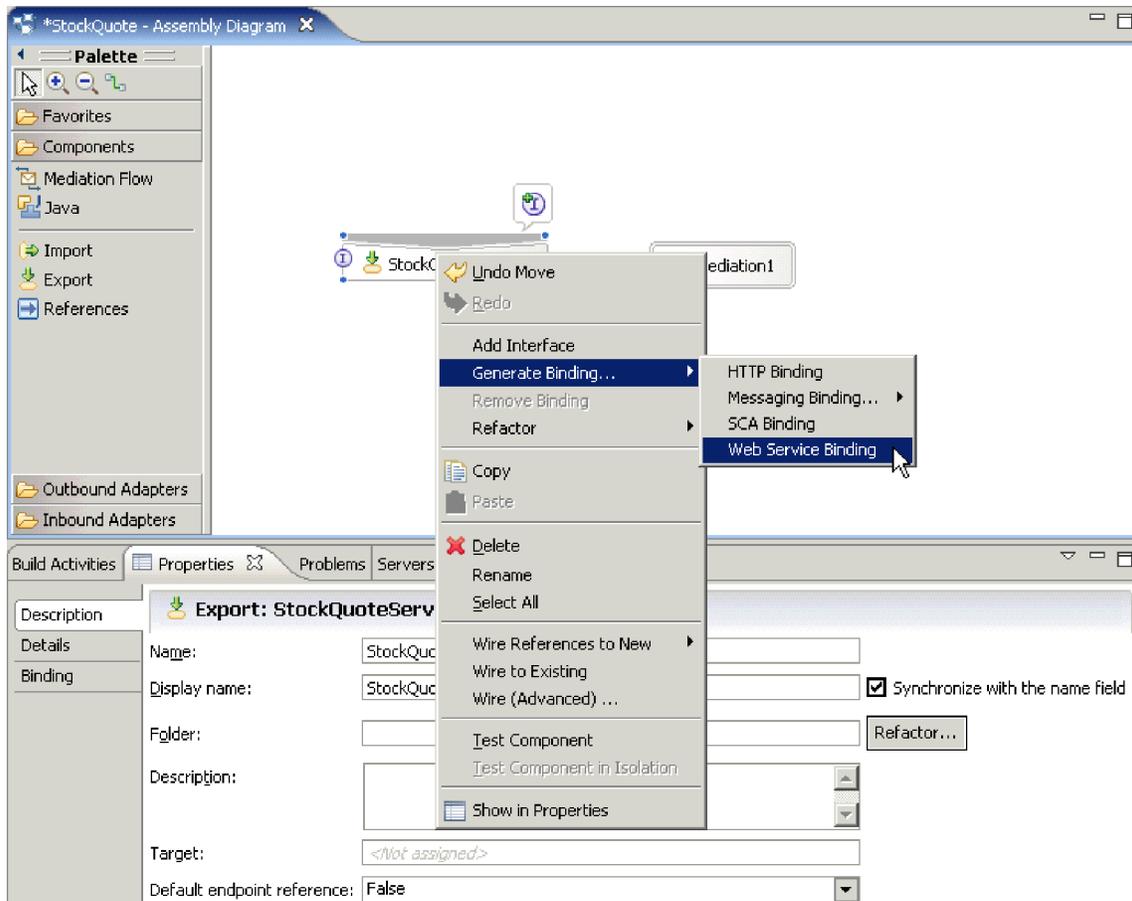
1. In the Business Integration view, expand the **StockQuote** module.
2. To open the Assembly Editor, double-click **Assembly Diagram**. The assembly editor opens showing mediation flow component named StockQuote.
3. In the **Resources** library's Interfaces category, select **StockQuoteService** and drag it on to the assembly editor canvas.



4. In the Component Creation dialog that pops up, choose to create an **Export with no Binding**.

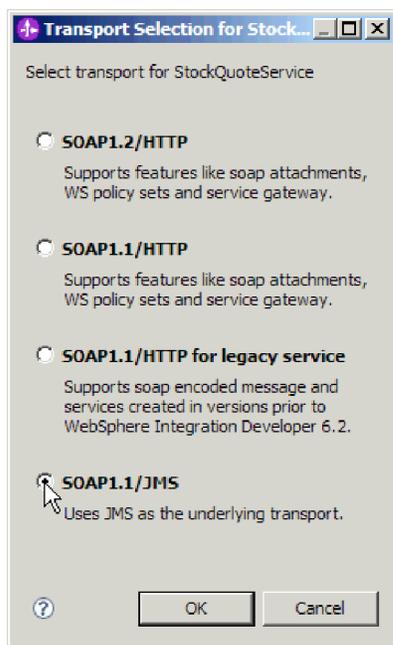


5. Click **StockQuoteServiceExport1** in the Assembly Editor to highlight its name. Type **StockQuoteService** to rename the export. Notice that the name change is reflected in the Description tab in the export icon's Properties view.
6. Right-click **StockQuoteService** and choose **Generate Binding... > Web Service Binding**.



You are generating a binding of type web service, using soap/jms as the transport protocol.

7. Select **SOAP1.1/JMS** as the transport, and click **OK**.

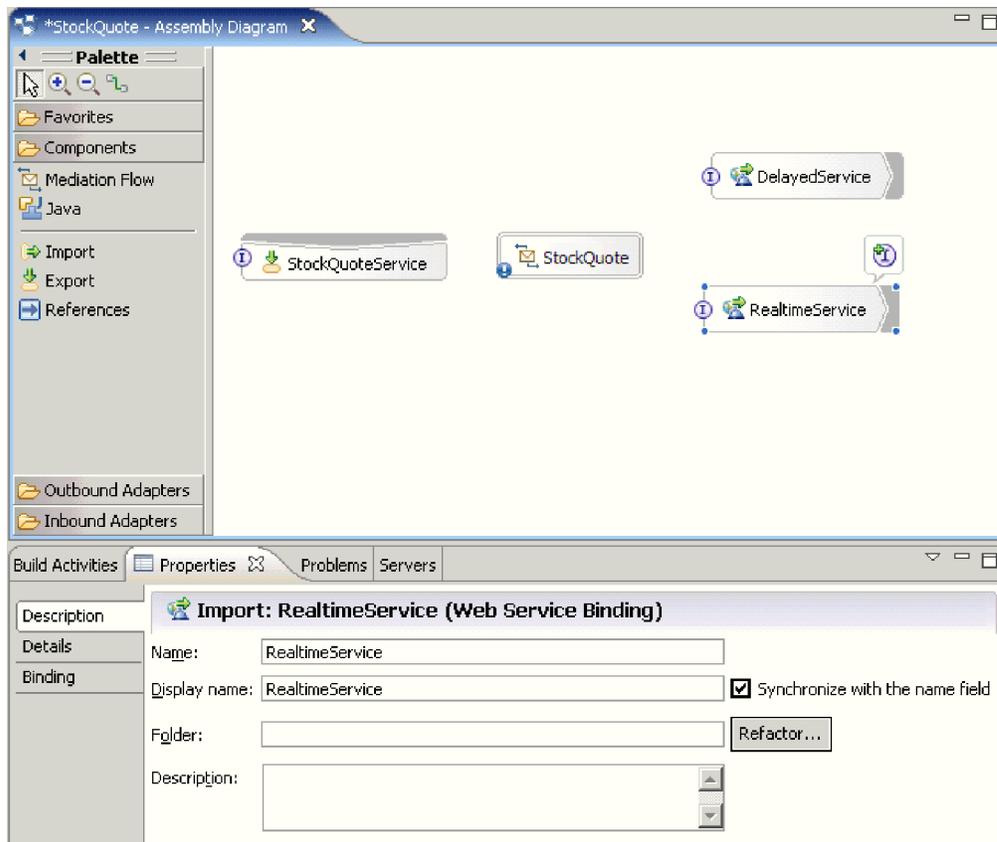


Look under the resource library's **Web Service Ports** category for the generated wsdl port, `StockQuoteService_StockQuoteServiceJms` Port.

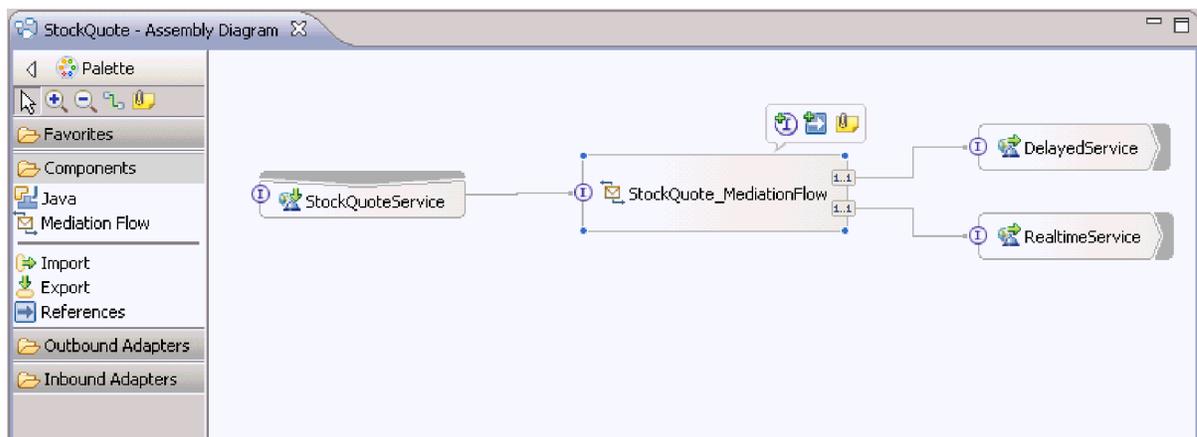
- In the **Resources** library's Interfaces category select **DelayedServicePortType** and drag it onto the Assembly Editor canvas. In the **Component Creation** dialog choose to create an **Import with Web Service Binding** and click **OK**.
- In the details dialog that pops up, keep the default setting to use an existing web service port. Click **Browse**.



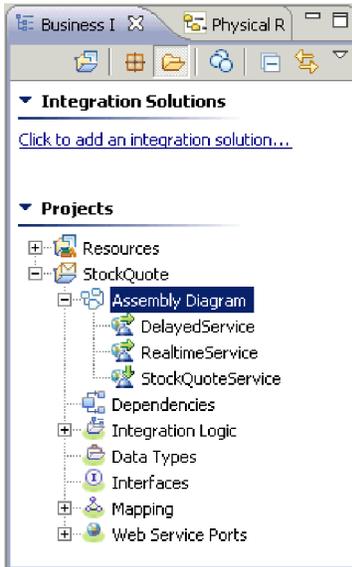
- In the selection wizard, click **DelayedServiceSOAP**. Click **OK**.
- In the transport selection window, click **OK** to keep the default transport.
- Click **OK** in the details dialog.
- Select the import and rename it to **DelayedService**. Click the Binding tab in the Properties view to see the binding information.
- Drag **RealtimeServicePortType** onto the Assembly Editor canvas, and create an **Import with Web Service Binding**.
- Specify **RealtimeServiceSOAP** as the port, and keep the default transport protocol.
- Rename the import to **RealtimeService**.



