

IBM Integration Designer
Version 8 Release 5

Stock Quote Sample



Note

Before using this information and the product it supports, read the information in Notices.

Contents

Chapter 1. StockQuote	1	Define the end points of the mediation	16
Chapter 2. Overview	3	Build the request flow	18
Chapter 3. Mediation design	5	Build the response flow	26
Chapter 4. Build it yourself	9	Chapter 5. Run the sample	31
Import resources	9	Install the runtime components.	31
Create the StockQuoteService interface	11	Test the mediation flow	33
Set up a business object for temporary data	13	Debug the mediation flow	35
Create the mediation module	13	Change the quality of service at run time	38
Assemble the mediation module	14	Notices	43
Implement the mediation.	16	Terms of use	47

Chapter 1. StockQuote

Advanced

The Stock Quote sample demonstrates how you can easily integrate different services and manipulate messages at run time without affecting the client applications. This integration is achieved through the Service Component Architecture and mediation functionality provided by IBM® Process Server. 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.

Prerequisites

IBM Integration Designer with a IBM Process Server unit test environment.

Learning objectives

This sample shows you how to do the following:

- Build the Stock Quote application using web services and mediation flows
- Test and debug the application using a provided client and back-end
- Redirect a service request from the administrative console while the application is running
- Run the application in the unit test environment

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

Chapter 2. Overview

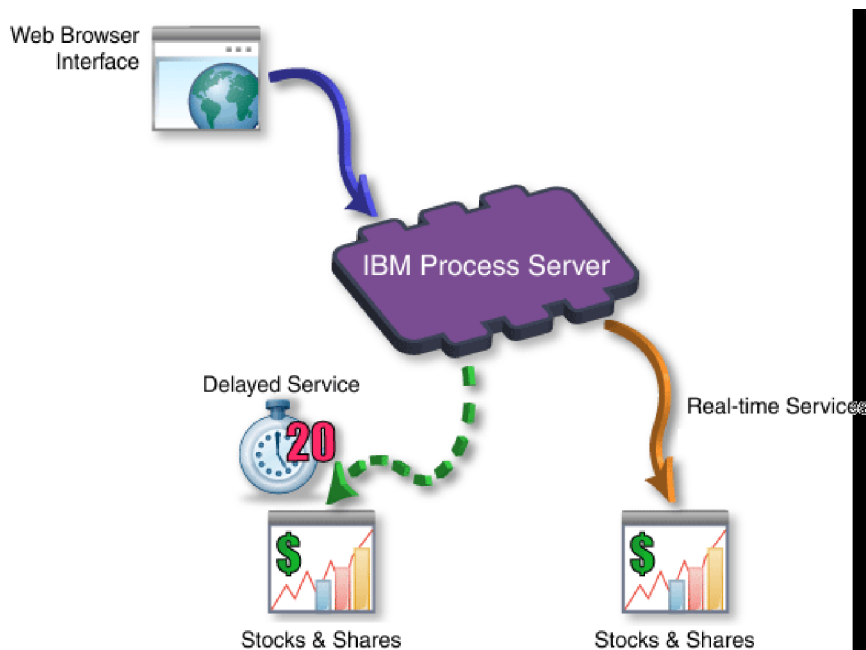
Advanced

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.

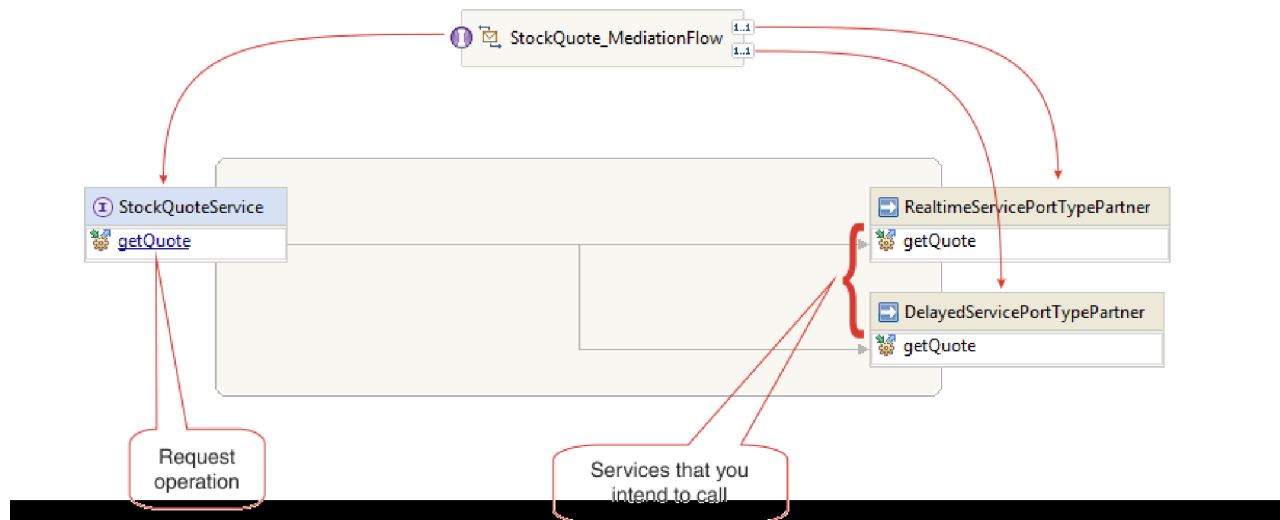


Chapter 3. Mediation design

Advanced

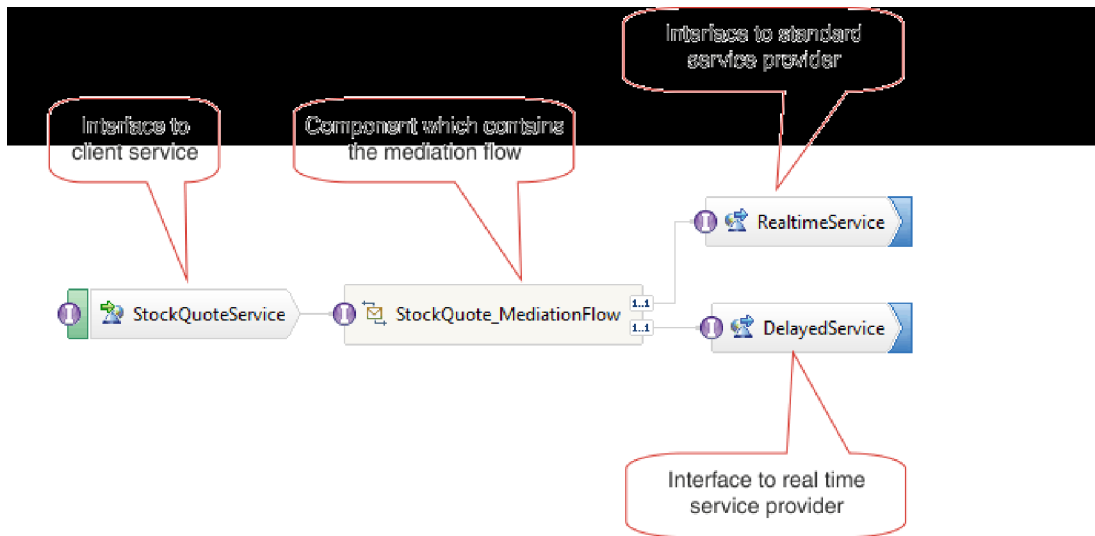
The mediation service that runs on the IBM Process Server is contained in a single *mediation module* called StockQuote.

The mediation module consists of: an *export* that 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:

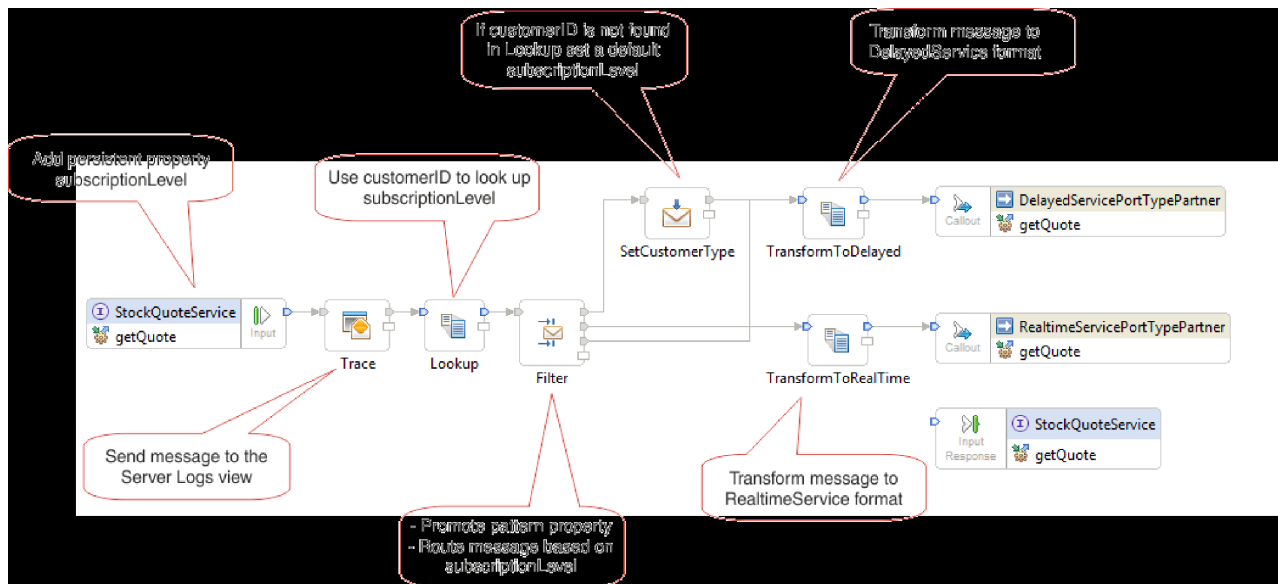


The StockQuote mediation module consists of the following elements:

- **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.
- **DelayedService** has a web service binding and an interface that matches the delayed (standard) service.

StockQuote_MediationFlow

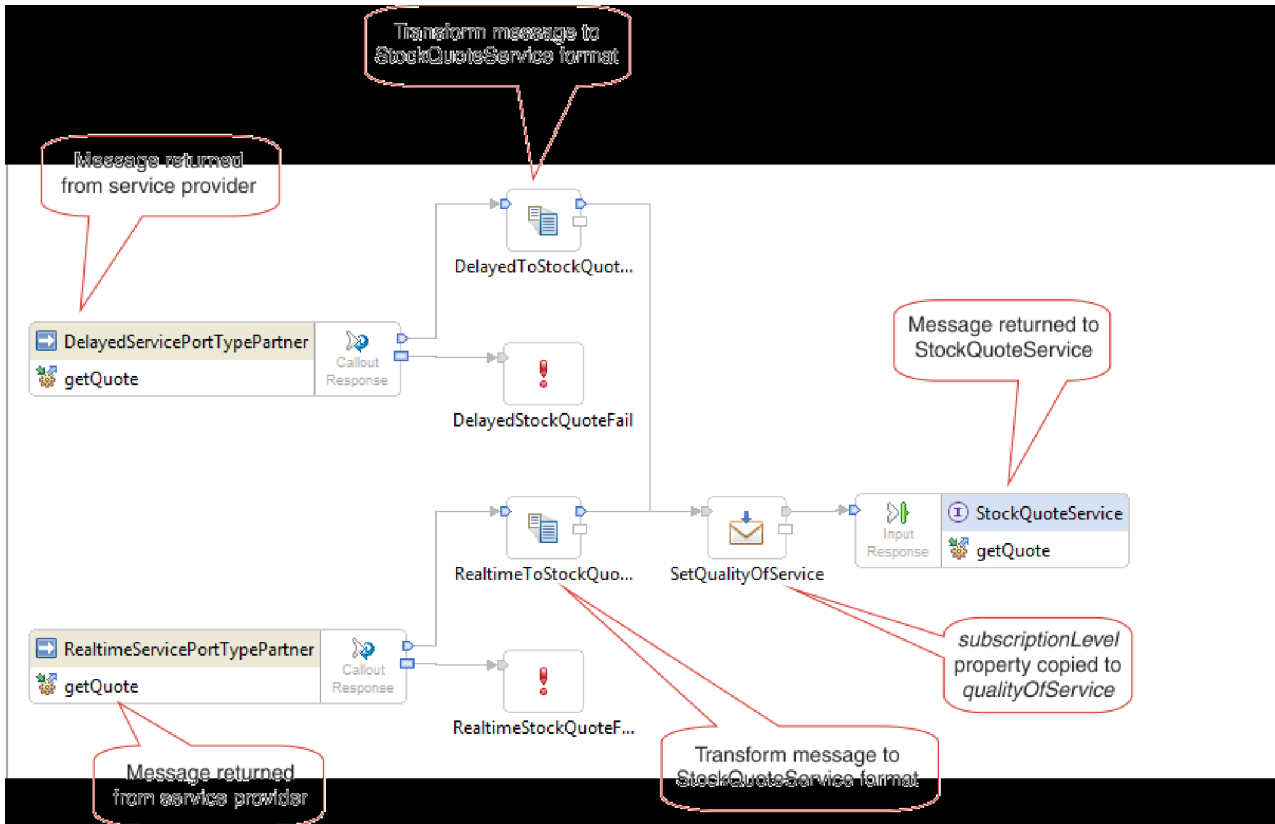
The diagram below shows the request flow that defines the mediation logic applied to the message as it flows through the StockQuote_MediationFlow component to the target service providers.



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 message is output to the server log view using the Trace mediation primitive..
3. A **Map** 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 `CustomerType.csv` file. 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 run time to redirect the stock quote request to the delayed service if the premium service is unavailable.
5. If the `customerID` was not found in the `CustomerType.csv` file, then a default `subscriptionLevel` is set in the `SetCutomerType` using a Message Element Setter.
6. The message is transformed on the way to either service by XSLT primitives `TransformToDelayed` and `TransformToRealtime` so that it matches what the service expects.
7. 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.



Lessons in this Module

Build it yourself

Build the sample, and then test it.

Implement the mediation

Select the sending and receiving operations. 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.

Run the sample

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

Chapter 4. Build it yourself

Advanced

Build the sample, and then test it.

Start building the Stock Quote sample by importing the starter artifacts and creating the mediation module for your sample. Resources such as interfaces and business objects can be contained in the Resources library 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. Import the start artifacts (Resources library and StockQuoteProvider service)
2. Create the mediation module
3. Assemble the mediation module
4. Implement the mediation

Lessons in this module

Import resources

To build this sample, you need the DelayedStockQuote and RealtimeStockQuote web services. These are presupplied for you in the StockQuoteProvider module, along with a Resources library. The Resources library provides the web services interfaces, the StockQuoteService interface for connecting a web client to your mediation module, and the business objects for storing the subscription level. The subscription level value is used in the mediation for routing.

Create the mediation module

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

Assemble the mediation module

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

Implement the mediation

Select the sending and receiving operations. 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.

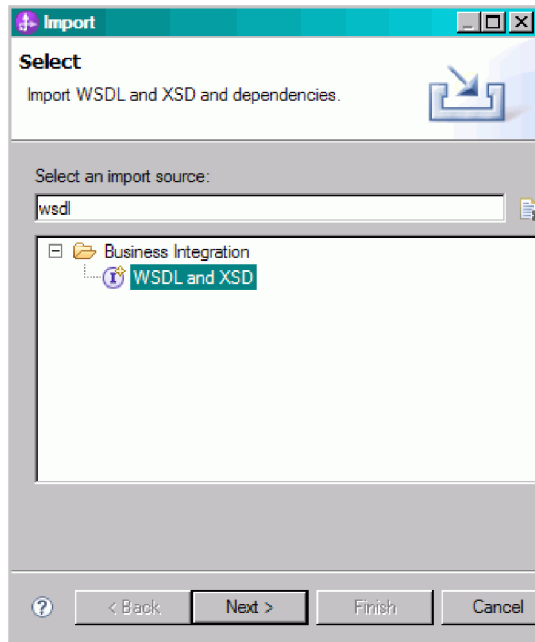
Import resources

Advanced

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.

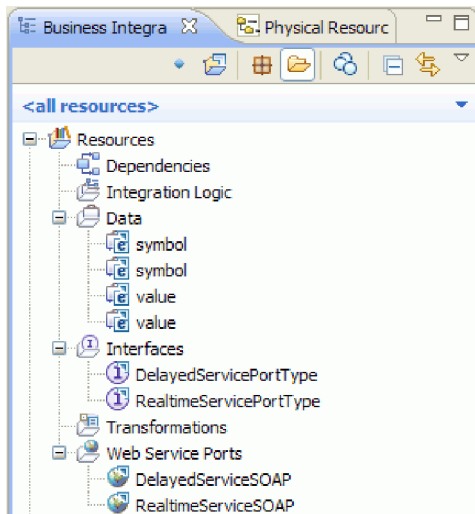
To create the **Resources** library and import the ready-made WSDL files into it, complete the following steps.

1. In the Business Integration view, under Projects click **New...** Select **Library** as the type of project and click **Next**. For the Library name specify Resources. Click **Finish**. If you already have projects in the Business Integration view, then right-click a project and select **New > Project > Library**. Name it Resources. Click **Finish**.
2. Right-click the **Resources** project and select **Import**. In the Select an import source text box, type *wsdl* and select **WSDL and XSD**. Click **Next**.



3. Choose the option to import the wsdl locally and click **Next**.
4. In the source directory field, browse to the directory where you installed shared resources (the default directory is Shared_Resources). Navigate to *Installed_Shared_Resources_Directory/plugins/com.ibm.wbit.samples.content_qualifier/artifacts/stockquote/wsdl* (where *qualifier* is a string that is appended to the plugin name and indicates the version number and time stamp, for example, 7.0.0.v20091113_0200). Click **OK**.
5. In the Import wizard, select both WSDL files: **DelayedService.wsdl** and **RealtimeService.wsdl**.
6. The target module is **Resources**.
7. 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.



Create the StockQuoteService interface



Advanced

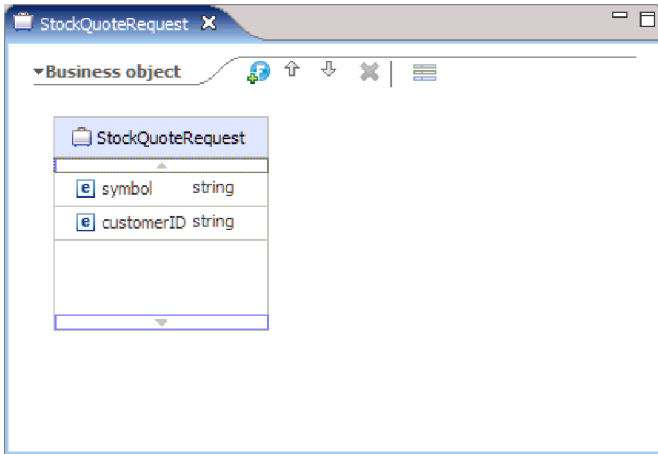
Create the StockQuoteService interface. This is the interface that you can use to connect your web client to the mediation module.


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.

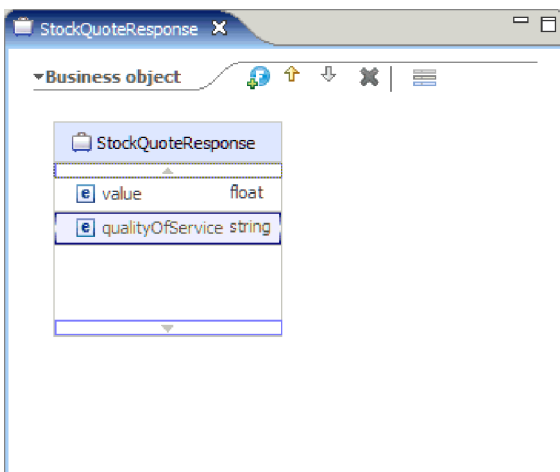
The `getQuote` operation will be used to send the request for a stock quote. The operation will send the request data as a business object named `StockQuoteRequest` that contains the fields `symbol` and `customerID`. The operation will receive the response data as business object named `StockQuoteResponse`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

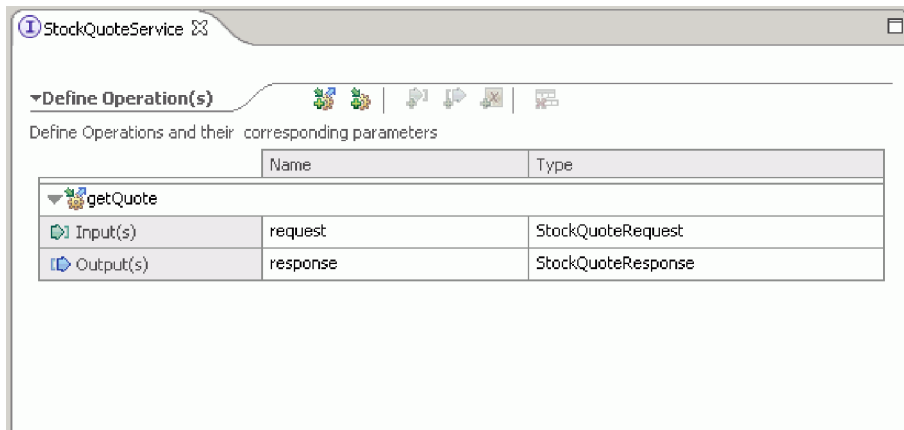


6. Right-click **Data**, and choose **New > Business Object** to create another business object.
7. In the New Business Object Wizard, type StockQuoteResponse. Click **Finish**.
8. In the Business Object Editor, click the **Add a field to a business object** button . An attribute named *field1* is created. 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 the request input's type ("string") and in the resulting Data Type selection window, 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.