17. Click **StockQuote**, the mediation flow component that was created with the mediation module. Rename it to **StockQuote\_MediationFlow**.
18. Hover your mouse over **StockQuoteService**. An orange handle appears. This is the wire's source node.
19. Drag the wire to the target node, **StockQuote\_MediationFlow**, and release.
20. Click **OK** on the Add Wire dialog. The source node's interface, **StockQuoteService**, is added to the target and the wire is created.
21. Create a wire from source **StockQuote\_MediationFlow** to target **DelayedService**, and click **OK**. A matching reference, **DelayedServicePortTypePartner**, is created on the source and the wire is created.
22. Create a wire from source **StockQuote\_MediationFlow** to target **RealtimeService**, and click **OK**. A matching reference, **RealtimeServicePortTypePartner**, is created on the source and the wire is created.



23. Generate the implementation for StockQuote\_MediationFlow. In the assembly editor, select the component, right click, and select **Generate Implementation**. Select the StockQuote folder, and click OK. The mediation flow editor opens, showing the source interface and target references.
24. Since the initial mediation component was created without references or an interface attached to it, they have to be synchronized. To do this, right click **StockQuote\_MediationFlow**, select **Synchronize Interfaces and References... > to Implementation**. Click **Yes**.
25. Save the assembly diagram. Expand the StockQuote assembly under the module in the Business Integration view to see the artifacts created.




---

## Implementing the mediation

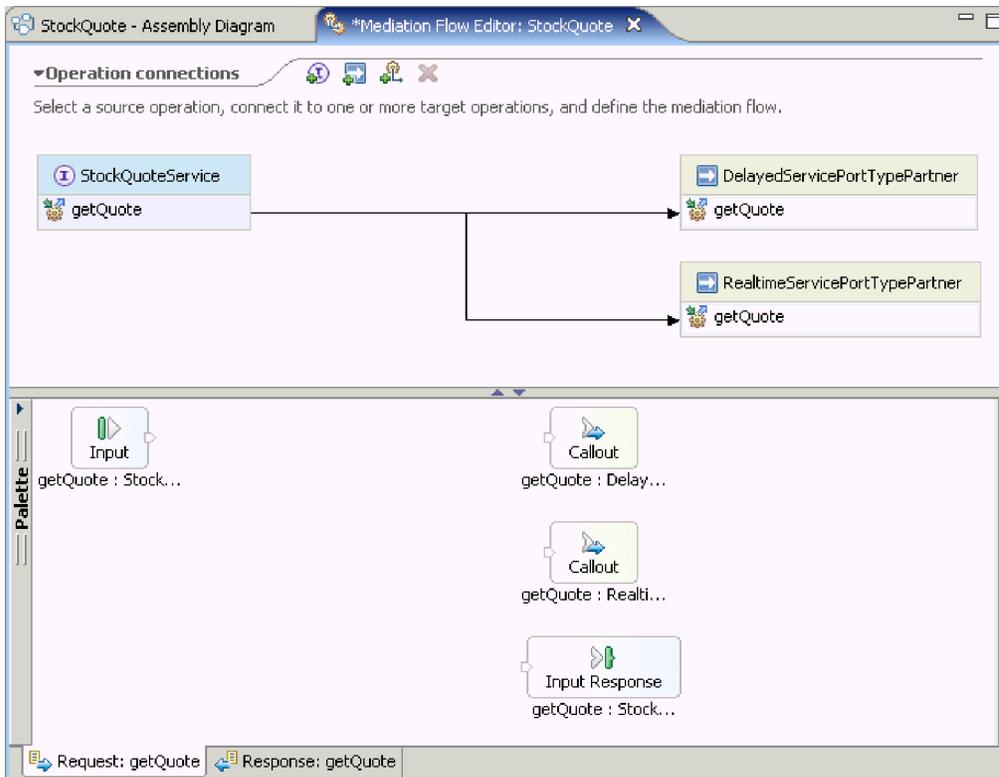
Select the sending and receiving operations, and visually connect them together. These will be end points of the mediation flow. Next, add mediation primitives between the end points to log the message, retrieve the customer's subscription level from a database, conditionally route the message based on the value retrieved, and transform the message to match the format of the receiving operation.

Follow the instructions in these topics to build your mediation flow.

## Defining the end points of the mediation

Follow these steps to define the end points of the mediation flow.

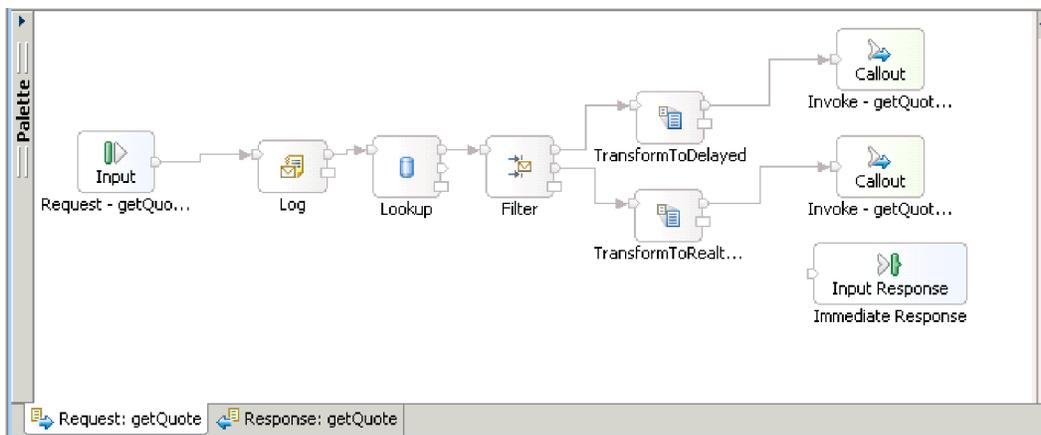
1. Double-click the mediation flow component to open the Mediation Flow Editor.
2. In the Operation connections section, hover over the getQuote operation of the StockQuoteService interface. An orange handle appears. Grab the handle and connect StockQuoteService interface's getQuote operation to DelayedServicePortTypePartner interface's getQuote operation.
3. Connect StockQuoteService interface's getQuote operation to RealtimeServicePortTypePartner interface's getQuote operation. When you select a source operation, the mediation flow palette becomes available.



## Building the request flow

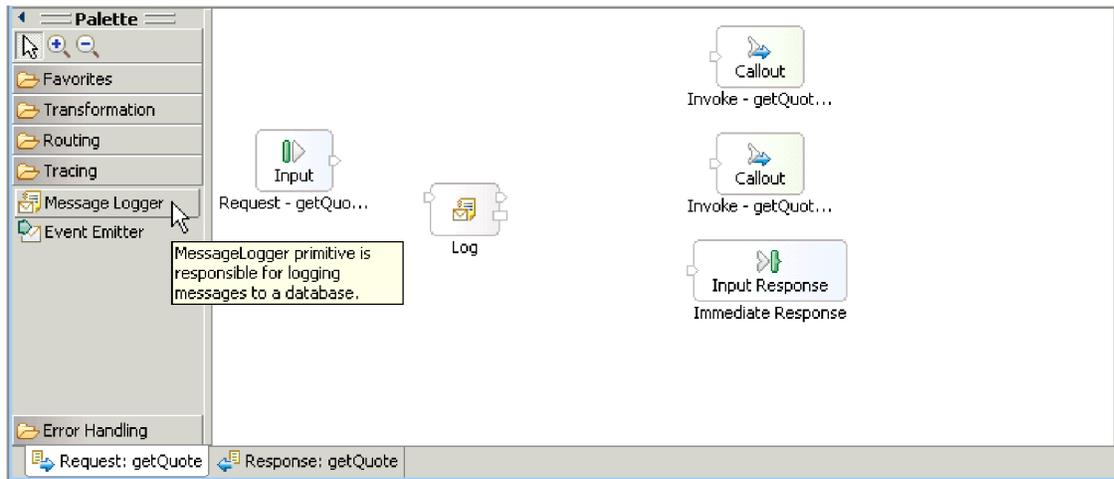
Build the request flow for the getQuote operation: create the mediation primitives, set their properties, and wire the flow.

When you connect source and target operations, a request flow and a response flow are created for each source operation that is connected. Select the source operation to view the request flow. The flow is represented from left to right. You will see an input node on the left. This is where the request message enters the flow. On the right are two callout nodes, one for each target operation. You will also see an ImmediateResponse node. This node is used when a message is to be returned to the client after the flow is executed. We will not use this node in our sample. You will add mediation primitives between the input and callout nodes and wire the flow. The completed request flow is shown below:

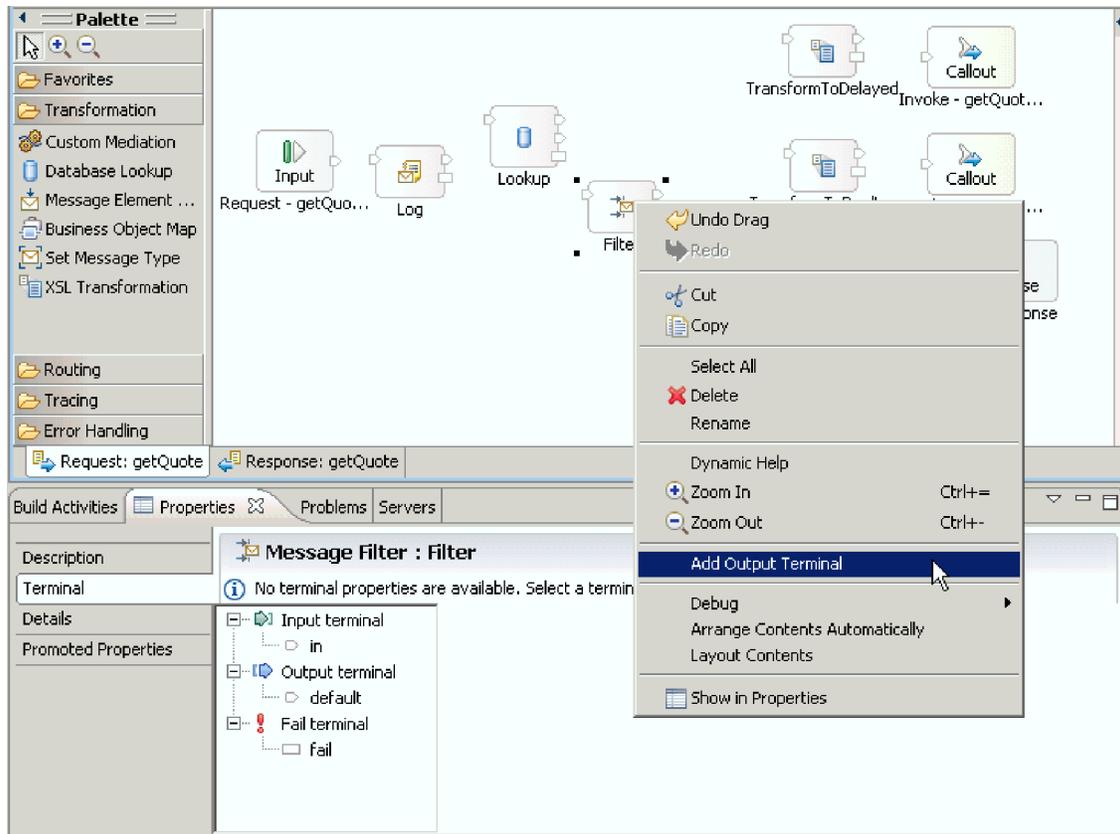


To build the request flow, complete the following steps:

1. In the Operation connections section at the top of the editor, select the source interface's getQuote operation. You will see the request flow with an input and a callout for each target operation. You can now add mediation primitives and wire the request flow.
2. Click the Request: getQuote tab in the middle section. Click the Tracing palette category to expand the group.
3. Click the **Message Logger** primitive and drop it onto the request flow canvas, and rename the primitive Log.