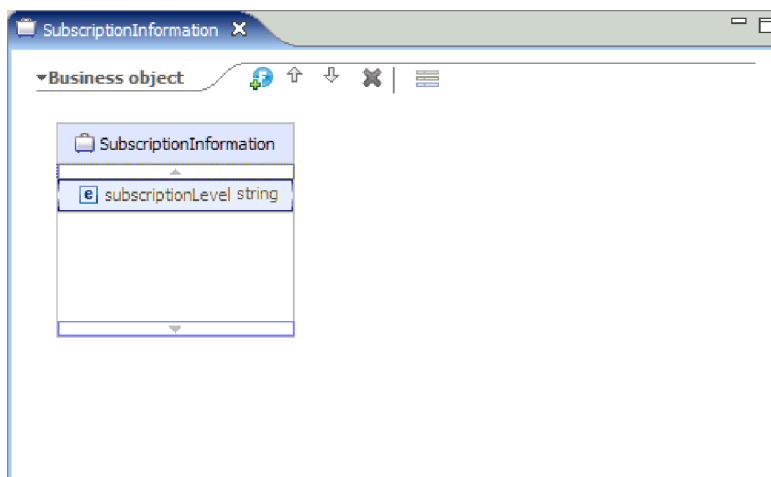
Set up a business object for temporary data

Advanced

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. You will create a business object to contain the subscription level, and then set the value of the subscription level when you 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.

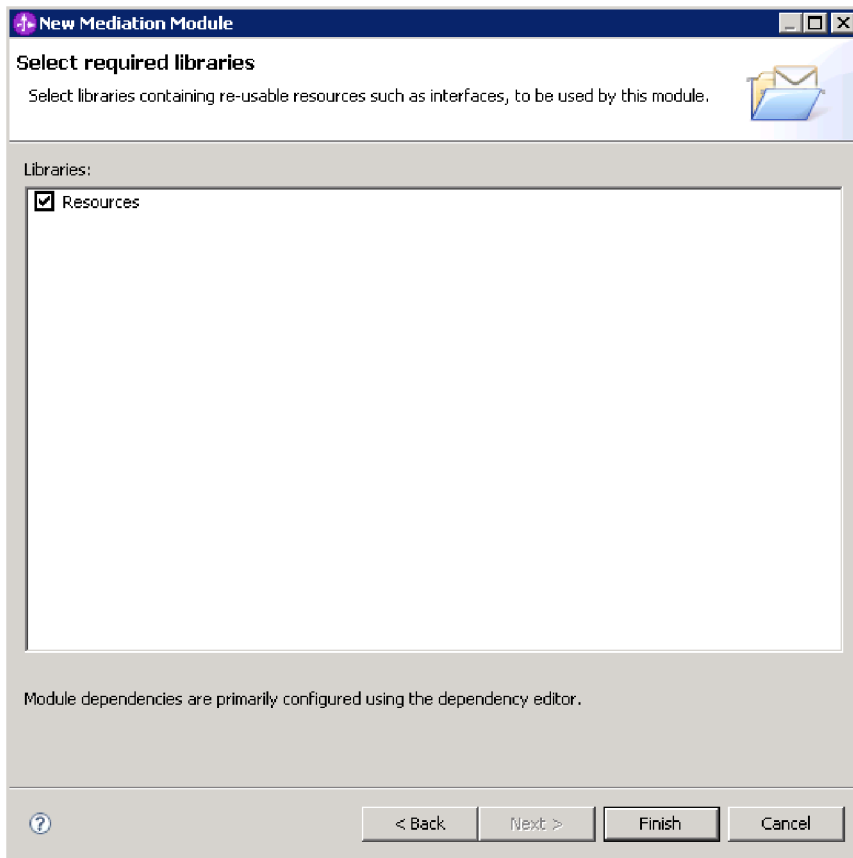
Create the mediation module

Advanced

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 > Project > Mediation Module**. The New Mediation Module window opens.
2. In the **Module Name** field, type StockQuote.
3. Keep the **Create mediation component** box checked.
4. Click **Next**.
5. 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.



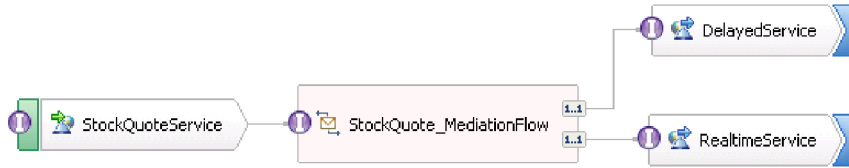
A mediation module called StockQuote is created, and contains a dependency on the Resources library. A mediation flow component called StockQuote is created in the module's assembly diagram.

Assemble the mediation module

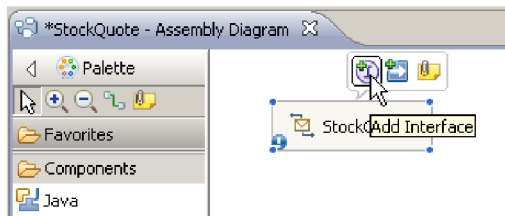
Advanced

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

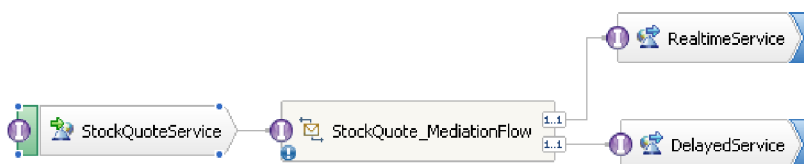
To build the StockQuote mediation module assembly diagram, as shown in the following diagram, complete the following steps:



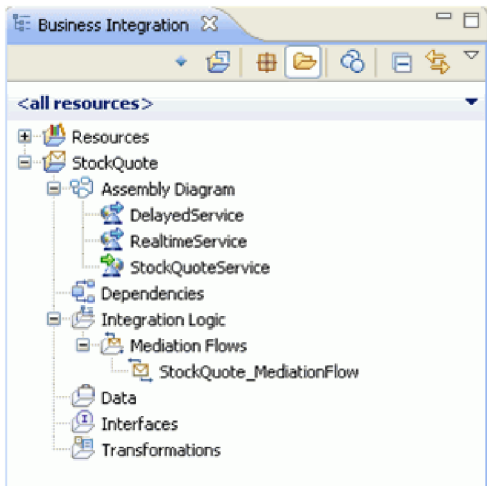
1. In the Business Integration view, expand the **StockQuote** module.
2. To open the assembly editor, double-click **Assembly Diagram**. The assembly editor opens showing the StockQuote mediation flow component.
3. Select the StockQuote mediation flow component and hover over it. The select **Add Interface > StockQuoteService Interface**.



4. Right-click the StockQuote mediation flow component, and select **Generate Export > Web Service Binding**.
5. In the Configure Web Service Export wizard, select the SOAP 1.1/JMS option, and then click **Finish**.
6. Right-click **StockQuoteServiceExport1**, and then select **Refactor > Rename**. You need to save the StockQuote Assembly diagram. Change the name to StockQuoteService in the Rename Artifact window.
7. Locate the generated WSDL port, **StockQuoteService_StockQuoteServiceJms Port** under the resource library's Web Service Ports category.
8. In the Resources library's Interfaces category select DelayedServicePort and drag it onto the assembly editor canvas. In the Configure Web Service Import window, select **Import with Web Service Binding** and click **OK**.
9. Select the import and use refactoring to rename it to DelayedService. Click the Binding tab in the Properties view to see the binding information.
10. Drag the RealtimeServicePort artifact onto the assembly editor canvas, and then select **Import with Web Service Binding**.
11. Select the import and use refactoring to rename the import to RealtimeService.
12. Click StockQuote. This is the mediation flow component that was created with the mediation module. Use refactoring to rename it to StockQuote_MediationFlow
13. 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.
14. 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.



15. Generate the implementation for StockQuote_MediationFlow. In the assembly editor, select the component, and then 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.
16. Save the assembly diagram. Expand the StockQuote assembly under the module in the Business Integration view to see the artifacts created.



Implement the mediation

Advanced

Implement the mediation flow using the instructions in this topic.

Select the **StockQuoteService getQuote** operation, and select to create a **Service Integration**. Select to add references to the **DelayedServicePortTypePartner getQuote** and **RealtimeServicePortTypePartner getQuote** operations. These will be end points of the mediation flow. Do the following:

- Add a mediation primitive between the end points to log the message.
- Retrieve the customer's subscription level from the CustomerType csv file.
- Conditionally route the message based on the value retrieved.
- Transform the message to match the format of the receiving operation.

Lessons in this module

Define the end points of the mediation

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

Build the request flow

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

Build the response flow

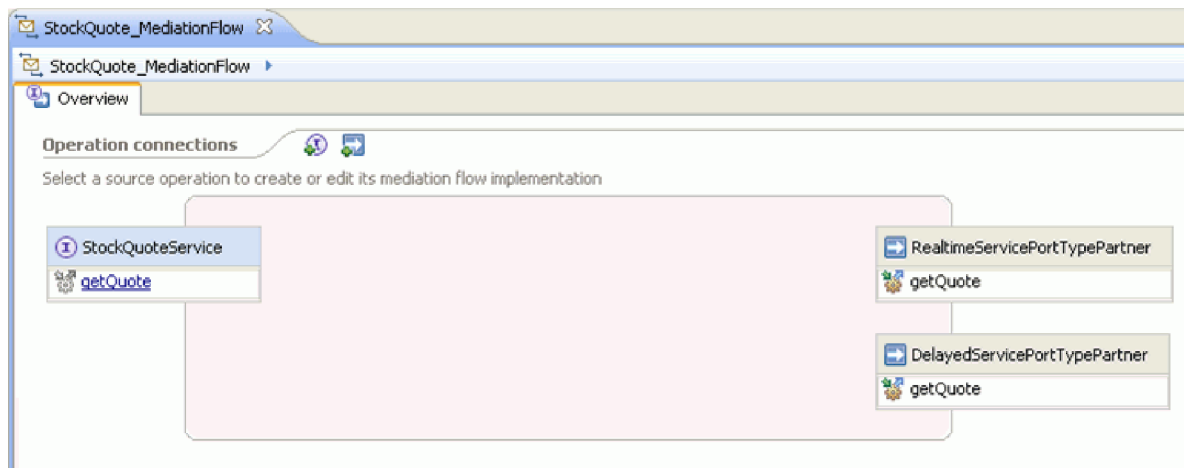
Build the response flow for the getQuote operation: create the mediation primitives, set their properties, and wire the flow.