4. Select a **Database Lookup** primitive from the Transformation palette group, drop it onto the request flow canvas, and name it Lookup.
5. Select a **Message Filter** primitive from the Routing palette group, drop it onto the request flow canvas, and name it Filter.
6. Select an **XSL Transformation** primitive from the Transformation palette group, drop it onto the request flow canvas, and name it TransformToDelayed.
7. Select another **XSL Transformation** primitive from the palette, drop it onto the request flow canvas, and name it TransformToRealtime.
8. Right-click the **Filter** primitive and select **Add Output Terminal**. In the **Terminal name** field, type realtime and click **OK**.



You will create an XPath expression and associate it with this terminal in step 13.

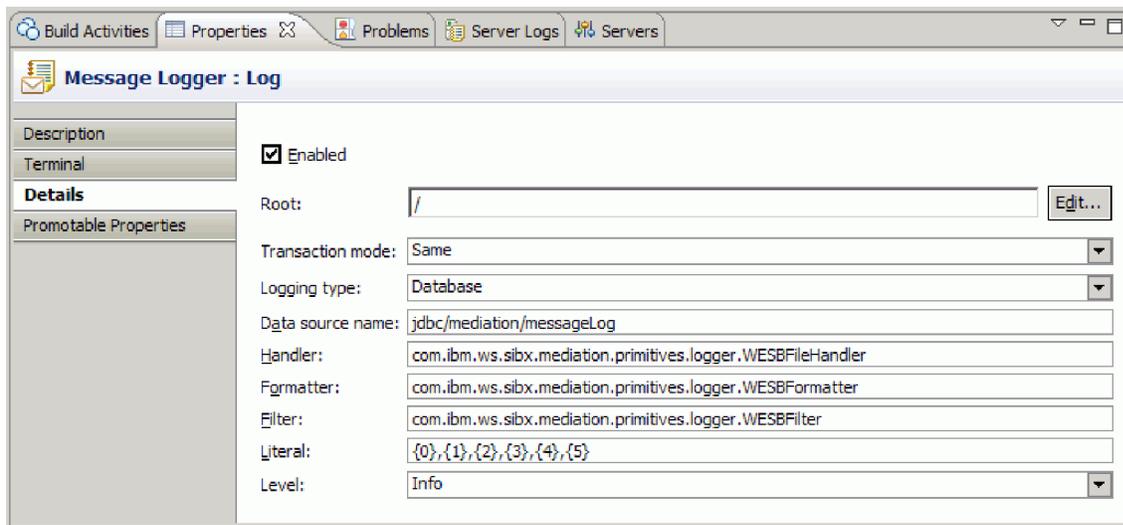
9. In the request flow canvas, wire the primitives:
  - The output terminal of **getQuote : StockQuoteService** to the input terminal of **Log**
  - The output terminal of **Log** to the input terminal of **Lookup**
  - The output terminal of **Lookup** to the input terminal of **Filter**
  - The default terminal of **Filter** to the input terminal of **TransformToDelayed**
  - The realtime terminal of **Filter** (created in step 8) to the input terminal of **TransformToRealtime**
  - The output terminal of **TransformToDelayed** to the input terminal of **getQuote : DelayedServicePortTypePartner**
  - The output terminal of **TransformToRealtime** to the input terminal of **getQuote : RealtimeServicePortTypePartner**
10. We will now add the business object that was created earlier to the correlation context of the input node **getQuote : StockQuoteService**. This will allow the property **subscriptionLevel** to persist in the message flow. Click **getQuote : StockQuoteService** input node and switch to the Details tab in the Properties view. In the **Correlation context** field, click **Browse**. Select **SubscriptionInformation** under matching data types, and click **OK**. `{http://Resources}SubscriptionInformation` now appears in the **Correlation context**.



11. Click **Log** in the request flow canvas to see the primitive's properties in the Properties view. Click the Details tab to view the properties. Use the default database to log the message. Enter these properties:

Table 1. Message Logger properties

Property	Value
Data source name	jdbc/mediation/messageLog
Root	/
Transaction mode	Same

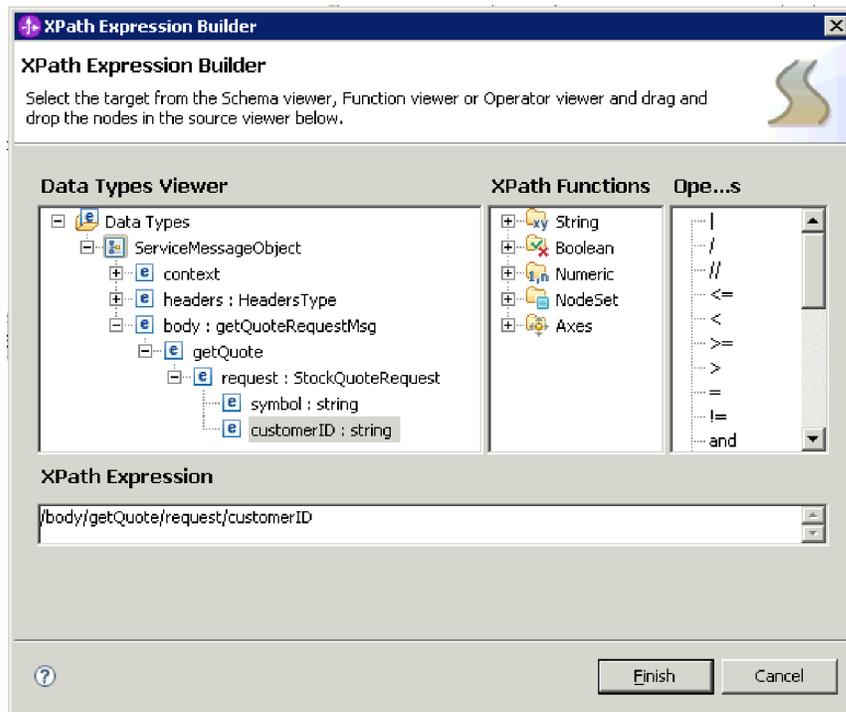


Root specifies the part of the message to be logged. '/' logs the complete message while '/body' logs only the body of the message.

12. Click **Lookup** in the request flow canvas, and enter these property values in the Details tab:

Table 2. Database Lookup properties

Property	Value
Data source name	jdbc/sample/CustomerDatabase
Table	CUSTOMERTABLE
Search column	CUSTOMERID
Search location	Click <b>Edit...</b> In the XPath Expression Builder, navigate to/body/getQuote/request/customerID and double click on customerID. The path appears in the XPath Expression section of the builder. Click <b>Finish</b> .

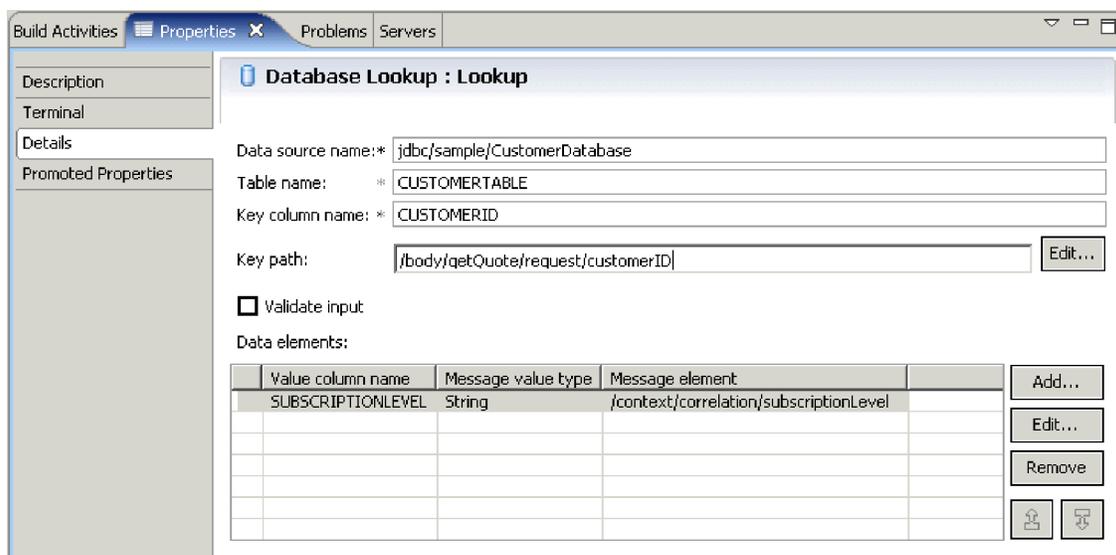


In the Data elements table, click **Add...** and enter the following values:

Table 3. Data elements table properties

Column	Value
Column	SUBSCRIPTIONLEVEL
Type	String
Target location	/context/correlation/subscriptionLevel

Leave **Validate input** unchecked.



- Click **Filter** in the request flow canvas. The default is to map to TransformToDelayed. The pattern for mapping to TransformToRealtime must be set. Select the Details view:

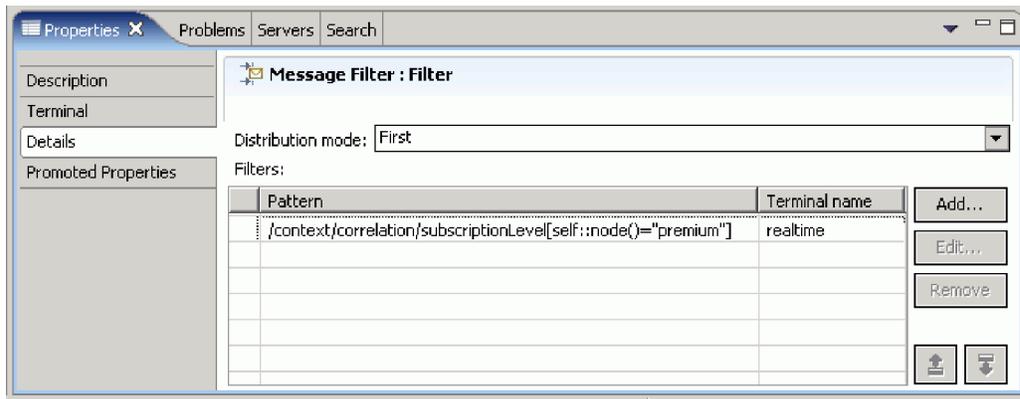
Table 4. Message Filter properties

Property	Value
Distribution mode	First

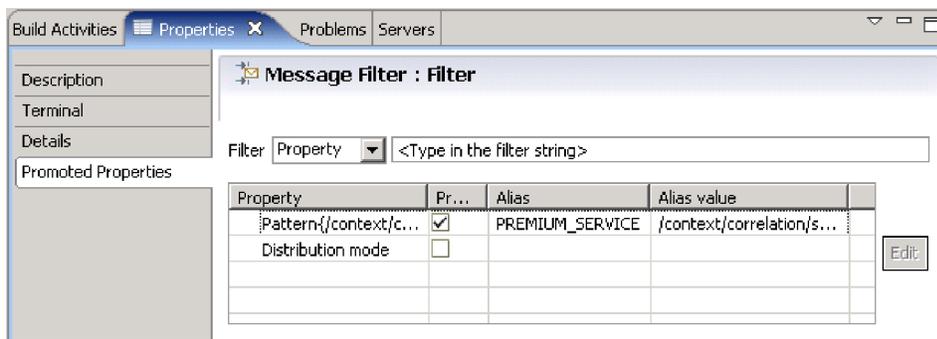
In the Filter table, click **Add...** and enter the following values:

Table 5. Filter table properties

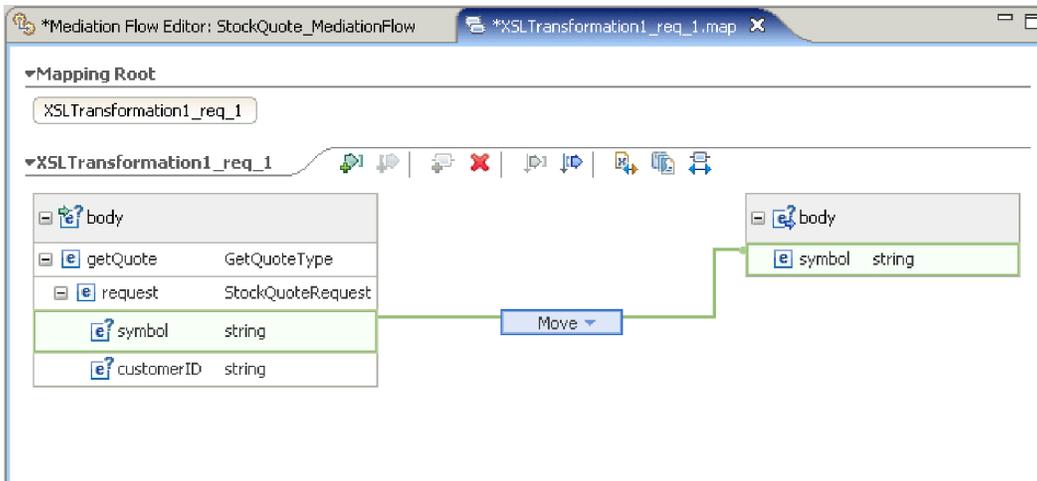
Column	Value
Pattern	/context/correlation/ subscriptionLevel[self::node()="premium"]
Terminal name	realtime



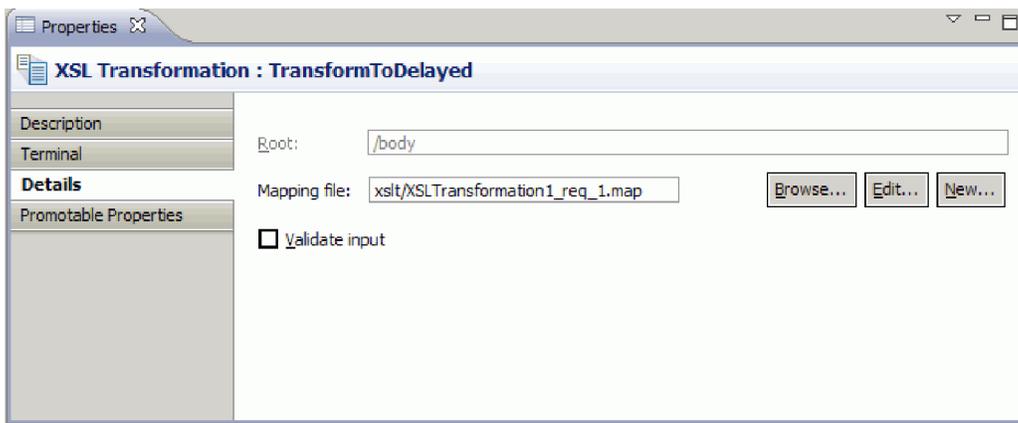
14. A promoted property can be changed by an administrator at runtime. The pattern property can be changed at runtime to change the quality of service. To promote the pattern property:
  - a. Click the Promoted properties tab.
  - b. Click the promoted check box of the pattern property.
  - c. Click the alias Filter1.filters1. Type PREMIUM\_SERVICE to rename the alias.



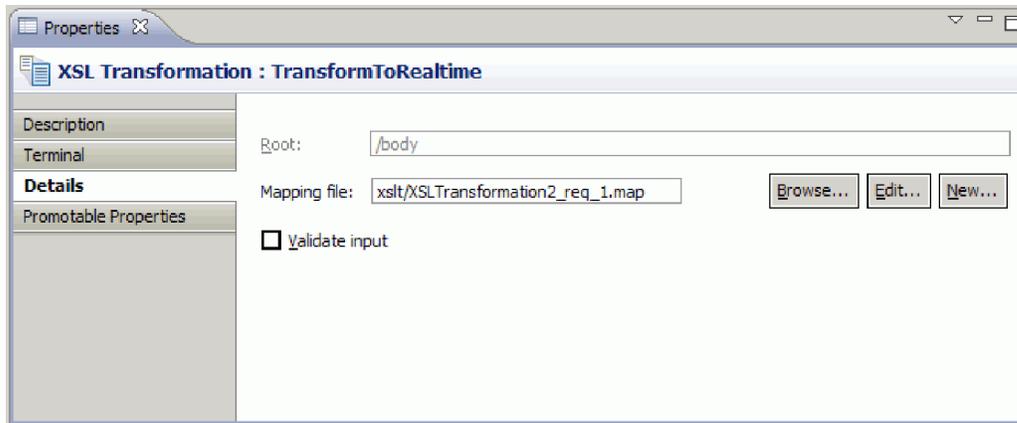
15. Set the properties for the XSL Transformation primitive TransformToDelayed:
  - a. Select the **TransformToDelayed** primitive in the request flow canvas and double-click it.
  - b. Click **Next** to see the root, input, and output message types that will be mapped. Click **Finish** to accept the defaults. This launches the mapping editor.
  - c. On the left side, the Input object side, expand **body > getQuote > request**. In the Output object side, the right side, expand **body**. Click symbol on the left side and drag it to symbol on the right side to wire them together and create the mapping.



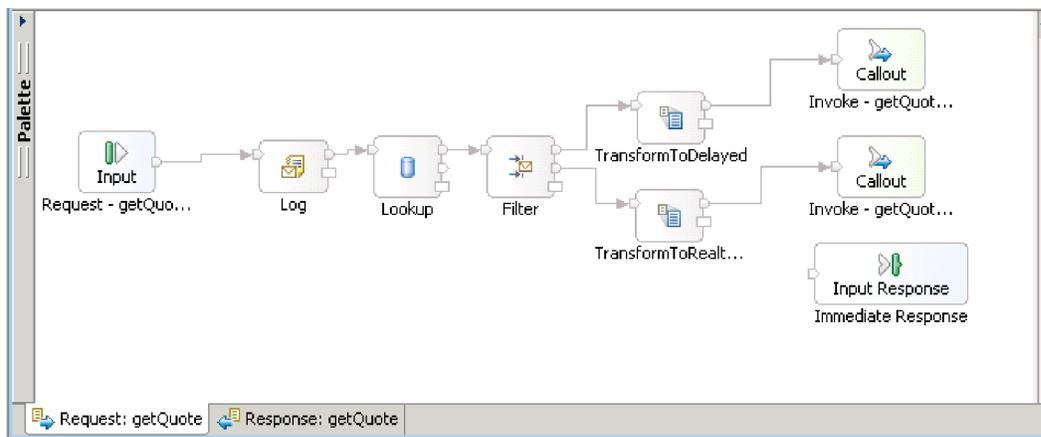
- d. Save your changes and close the mapping editor. The mapping file is displayed in the Details tab of the Properties view.



16. Similarly, set the properties for the XSL Transformation primitive TransformToRealtime:
  - a. Select the **TransformToRealtime** primitive in the request flow canvas and double-click it.
  - b. The New XML Mapping wizard opens. Click **Next** to see the root, input, and output message types that will be mapped. Click **Finish** to accept the defaults, this opens the mapping editor.
  - c. On the left side, expand **body > getQuote > request**. On the right side, expand **body**. Click *symbol* on the left side and drag it to *symbol* on the right side to create the mapping.
  - d. Save your changes and close the mapping editor. The mapping file is displayed in the Details tab of the Properties view.



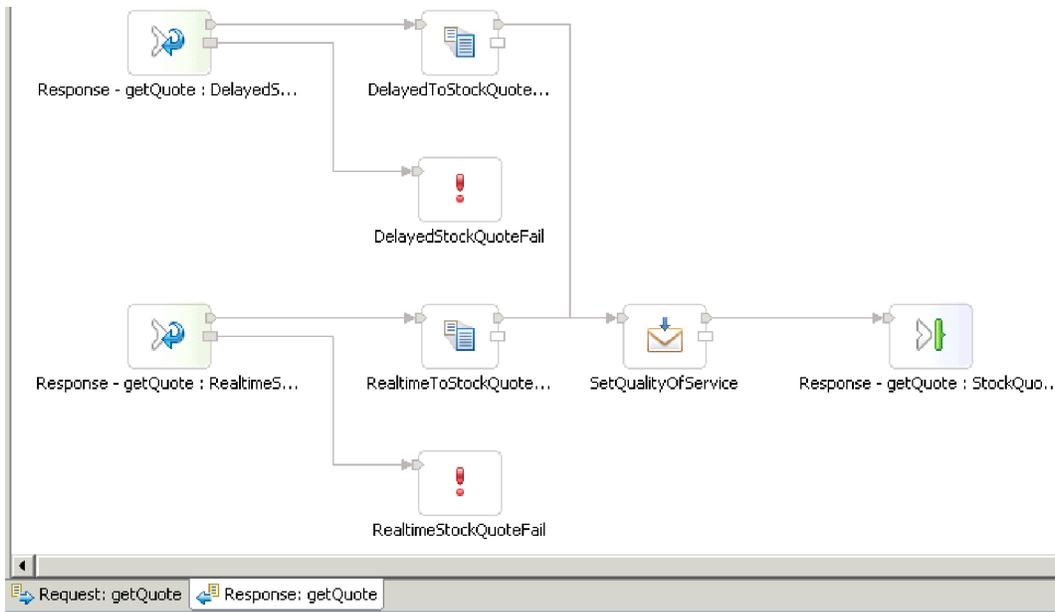
17. Save the request flow.



## Building the response flow

Build the response flow for the getQuote operation: create the mediation primitives, set their properties, and wire the flow.