Define the end points of the mediation

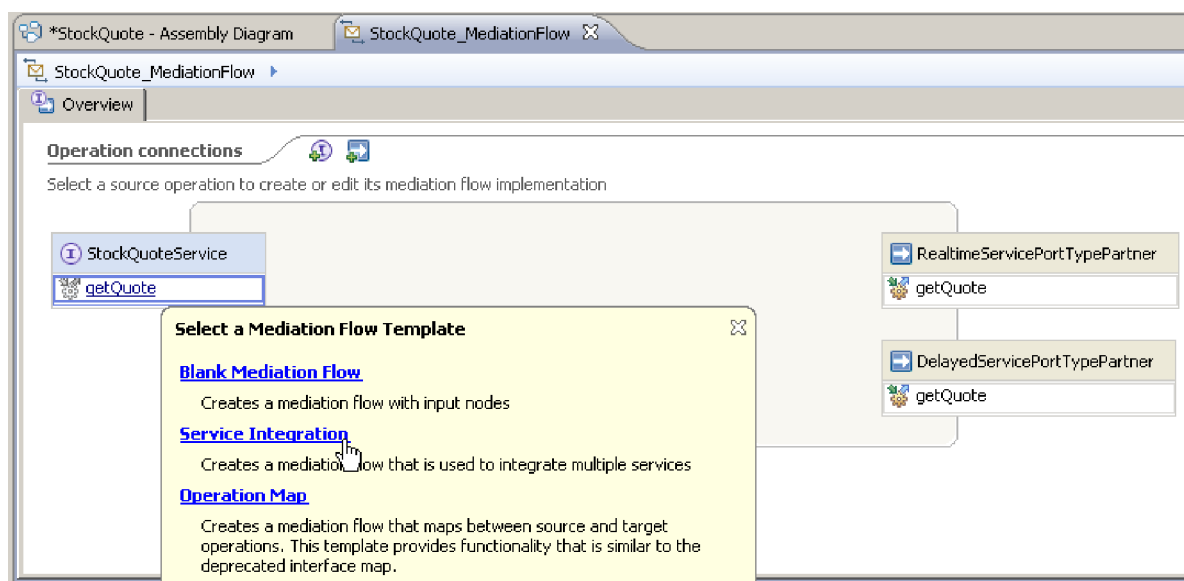
Advanced

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

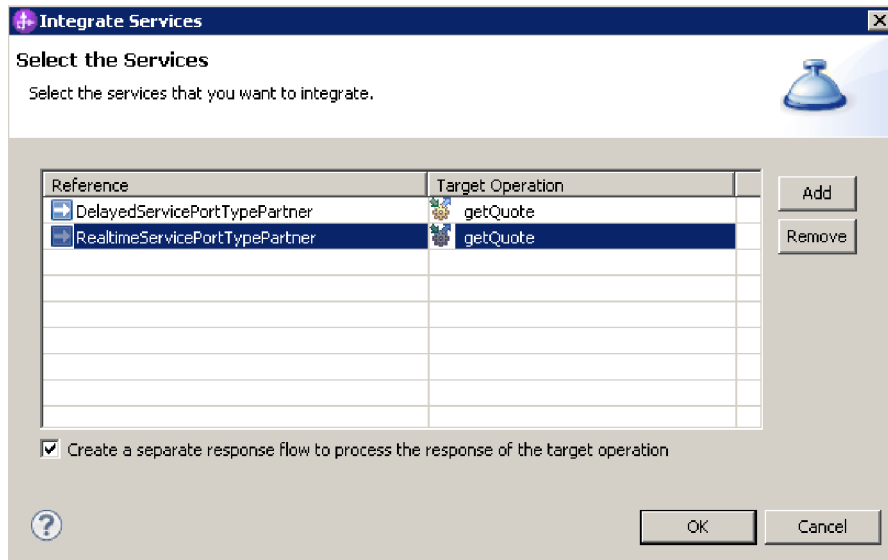
1. Double-click the mediation flow component to open the Mediation Flow Editor.



2. In the Operation connections section, click the getQuote operation of the StockQuoteService interface.
3. In the template selection window, choose the service integration template.



4. In the service selection window, choose the target operations that you want to invoke. Click **Add**.



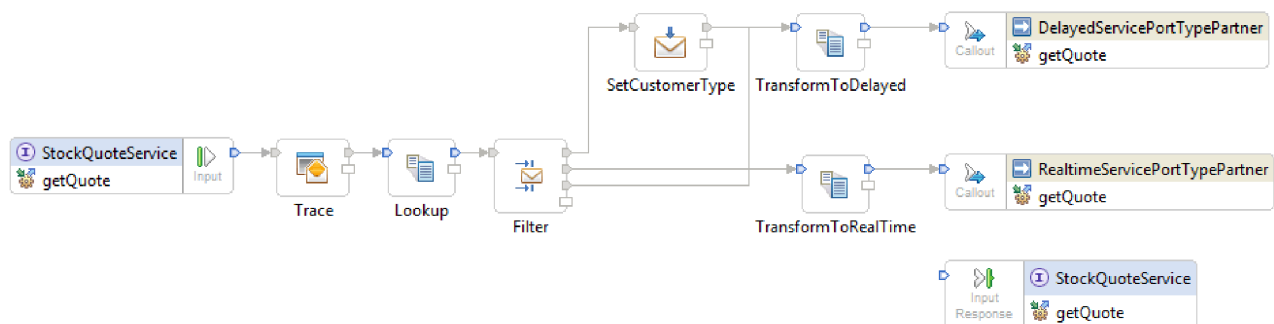
5. Select **DelayedServicePortTypePartner**, and select the **getQuote** operation.
6. Add the **getQuote** operation of the **RealtimeServicePortTypePartner**.
7. Keep the default setting to create a separate response flow.
8. Click **OK**. The mediation flow editor opens. You can now proceed to build the request flow.

Build the request flow

Advanced

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

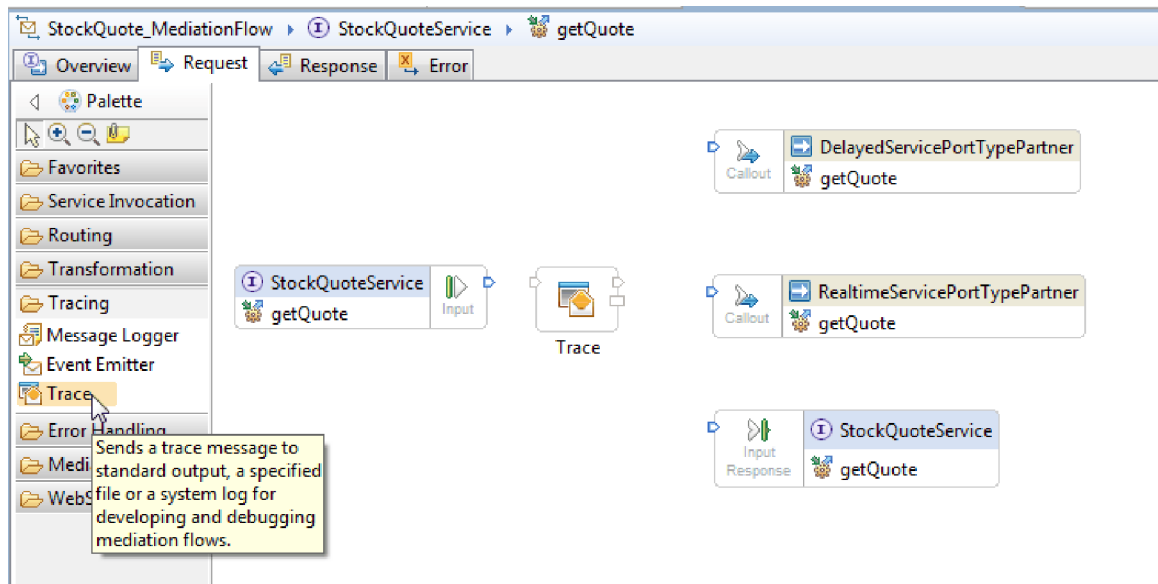
In the mediation flow editor, the request flow is represented from left to right. You will see an input node on the left. This is where the request message enters the flow. On the right are two callout nodes, one for each target operation. You will also see an Input Response. This node is used when a message is to be returned to the client after the flow is executed. This node is not used 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:

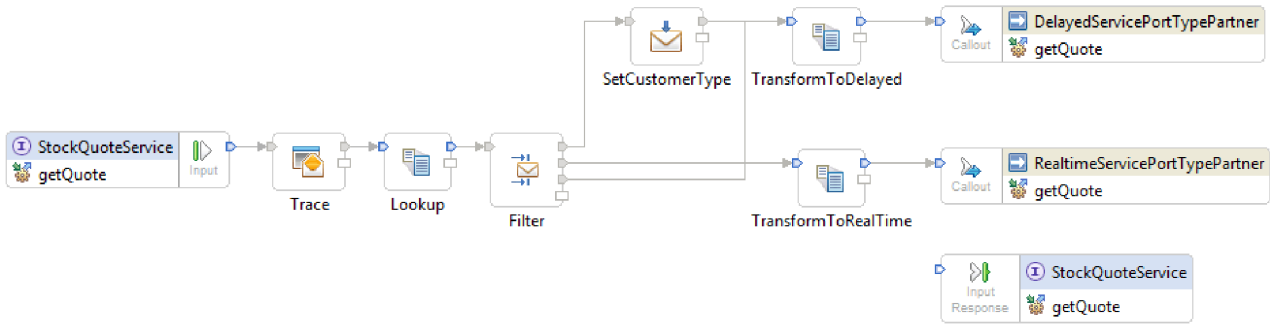
1. When the mediation flow opens, you will see a tip explaining how to invoke a service. Since you have already added a callout to the target service using the Service Integration template, you can close the tip.

- Click the Tracing palette category to expand the group.
- Click the **Trace** primitive and drop it onto the request flow canvas, and rename the primitive Trace.

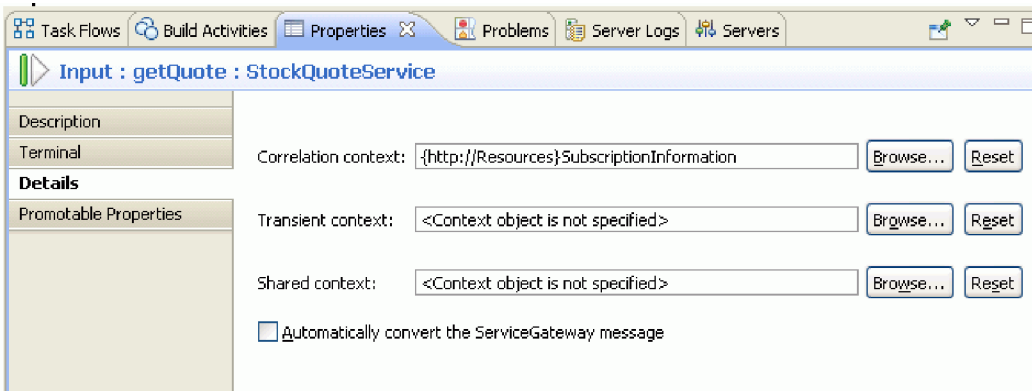


- Select a **Mapping** primitive from the Transformation folder in the palette, drop it onto the request flow canvas, and name it Lookup.
- Select a **Message Filter** primitive from the Routing folder, drop it onto the request flow canvas, and name it Filter.
- Select a **Mapping** primitive from the Transformation folder, drop it onto the request flow canvas, and name it TransformToDelayed.
- Select another **Mapping** primitive, drop it onto the request flow canvas, and name it TransformToRealtime.
- Select a **Message Element Setter** primitive from the Transformation folder, drop it onto the request flow canvas, and name it SetCustomerType.
- Before you proceed to wire the primitives, right click the canvas and make sure that the **Automatic Layout** option is on.
- In the request flow canvas, wire the primitives:
 - The output terminal of **getQuote : StockQuoteService** to the input terminal of **Trace**
 - The output terminal of **Trace** 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 **SetCustomerType**
 - The match1 terminal of **Filter** to the input terminal of **TransformToRealtime**
 - The output terminal of **SetCustomerType** to the input terminal of **TransformToDelayed**
 - The output terminal of **TransformToRealtime** to the input terminal of **getQuote : RealtimeServicePortTypePartner**
 - Select Filter. In the Properties view, select the Terminal tab. Right-click the Output terminal and select **Add output terminal**. In the New Dynamic Terminal window, take the defaults of Terminal type match and change the Terminal name delayedTime. Select **OK**.
 - Wire the delayedTime output terminal of Filter to the input terminal of TransformToDelayed.