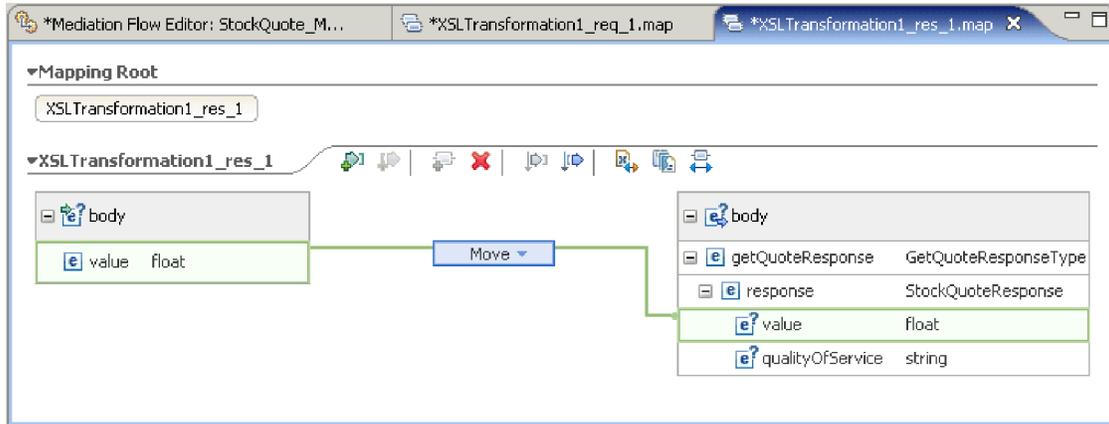
Response flows are represented from left to right in the editor. On the left are the callout nodes, one for each target operation; this is where the returning message enters the response flow. On the right is an inputResponse node, which represents the message returning to the source operation. You will add mediation primitives between the nodes and wire the flow. The completed response flow is shown below:



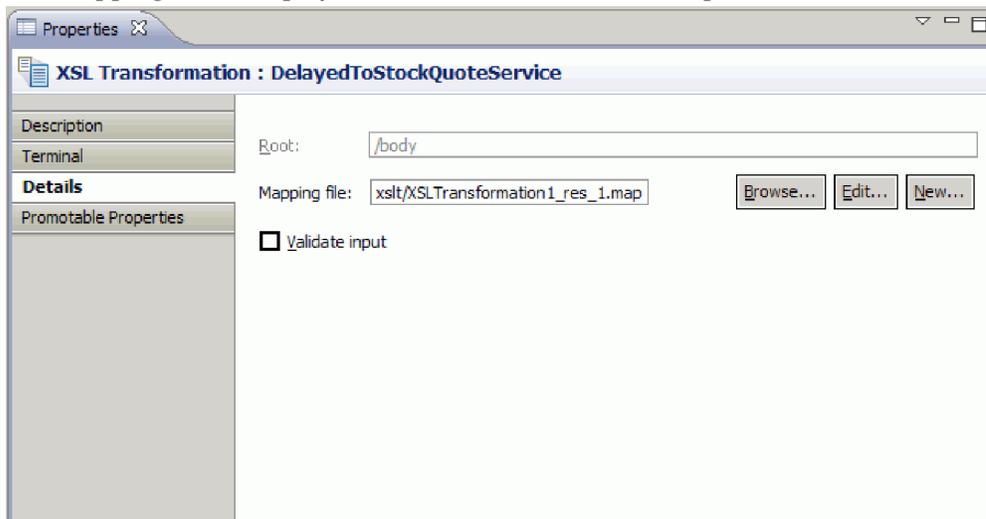
To build the response flow, complete the following steps:

1. Click the **Response** tab to view the response flow.
2. Select an **XSL Transformation** primitive from the palette, drop it onto the response flow canvas, and rename it **DelayedToStockQuoteService**.
3. Select another **XSL Transformation** primitive from the palette, drop it onto the response flow canvas, and rename it **RealtimeToStockQuoteService**.
4. Select a **Message Element Setter** primitive from the palette, drop it onto the canvas, and rename it **SetQualityOfService**.
5. Select a **Fail** primitive from the palette, drop it onto the canvas and rename it **DelayedStockQuoteFail**. Add another **Fail** primitive to the canvas and rename it **RealtimeStockQuoteFail**.
6. Wire the primitives:
  - The output terminal of **getQuote : DelayedServicePortTypePartner** to the input terminal of **DelayedToStockQuoteService**
  - The output terminal of **getQuote : RealtimeServicePortTypePartner** to the input terminal of **RealtimeToStockQuoteService**
  - The output terminal of **DelayedToStockQuoteService** to the input terminal of **SetQualityOfService**
  - The output terminal of **RealtimeToStockQuoteService** to the input terminal of **SetQualityOfService**
  - The output terminal of **SetQualityOfService** to the input terminal of **getQuote : StockQuoteService**
  - The fail terminal of **getQuote : DelayedServicePortTypePartner** to the input terminal of **DelayedStockQuoteFail**
  - The fail terminal of **getQuote : RealtimeServicePortTypePartner** to the input terminal of **RealtimeStockQuoteFail**
7. Set the properties for the XSLT primitive **DelayedToStockQuoteService**:
  - a. Select the **DelayedToStockQuoteService** primitive in the response flow canvas and double-click it.
  - b. The New XML Mapping wizard opens. Click **Next** to see the root, input, and output message types that will be mapped. Accept the defaults and click **Finish**.

- c. In the input object section (left side) of the mapping editor, expand **body**. In the output object section (right side), expand **body > getQuoteResponse > response**.
- d. Click value on the left side and drag it to value on the right side to wire them and create the mapping.
- e. Save your changes and close the mapping editor.

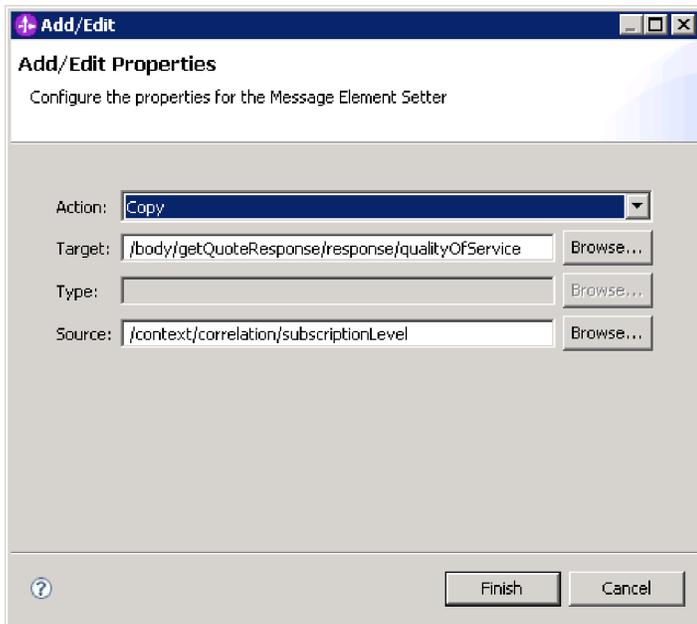


The mapping file is displayed in the Details tab of the Properties view.

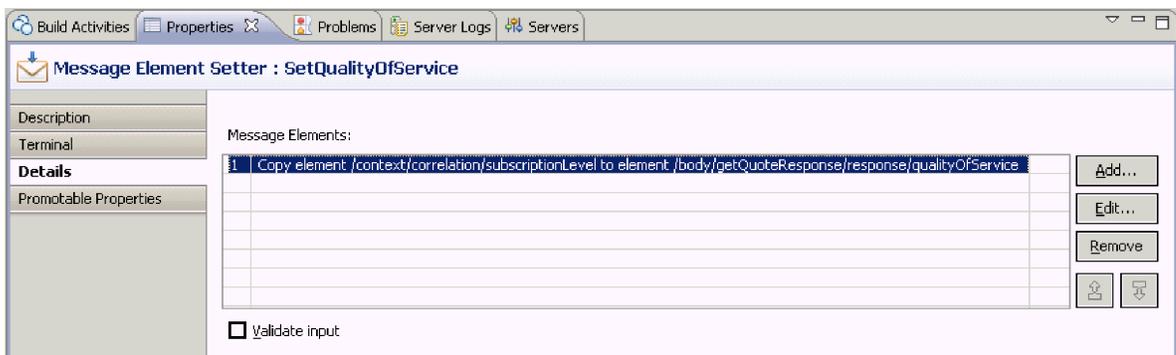


8. Similarly, set the properties for the XSLT primitive **RealtimeToStockQuoteService**:
  - a. Select the **RealtimeToStockQuoteService** primitive in the response flow canvas and double-click it.
  - b. The New XML Mapping wizard opens. Click **Next** to see the root, input, and output message types that will be mapped. Accept the defaults and click **Finish**.
  - c. In the input object section (left side) of the mapping editor, expand **body**. In the output object section (right side), expand **body > getQuoteResponse > response**.
  - d. Click value on the left side and drag it to value on the right side to wire them and create the mapping.
  - e. Save your changes and close the mapping editor. The mapping file and associated XSL style sheet are displayed in the Details tab of the Properties view.
9. Set the properties for the Message Element Setter primitive **SetQualityOfService**:
  - a. Select the **SetQualityOfService** primitive in the response flow canvas. Switch to the **Details** tab in the Properties view.
  - b. Click **Add...** to start the Add/Edit Properties wizard.

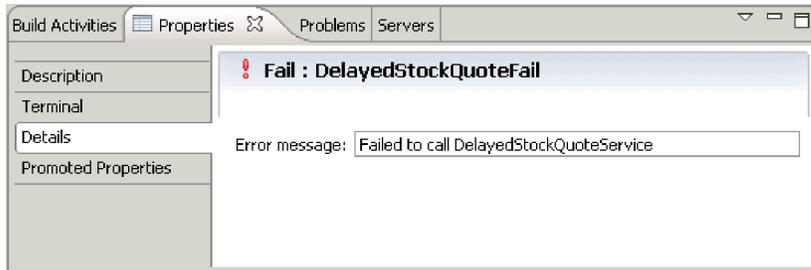
- c. From the Action dropdown select **Copy**. For Target select **Browse**, which will bring up the XPath Expression Builder.
- d. In the Data Types Viewer, expand **ServiceMessageObject > body > getQuoteResponse > response : StockQuoteResponse**. Select **qualityOfService** and drag it to the XPath Expression field below. Click **Finish**.
- e. For the Source field select **Browse**. In the Data Types Viewer, expand **ServiceMessageObject > context > correlation**. Select **subscriptionLevel** and drag it to the XPath Expression field below. Click **Finish**.
- f. In the source field enter `/context/correlation/subscriptionLevel`.



The target, type and value columns of the table are populated in the first row as shown. To edit them later, select the row and click **Edit...**



10. For the two Fail primitives:
  - a. Right click **DelayedStockQuoteFail** and select **Show in Properties**. Click **Details** and in the Error message field enter Failed to call DelayedStockQuoteService.



- b. For **RealtimeStockQuoteFail** enter Failed to call RealtimeStockQuoteService in the Error message field.
11. Save the flow by pressing **Ctrl-S**.

---

## Chapter 5. Test

After you build or import the sample, you can test the StockQuote application in the integrated test client or the debugger.

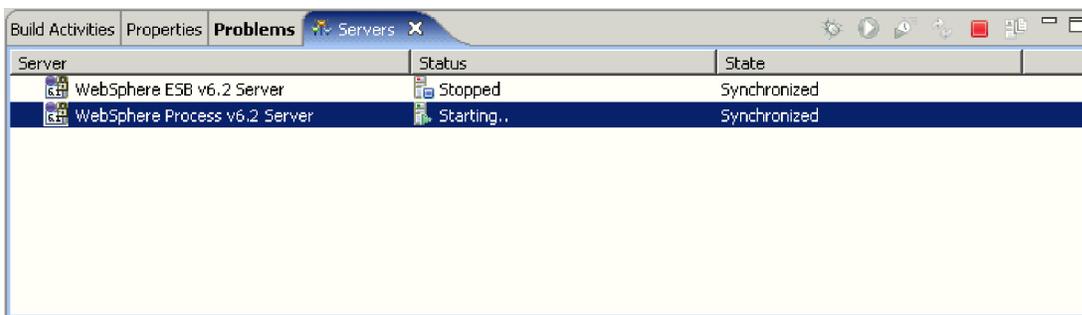
The Stock Quote sample includes some runtime components; a real-time Web service and a delayed Web service that generate quotes randomly for the symbol specified in the test client, and a Derby database that has preloaded customer data. Before you test the sample, you must run scripts to install the runtime components; You only need to run the scripts once. Then, to run the sample, add the StockQuoteApp project to the server and launch the test client.

---

### Installing the runtime components

To run the script that installs the customer database and back-end on the server, complete the following steps:

1. Switch to the Server view. If only one server profile is installed (WebSphere Process Server or WebSphere ESB Server), right-click on the server and select **Start** to start the server. If both server profiles are installed, choose one of them.



2. You will now install the runtime components used in this sample. When the server has started successfully, open a command prompt and switch to *Install Shared Resources Directory/plugins/com.ibm.wbit.samples.content/artifacts/stockquote/bin/*. **Note:** You may find multiple versions of the *com.ibm.wbit.samples.content* in your install. Always choose the one with the highest version number since it will be the latest one.

3. Run one of the following commands:

- If you started WebSphere Process Server: *Installation Directory/runtimes/bi\_v62/bin/wsadmin -f wid-install.jacl -profileName wps -username username -password password*
- If you started WebSphere ESB Server: *Installation Directory/runtimes/bi\_v62/bin/wsadmin -f wid-install.jacl -profileName esb -username username -password password*

Keep the command prompt open as you will need it again. The default user is 'admin' and password is 'admin'.

4. When the script has finished successfully it will display the port number of the server on which it was installed. The port number should be 9080 as shown in the following image.

```

Command Prompt
erver repository.
ADMA5053I: The library references for the installed optional package are created
ADMA5005I: The application SQSample is configured in the WebSphere Application S
erver repository.
ADMA5001I: The application binaries are saved in D:\Program Files\IBM\WID61\pf\w
ps\wstemp\Script11851898e2b\workspace\cells\widCell\applications\SQSample.ear\SQ
Sample.ear
ADMA5005I: The application SQSample is configured in the WebSphere Application S
erver repository.
SECJ0400I: Successfully updated the application SQSample with the appContextIDFor
Security information.
ADMA5011I: The cleanup of the temp directory for application SQSample is complet
e.
ADMA5013I: Application SQSample installed successfully.
SAMP INFO: The install command for ESBStockQuoteSample completed successfully.

-----
Save
-----

-----
Summary
-----

*****
Port number is 9080.
*****

The install command succeeded for the following Samples:

    ESBStockQuoteSample

All commands ran successfully.
-----

```

If the port number is 9080 as shown, you can skip the following steps.

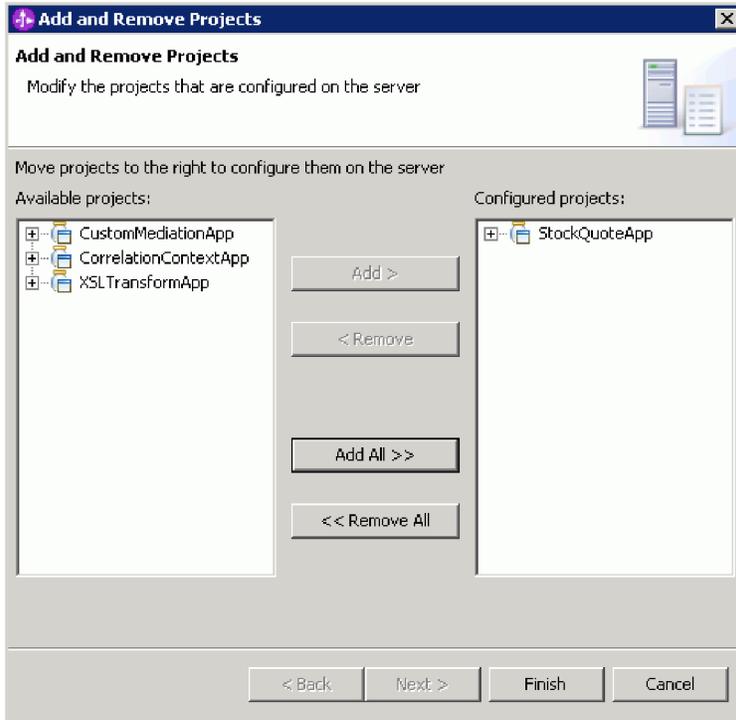
5. If the port is different from 9080 we have to ensure that the Stock Quote application looks for the database on the correct port. This will affect the imports. To change the port:
  - a. Open the Assembly Diagram of the Stock Quote module.
  - b. Select the DelayedService import and in the Properties view, click **Binding**.
  - c. The Address text box will contain something similar to `http://localhost:9080/DelayedService/services/DelayedServiceSOAP`.
  - d. Overwrite "9080" with the port number displayed when you ran the script.
  - e. Follow the previous steps for the RealtimeService import and then save your module.
6. Restart the server after the script has been installed successfully by right-clicking it, selecting **Restart**.

---

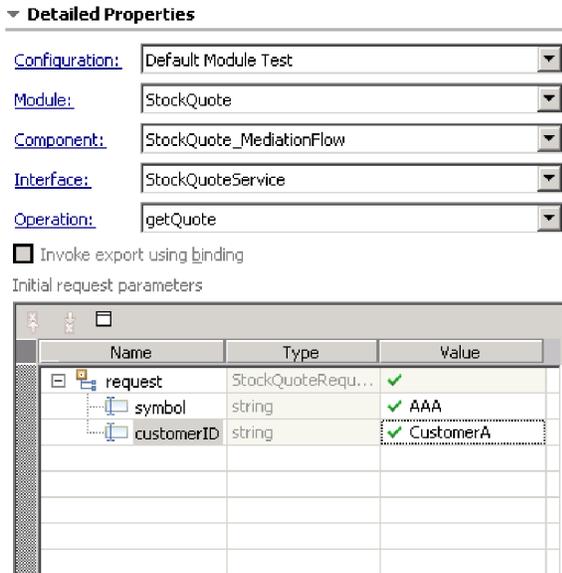
## Testing the mediation flow

After running the script, you have to add the StockQuoteApp to the running server. Then, you can test your mediation flow in the test client.

1. To add the StockQuoteApp project to the server, right-click on the server and select **Add and remove project....** Select the StockQuoteApp project from the Available projects list and click **Add >** to add it to the Configured projects list. Click **Finish** and wait for the server to finish publishing.



2. Open the Assembly Diagram of the StockQuote mediation module. Then right-click on the StockQuote\_MediationFlow component and select **Test Component**. The Events page of the integrated test client opens.
3. In the Events page, you can select the modules, components, interfaces and operations you wish to test. For this sample, ensure the Detailed Properties are:
  - Configuration: **Default Module Test**
  - Module: **StockQuote**
  - Component: **StockQuote\_MediationFlow**
  - Interface: **StockQuoteService**
  - Operation: **getQuote**
4. In the Initial request parameters table, you enter information by double clicking the cell in the value column. Double-click the value cell in the **symbol** row and enter AAA. In the same manner, enter CustomerA for the customerID.



- Next, push the **Continue** button  to invoke the getQuote operation. The Deployment Location dialog opens.
- Select a server and click **Finish**. You can run mediation flows on either WebSphere Process Server or WebSphere Enterprise Service Bus. Enter the User ID and password for your server. The default is admin/admin.
- You will see the resulting values for qualityOfService and value in the return parameters.

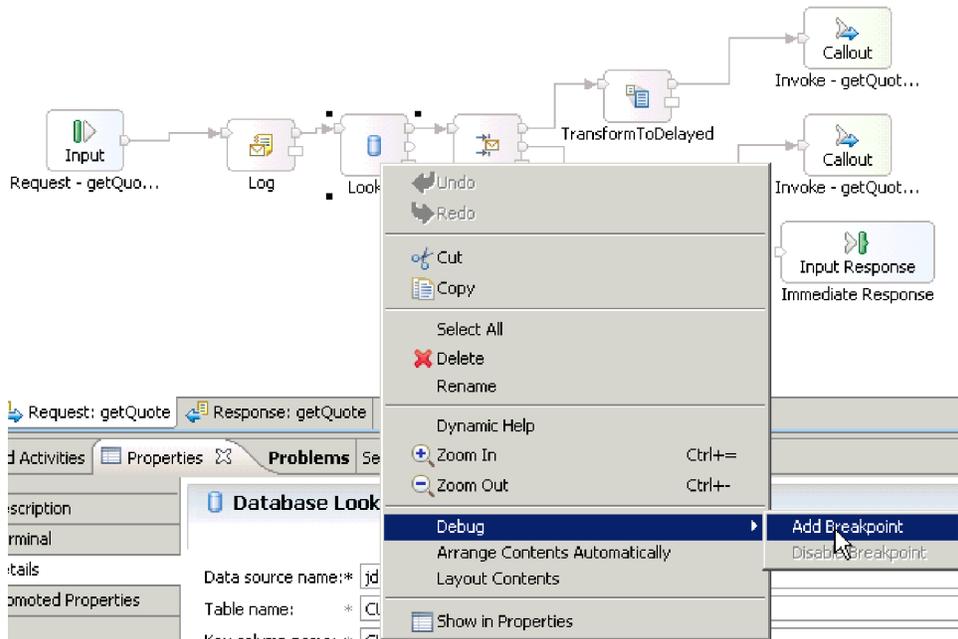
---

## Debugging the mediation flow

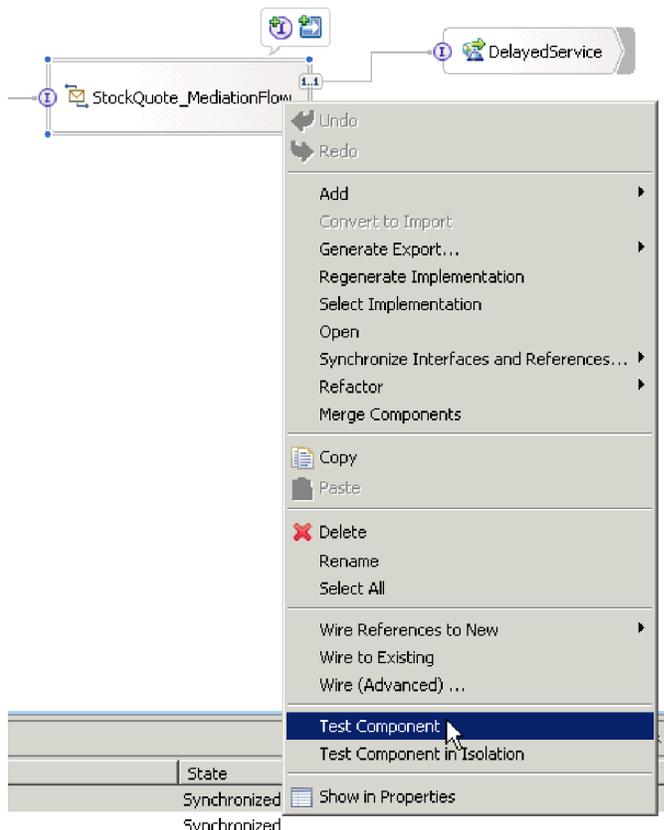
Debug the StockQuote\_MediationFlow component in the unit test environment by using the integration debugger.