The wired request flow should look like this:



- You will now add the business object that was created earlier to the correlation context of the input node **getQuote : StockQuoteService**. This enables 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**. The URI, `{http://Resources}SubscriptionInformation`, now appears in **Correlation context**.



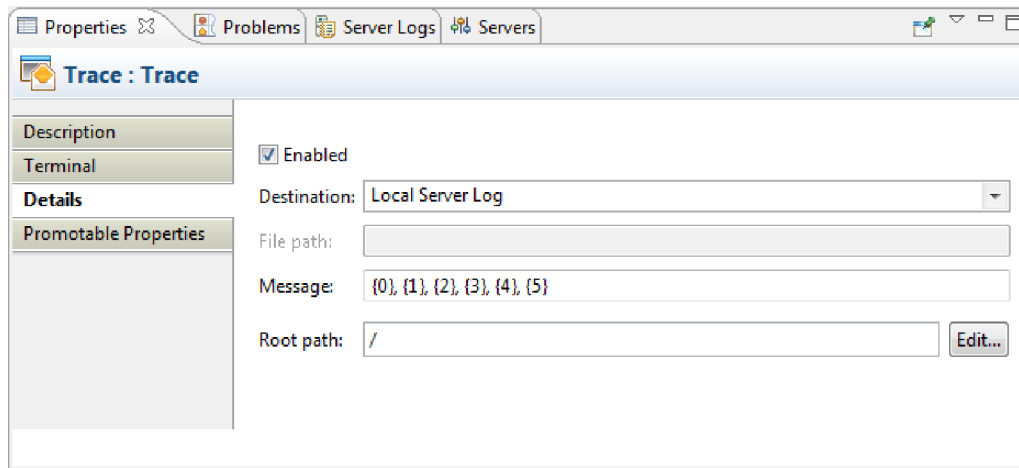
- Click **Trace** in the request flow canvas to see the primitive's properties in the Properties view. Click the Details tab to view the properties. Send the trace output to the Server Logs view. Ensure that these properties are set:

Table 1. Trace properties

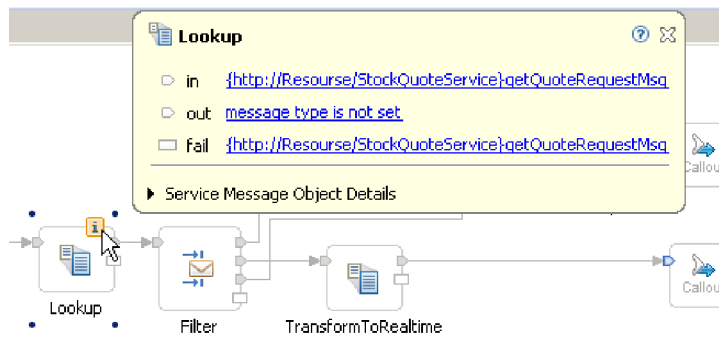
Property	Value	Description
Destination	Local Server Log	Outputs messages to the Server Logs view.

Table 1. Trace properties (continued)

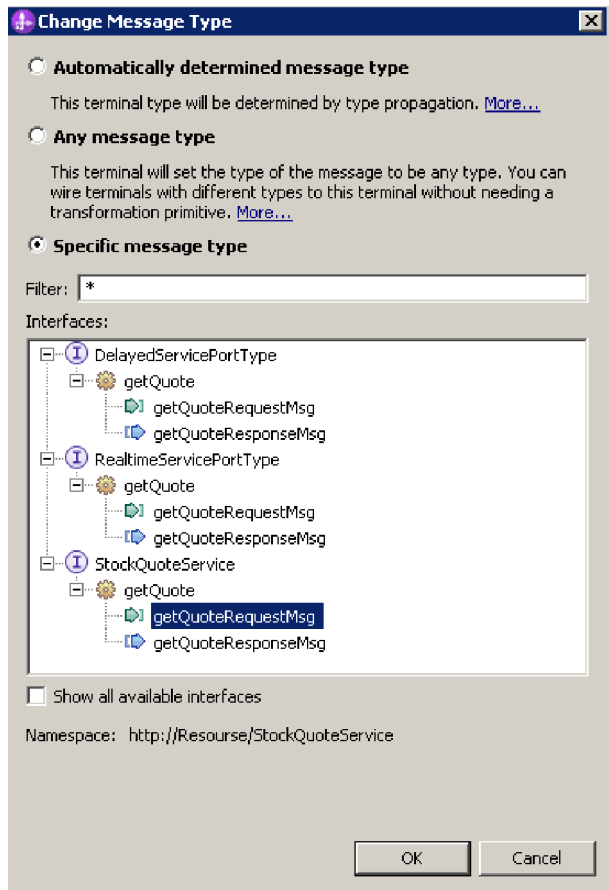
Property	Value	Description
Message	{0}, {1}, {2}, {3}, {4}, {5}	Includes the following information in the output: <ul style="list-style-type: none"> • {0} - the time stamps when the Trace primitive was invoked. • {1} - the message ID from the SMO. • {2} - the name of the Trace mediation primitive instance that generated the trace message. • {3} - t name of the module, containing the Trace mediation primitive instance, that generated the trace message. • {4} - the part of the SMO, as specified by the Root property XPath. • {5} - the version of the SMO.
Root path	/	The scope of the message and SMO to be inserted into the trace message at insert {4}.



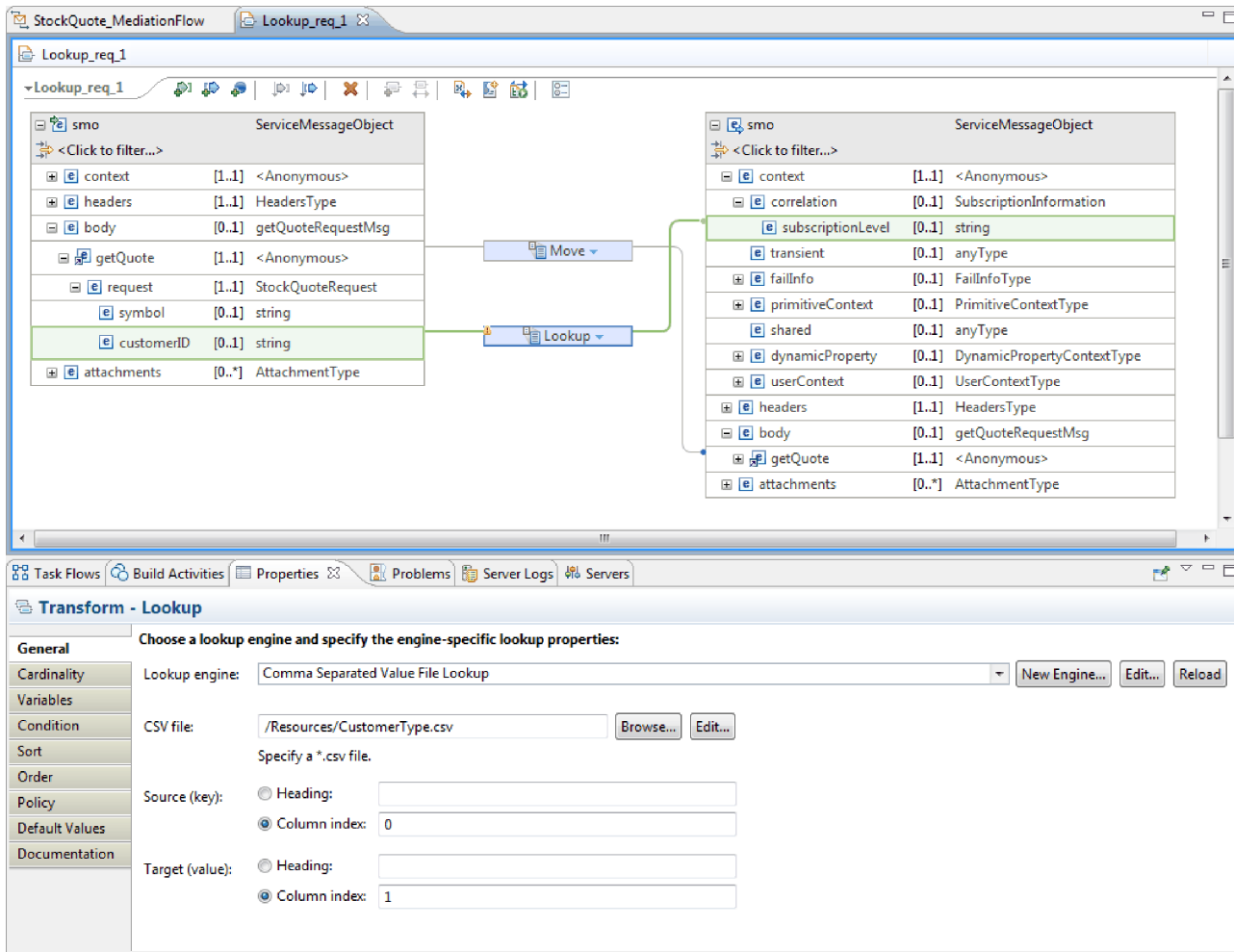
13. Select Lookup in the request flow canvas, hover over it and select .



Select the out message and set its type to StockQuoteService:getQuote:getQuoteRequestMsg.



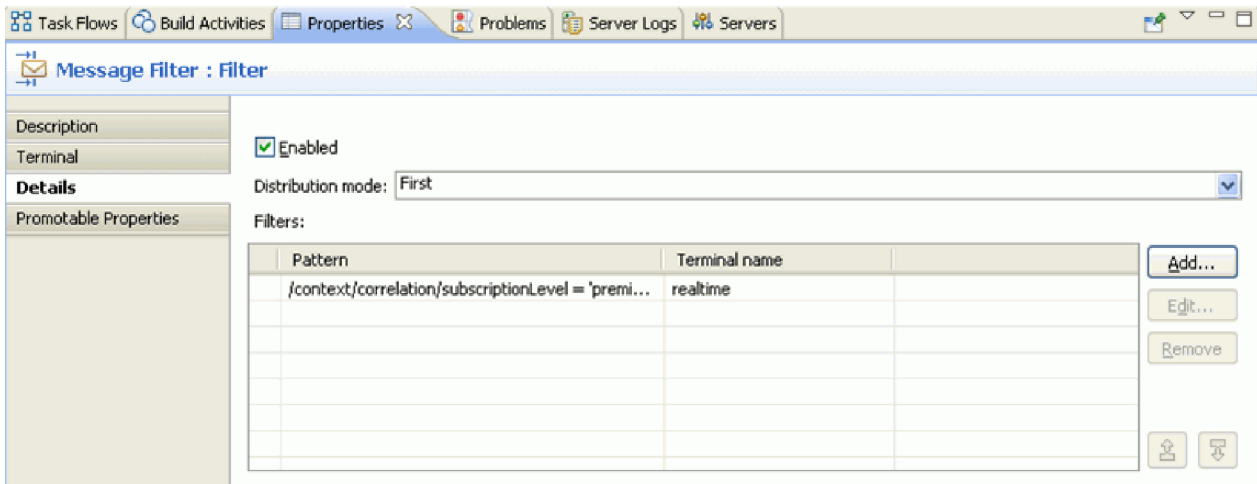
14. Double-click **Lookup**. In the New XML Map wizard click **Next**. Make sure the Message Root is "/" and then click **Finish**.



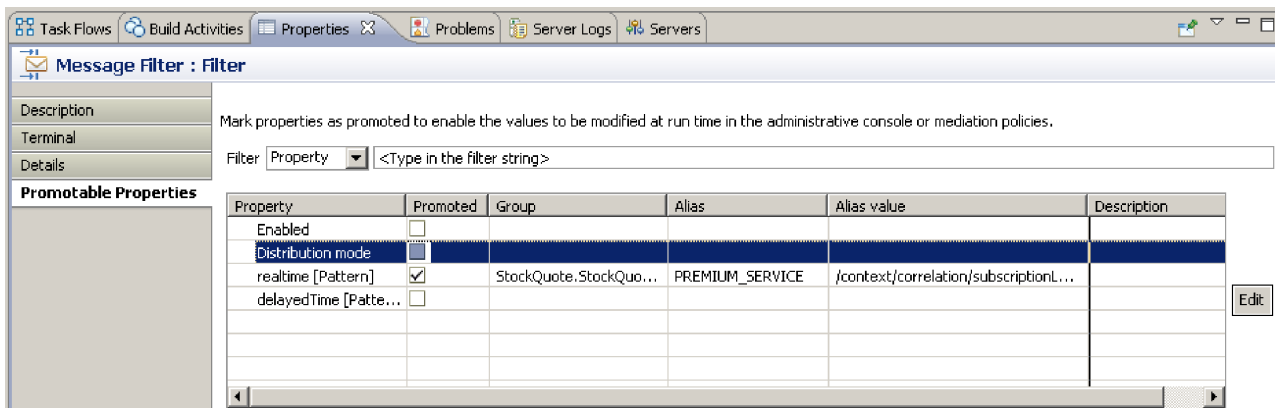
- a. In the map, wire the source body getQuote to the target body getQuote.
 - b. Fully expand the source getQuote. On the target, expand context > correlation. Wire the source customerID to the target subscriptionLevel. Change the transform to the **Lookup** transform. In the **Lookup** engine field, select **Comma Separated Value File Lookup**. For CSV file, browse and select CustomerType.csv from the Resources library. Click **Save** and then close the map.
15. Click **Filter** in the request flow canvas. Select the Terminal view:
 - a. Click **match1** in the **Output terminal** list. The properties of the match1 terminal appear on the right.
 - b. In the **Terminal name** field, change the name to realtime.
 16. By default, the message is sent to the TransformToDelayed primitive. You need to set the pattern for mapping to TransformToRealtime. Select the Details tab. In the Filter table, click **Add...** and enter the following values:

Table 2. Filter table properties

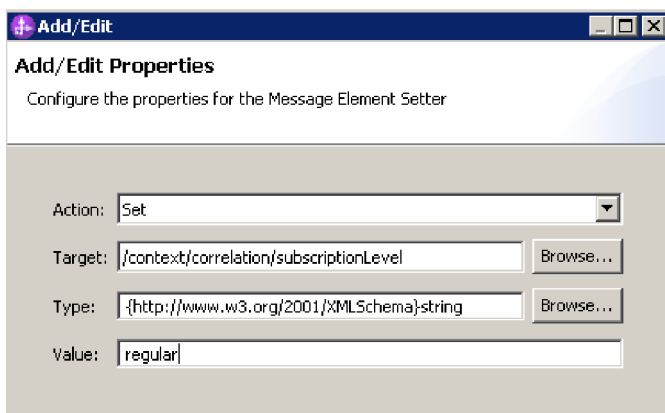
Column	Value
Pattern	/context/correlation/subscriptionLevel = 'premium'
Terminal name	realtime
Pattern	/context/correlation/subscriptionLevel = 'regular'
Terminal name	delayedTime



17. A promoted property can be changed by an administrator at run time. The pattern property can be changed at run time to change the quality of service. To promote the pattern property:
 - a. Click the Promotable properties tab.
 - b. Click the promoted check box of the realtime [Pattern] property.
 - c. Click the alias Filter.realtime. Type PREMIUM_SERVICE to rename the alias.

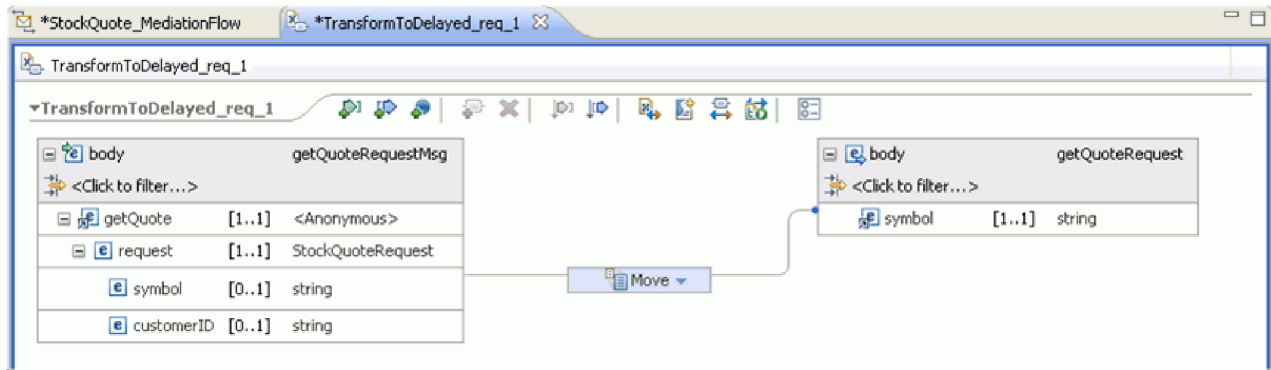


18. Select **SetCustomerType**. In the Properties view Details tab, select **Add**. Set the value regular into the correlation subscriptionLevel and then click **Save**.

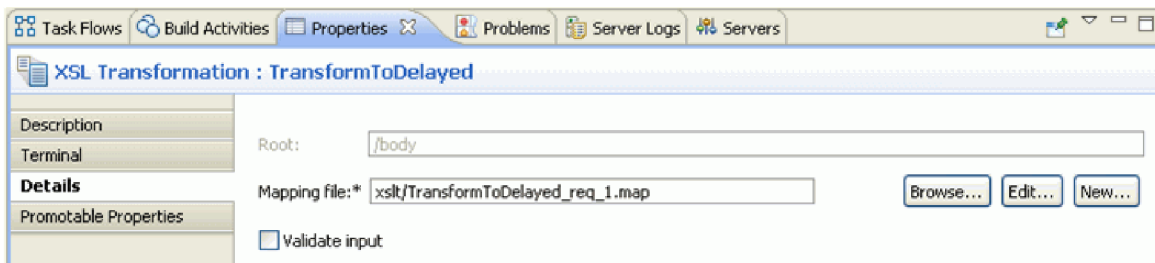


19. Set the properties for the Mapping primitive TransformToDelayed:

- 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 map 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.

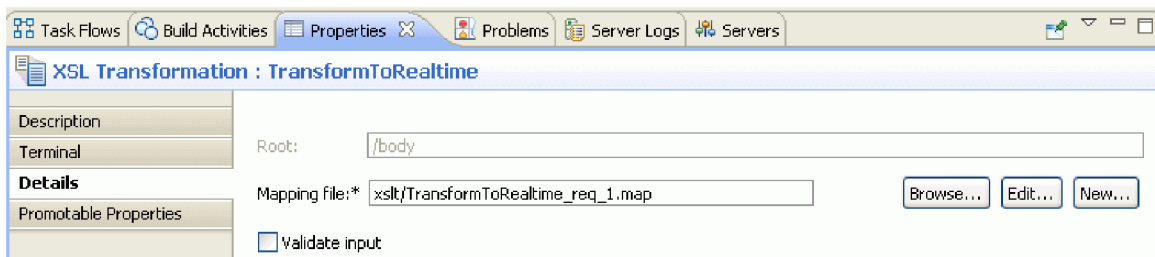


- d. Save your changes and close the map editor. The mapping file is displayed in the Details tab of the Properties view.

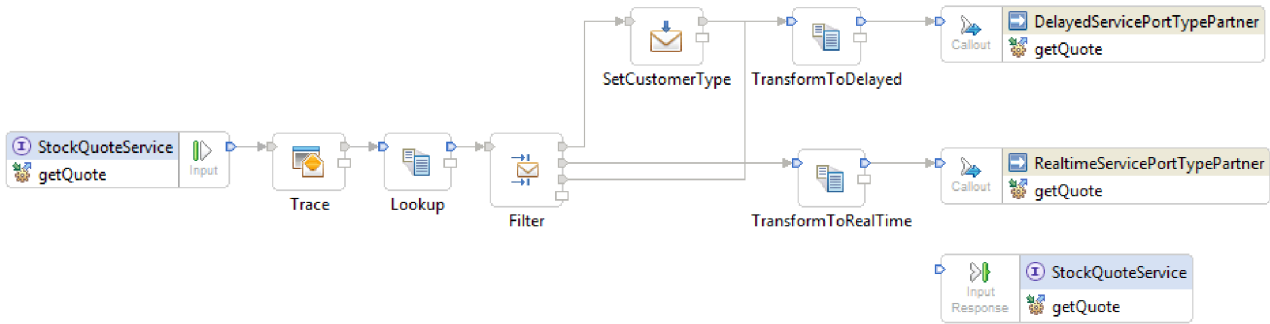


20. Similarly, set the properties for the **Mapping** primitive TransformToRealtime:

- 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 action opens the map 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 map editor. The mapping file is displayed in the Details tab of the Properties view.