The Instructions below assume that the runtime components have ready been installed. See “Installing the runtime components” on page 33 for information on how to install these components. database.

- To add a breakpoint to a mediation primitive, right-click on the primitive node in the request or response flow canvas, and select **Debug > Add Breakpoint**. Notice that a small blue icon is added to the top left-hand corner of the node, indicating that a breakpoint has been added to it.

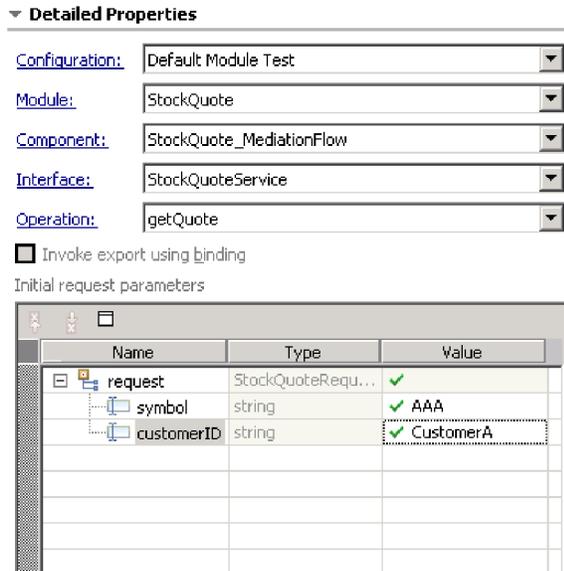


2. Select a server, right-click and click on it and select **Restart -> Debug** to restart the server in Debug mode. Note that if the server is already started, you must stop and start the server again in Debug mode.
3. When the server has started, open the Assembly Diagram of the StockQuote mediation module. Then right-click on the StockQuote\_MediationFlow component and select **Test Component**.



The Unit Test Environment opens.

4. In the Unit Test Environment you can select the modules, components, interfaces and operations you wish to test. For this sample, ensure the Detailed Properties are:
  - a. Configuration: **Default Module Test**
  - b. Module: **StockQuote**
  - c. Component: **StockQuote\_MediationFlow**
  - d. Interface: **StockQuoteService**
  - e. Operation: **getQuote**
5. In the Initial request parameters table, you enter information by double clicking the cell in the value column. Double-click the value cell in the **symbol** row and enter AAA. In the same manner, enter CustomerA for the customerID.



6. Next, push the **Continue** button . The Deployment Location dialog opens.
7. Select the server you started in Debug mode earlier and click **Finish**. Enter the User ID and password for your server. The default is admin/admin. When the flow reaches a breakpoint, you are prompted to open the Debug perspective. Click **Yes** to open the perspective.
8. You can view the values of message elements, paths taken and breakpoints reached in this view. To continue the flow through to the end or to the next breakpoint click the **Resume** button .

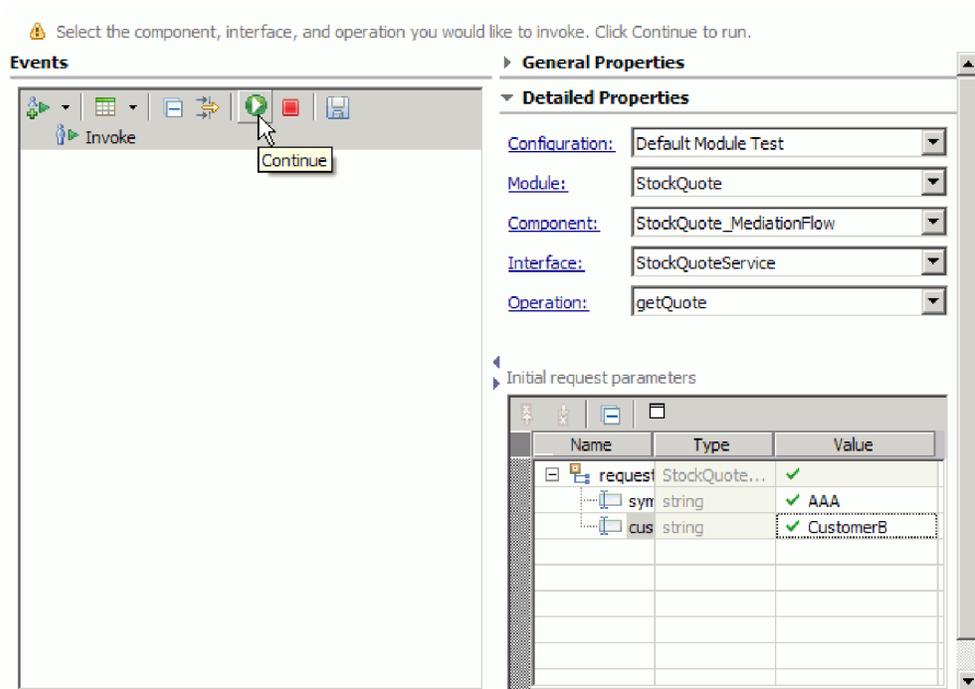
## Changing the quality of service at runtime

When we built the sample, we promoted the Filter pattern property, to allow us to change the value of the property at runtime. Now, we will change the value of the Filter pattern property in the unit test environment's administrative console, which will cause the request to be sent to a different service. We will view the result in the test client emulator.

To perform the actions described in this task, the server scripts must be run to install the runtime components. For instructions, see "Installing the runtime components" on page 33.

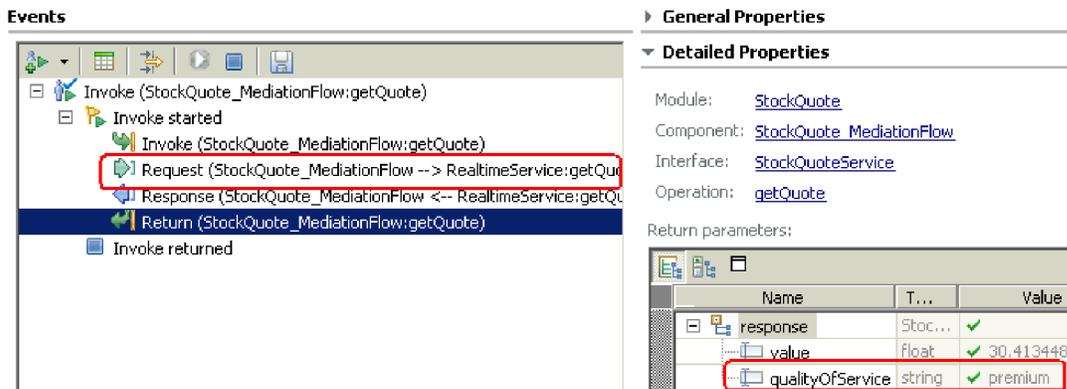
Open the StockQuote assembly diagram, and follow these steps to test the promoted property of the Filter primitive:

1. Right-click StockQuote\_MediationFlow and select **Test Component**.
2. In the Events page, enter these initial request parameters, and click **Continue**.
  - For symbol, enter AAA.
  - For customerID, enter CustomerB.

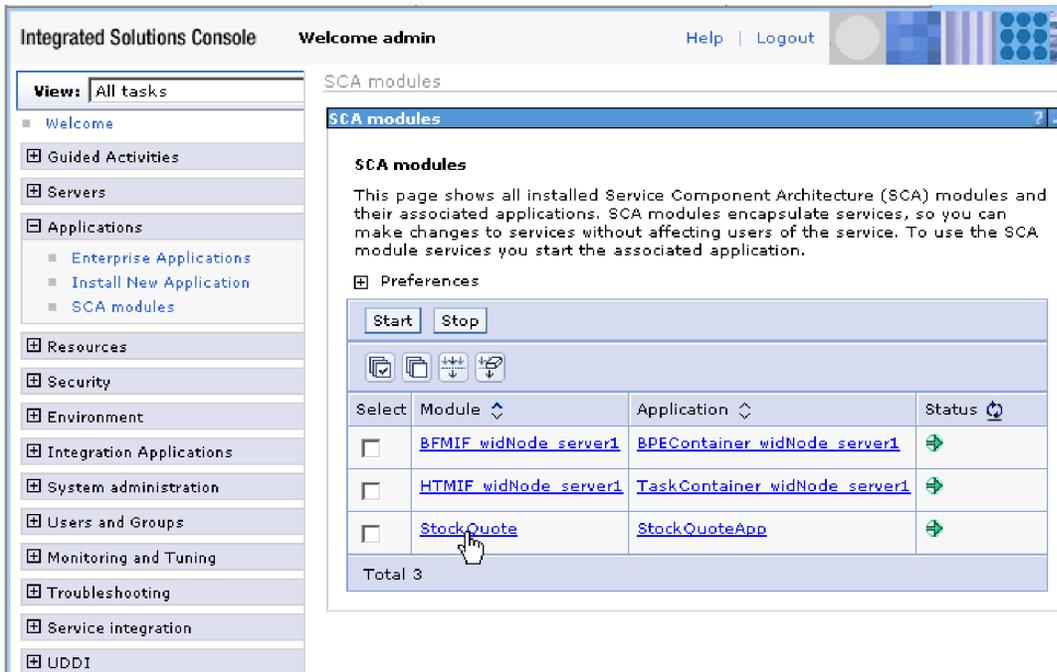


3. Select the server for the unit test environment. Click **Finish** and enter the username and password to login to the server. The default is admin\admin.