21. Save the request flow.

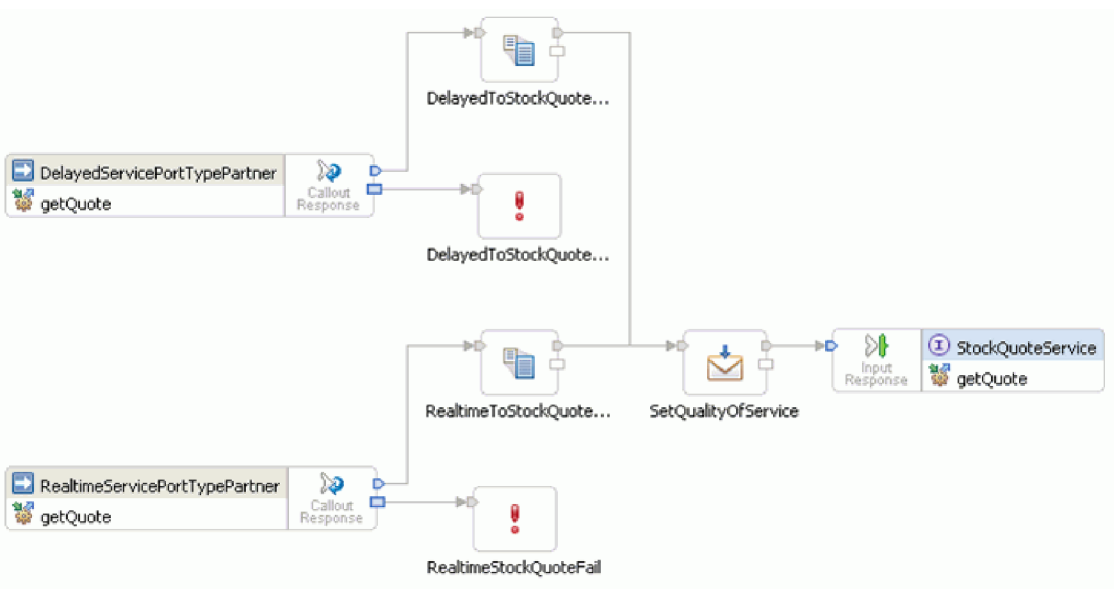


Build the response flow

Advanced

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 response nodes, one for each target operation; this is where the returning message enters the response flow. On the right is an input response 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:

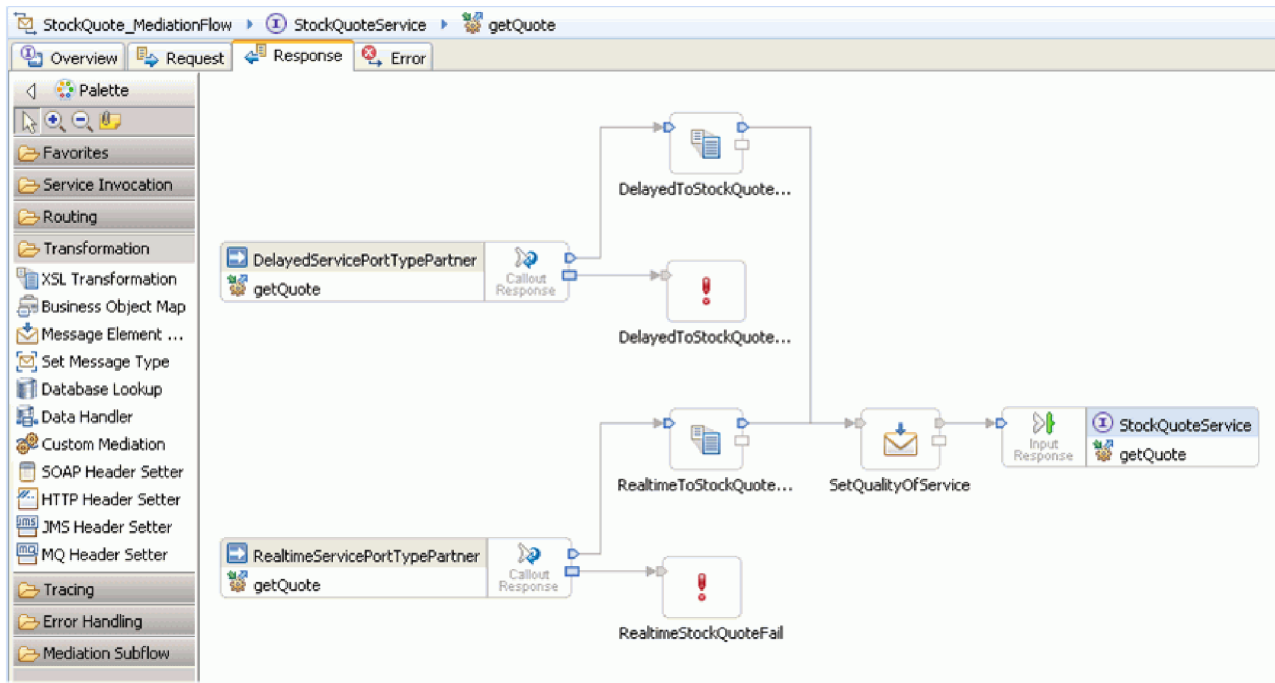


To build the response flow, complete the following steps:

1. Click the **Response** tab to view the response flow.
2. Select an **Mapping Transformation** primitive from the Transformation folder in the palette, drop it onto the response flow canvas, and rename it DelayedToStockQuoteService.
3. Select another **Mapping Transformation** primitive, drop it onto the response flow canvas, and rename it RealtimeToStockQuoteService.
4. Select a **Message Element Setter** primitive from the Transformation folder, drop it onto the canvas, and rename it SetQualityOfService.

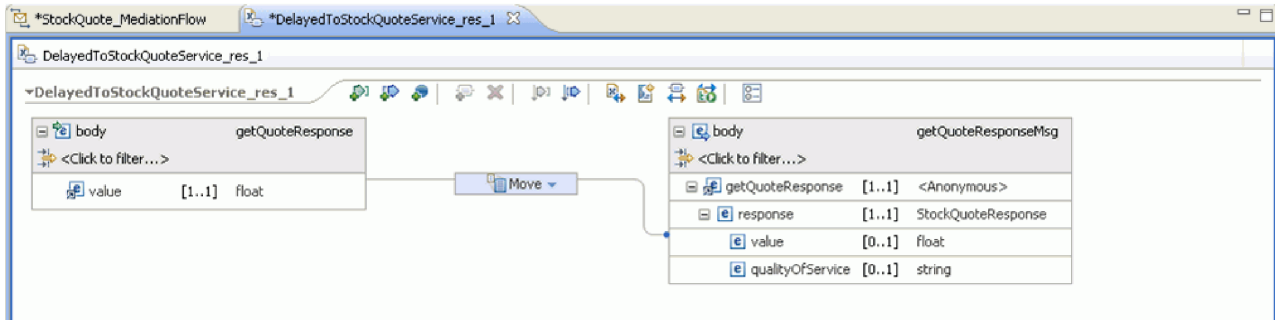
5. Select a **Fail** primitive from the Error Handling folder, drop it onto the canvas and rename it **DelayedStockQuoteFail**. Add another **Fail** primitive to the canvas and rename it **RealtimeStockQuoteFail**.
6. Before you proceed to wire the primitives, right click the canvas and make sure that the **Automatic Layout** option is on.
7. 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**

The wired response flow should look like this:

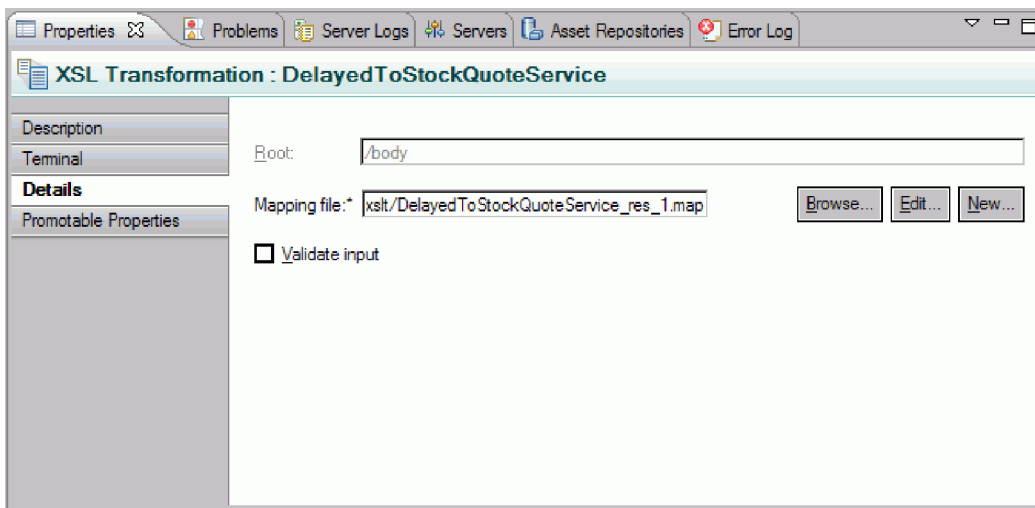


8. Set the properties for the XSLT primitive **DelayedToStockQuoteService**:
 - a. Select the **DelayedToStockQuoteService** primitive in the response flow canvas and double-click it.
 - b. The New XML Mapping wizard opens. Click **Next** to see the root, input, and output message types that will be mapped. Accept the defaults and click **Finish**.
 - c. In the input object section (left side) of the map 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.

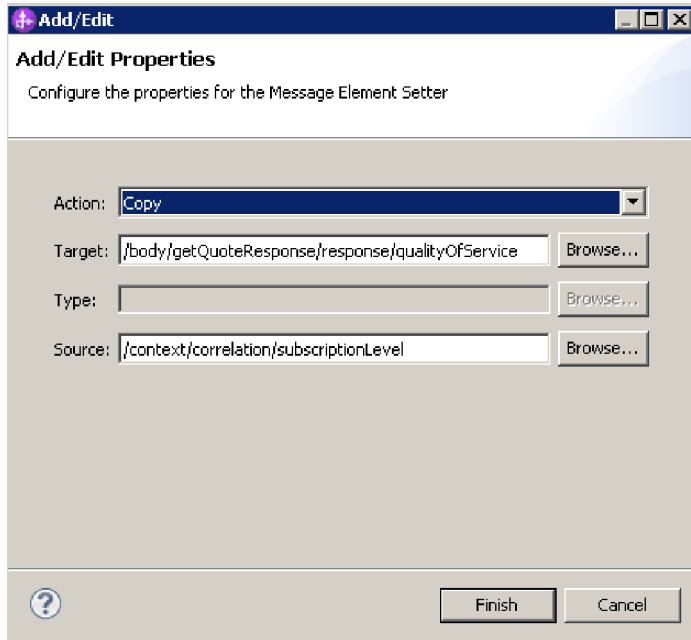


- e. Save your changes and close the map editor. The mapping file is displayed in the Details tab of the Properties view.

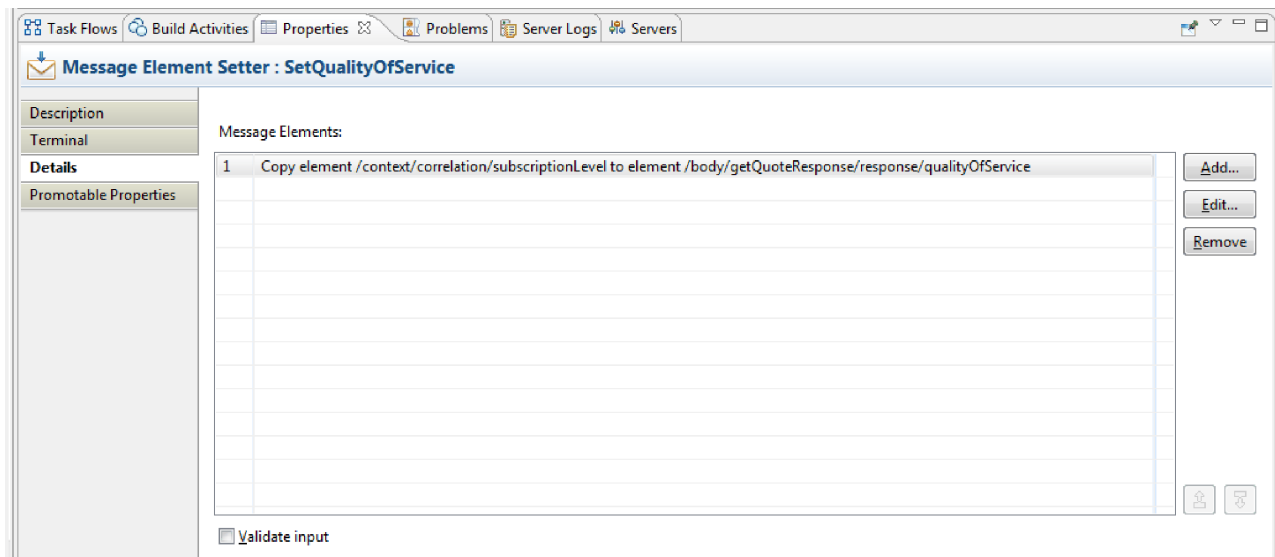


9. Similarly, set the properties for the XSLT primitive RealtimeToStockQuoteService:
 - a. Select the **RealtimeToStockQuoteService** primitive in the response flow canvas and double-click it.
 - b. The New XML Mapping wizard opens. Click **Next** to see the root, input, and output message types that will be mapped. Accept the defaults and click **Finish**.
 - c. In the input object section (left side) of the map 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 map editor. The mapping file and associated XSL style sheet are displayed in the Details tab of the Properties view.
10. 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 start the XPath Expression Builder.