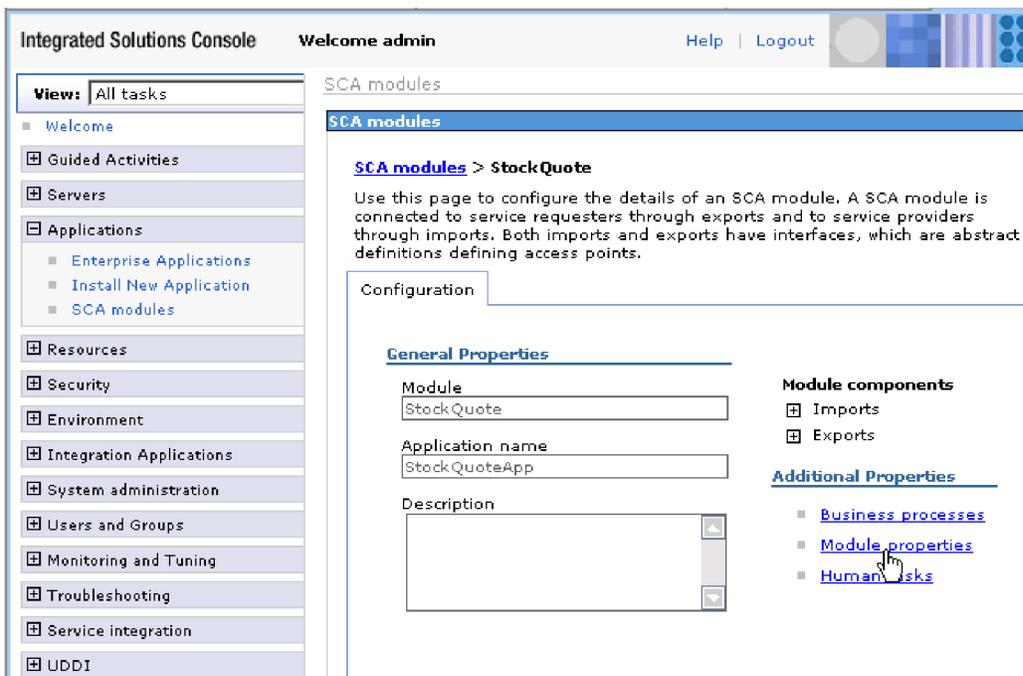
The results in the emulator show that the service invoked is RealtimeService. You can also see the value of the qualityOfService string which tells you that this customer's service level is premium.



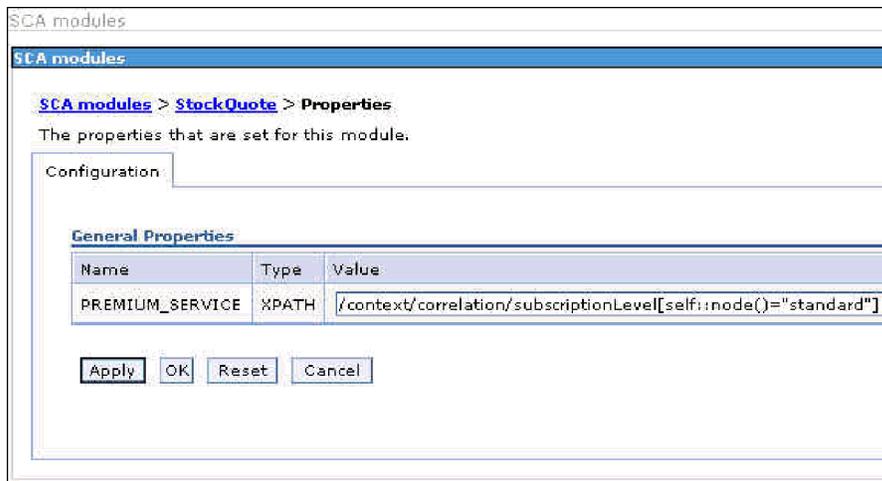
4. Switch to the Servers view. Right-click the server, and Run administrative console. In the log in window, enter your userid and password (the default is admin/admin). Click **Log in**.
5. In the administrative console, expand **Applications** and click **SCA modules**.
6. In the list of applications, click **StockQuote**.



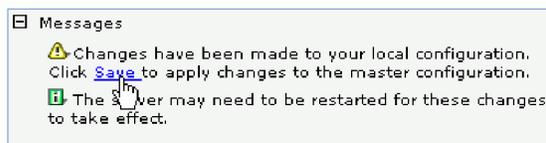
7. Click **Module Properties**.



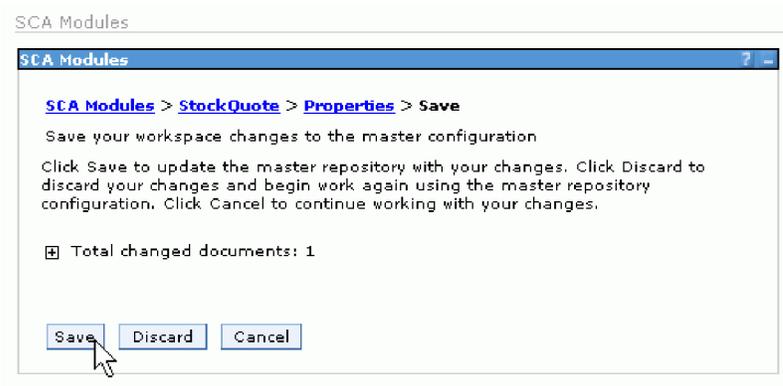
8. The property that we promoted earlier is displayed, showing the alias PREMIUM\_SERVICE. Click the value field of PREMIUM\_SERVICE, and change "premium" to "standard". Click **Apply**.



- In the messages window, click **Save**.



- In the SCA Modules window, click the **Save** button.



- Switch to the Events page of the test client. Click Invoke  on the upper left side of the page.
- Keep AAA as the symbol value, and enter CustomerB, the premium service customer, as the customerID value. Click **Continue**

The results in the Events area show DelayedService as the invoked service.

## Events

Events tree view showing a sequence of events for `StockQuote_MediationFlow:getQuote`. The events are:

- Invoke (StockQuote\_MediationFlow:getQuote)
  - Invoke started
  - Invoke returned
- Invoke (StockQuote\_MediationFlow:getQuote)
  - Invoke started
  - Invoke (StockQuote\_MediationFlow:getQuote)
    - Request (StockQuote\_MediationFlow --> DelayedService:getQuote) [highlighted]
    - Response (StockQuote\_MediationFlow <-- DelayedService:getQuote)
    - Return (StockQuote\_MediationFlow:getQuote)
  - Invoke returned

## General Properties

### Detailed Properties

Module: [StockQuote](#)  
Component: [StockQuote\\_MediationFlow](#)  
Interface: [StockQuoteService](#)  
Operation: [getQuote](#)

Return parameters:

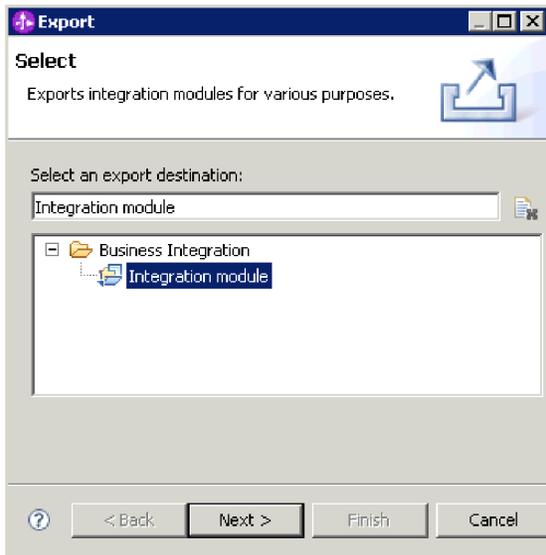
Name	T...	Value
response	Stoc...	✓
value	float	✓ 30.64931
qualityOfService	string	✓ premium

---

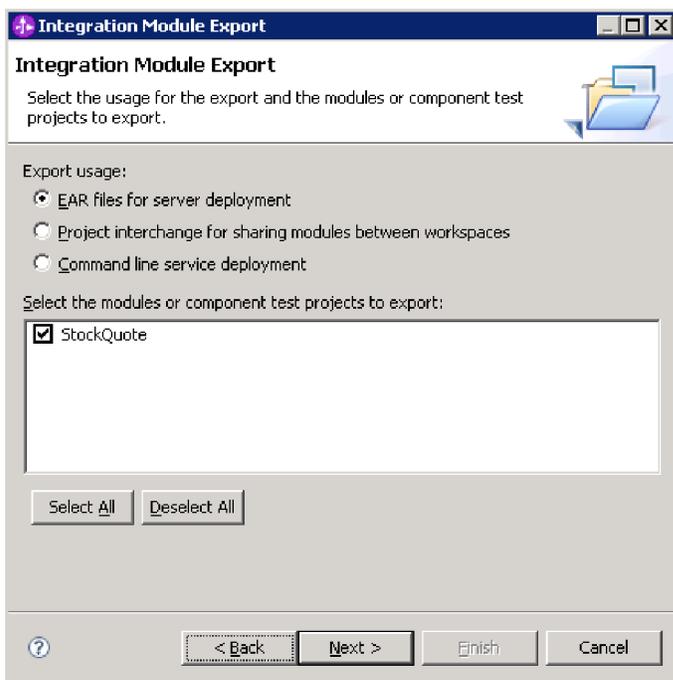
## Chapter 6. Deploy

Export the sample to the WebSphere ESB or WebSphere Process Server for deploying to the runtime.

1. Select File > Export and type **Integration module** in the text box. Select Integration module and click **Next**.



2. In the Integration module export panel, select EAR files for server deployment. Select the StockQuote module. Select **Next**. Specify a target directory and click **Finish**.



3. Install the exported EAR file on the WebSphere ESB or WebSphere Process Server.

The WebSphere ESB samples gallery ships the Stock Quote sample, along with a web client that you can use to test the sample that you built. For instructions see WebSphere ESB information centre - Samples



---

## Notices

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 document 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 document. The furnishing of this documentation does not grant 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 character set (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 106-0032, 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 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.

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

Licenseses 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 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 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.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples 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.

#### COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate 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. \_enter the year or years\_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

## **Programming interface information**

Programming interface information, if provided, 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**

IBM, the IBM logo, and [ibm.com](http://ibm.com) are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

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

Linux is a registered trademark of Linus Torvalds 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.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

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

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



---

## Terms of use

Permissions for the use of publications is granted subject to the following terms and conditions.

**Personal Use:** You may reproduce these publications for your personal, non commercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

**Commercial Use:** You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

© Copyright IBM Corporation 2005, 2008. All Rights Reserved.



---

## Readers' Comments — We'd Like to Hear from You

Integration Developer  
Version 6.2  
Stock Quote Sample  
Version 6 Release 2

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Comments:

Thank you for your support.

Send your comments to the address on the reverse side of this form.

If you would like a response from IBM, please fill in the following information:

\_\_\_\_\_

Name

\_\_\_\_\_

Address

\_\_\_\_\_

Company or Organization

\_\_\_\_\_

Phone No.

\_\_\_\_\_

E-mail address

**Readers' Comments — We'd Like to Hear from You**



Cut or Fold  
Along Line

Fold and Tape

**Please do not staple**

Fold and Tape

PLACE  
POSTAGE  
STAMP  
HERE

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

Fold and Tape

**Please do not staple**

Fold and Tape

Cut or Fold  
Along Line





Printed in Canada