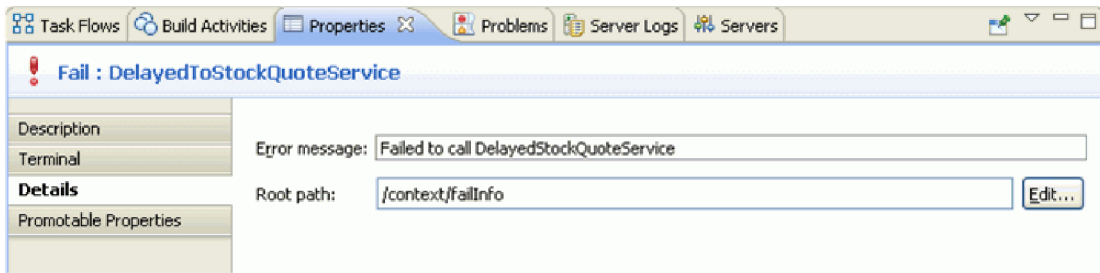
- d. In the Data Types Viewer, expand **ServiceMessageObject** > **body** > **getQuoteResponse** > **response : StockQuoteResponse**, and select **qualityOfService**. The XPath expression is shown in the expression field. Click **OK**.
- e. For the Source field select **Browse**. In the Data Types Viewer, expand **ServiceMessageObject** > **context** > **correlation** and select **subscriptionLevel**. The XPath expression is shown in the expression field. Click **OK**. In the Add/Edit window, click **Finish**.



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...**



11. 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.

12. Save the flow by pressing **Ctrl-S**.

Chapter 5. Run the sample

Advanced

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 csv file that has preloaded customer data. To run the sample, add the StockQuoteApp and the StockQuoteProviderApp projects to the server and launch the test client.

Lessons in this Module

Test the mediation flow

You have to add the StockQuoteApp and the StockQuoteProviderApp projects to the running server. Then, you can test your mediation flow in the test client.

Debug the mediation flow

Debug the StockQuote_MediationFlow component in the unit test environment by using the integration debugger.

Change the quality of service at run time

When you built the sample, you promoted the Filter pattern property, to allow you to change the value of the property at run time. Now, you 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. You will then view the result in the test client emulator.

Deploy to the runtime

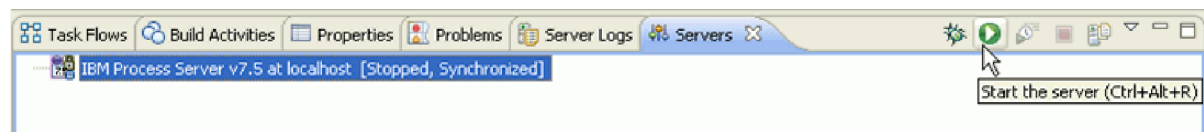
Now that your sample is tested, you may want to deploy it to IBM Process Server.

Install the runtime components

Advanced

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. Right-click the server and select **Start** to start the server.



Note: If you do not see a server in the Server view, it is likely that you did not install the IBM Process Server. Run the product launchpad to install the IBM Process Server.

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 the following command:

Installation Directory\runtimes\bi_version_number\bin\wsadmin -f wid-install.jacl -profileName wps_profile_name -userna

Keep the command prompt open as you will need it again.

Trouble shooting tips

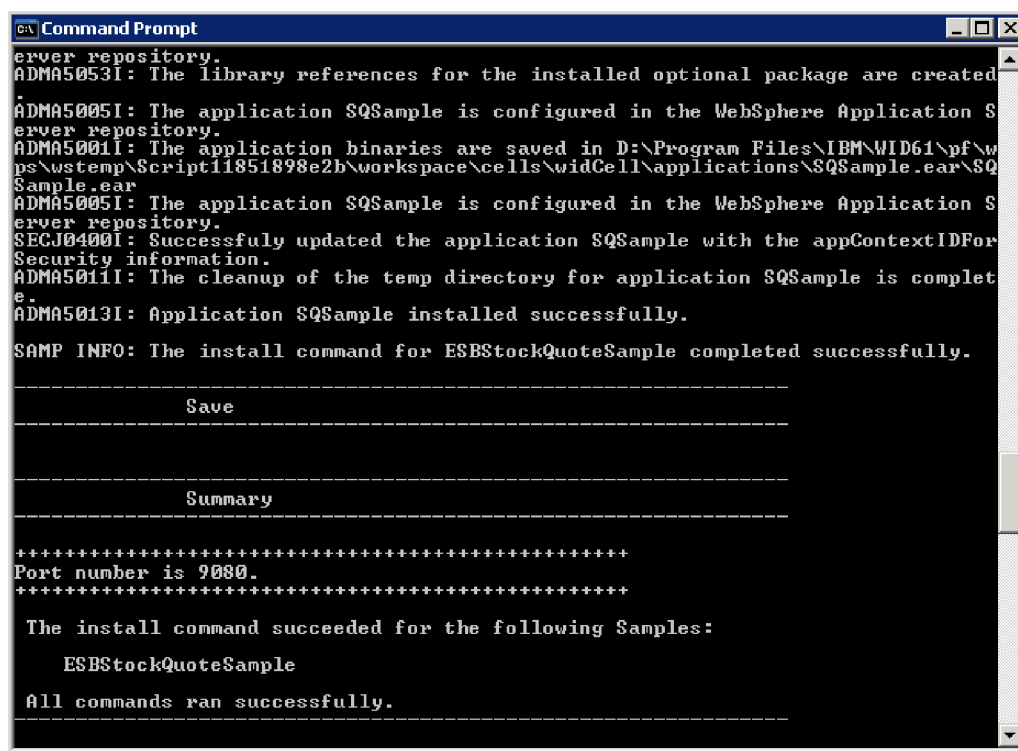
- If your installation directory has a space in it, you will see an error when you run the command. For example, if your installation directory is d:\Program Files, you will see this error:

```
'D:\Program' is not recognized as an internal or external command, operable program or batch file.
```

Put double quotation marks around the command, as follows:

```
"D:\Program Files\IBM\WID7_WTE\runtimes\bi_v7\bin\wsadmin" -f wid-install.jacl -profileName qwps -username admin -password admin
```

- The default user is "admin" and the default password is "admin".
 - The default IBM Process Server profile name is qwps.
 - To see the profile name of your server in IBM Integration Designer, switch to the servers view, right click the server and select **Manage Server Profiles**.
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.



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**.

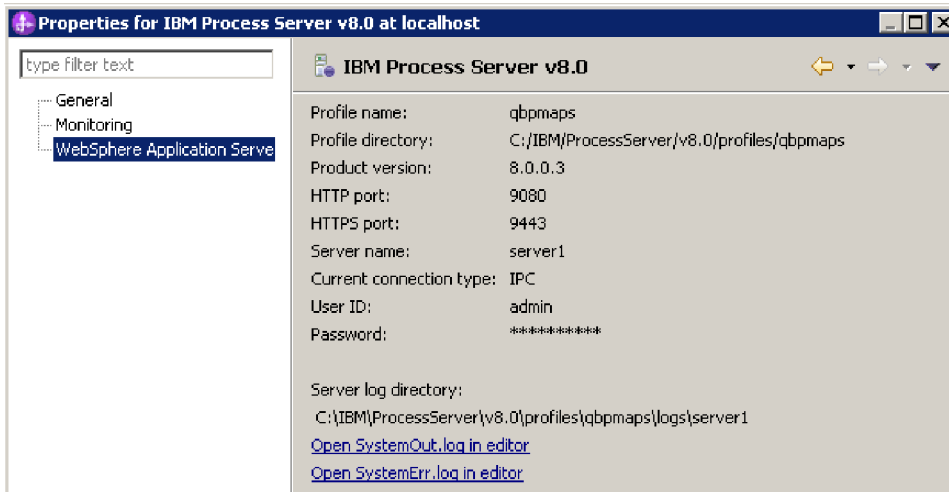
Test the mediation flow

Advanced

Test your mediation flow.

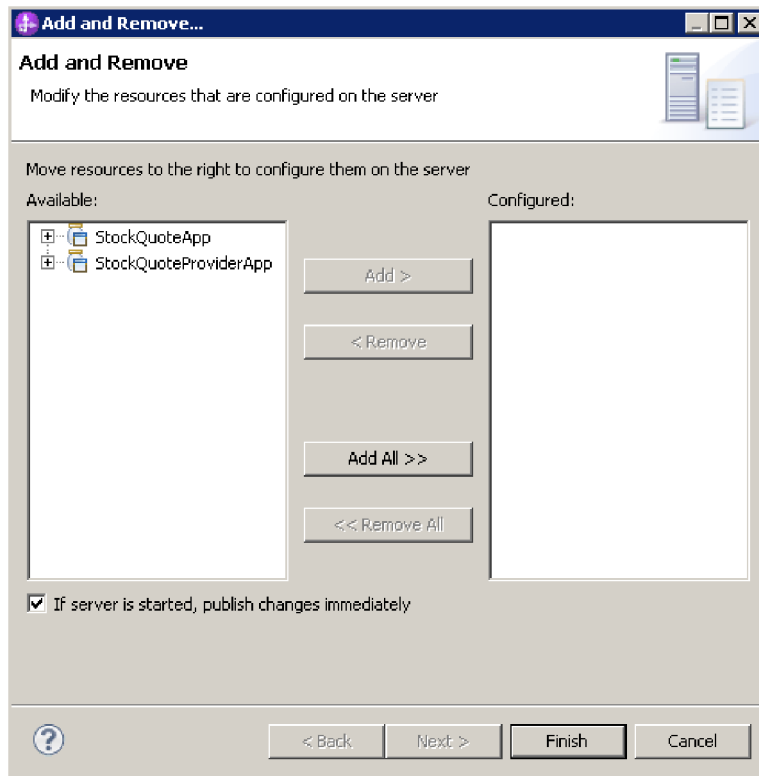
You may need to update the port used by the web services and then add the StockQuote application to the running server. Then, you can test your mediation flow in the test client.

1. In the Servers view, right click the server you will be deploying to, and select Properties. Select WebSphere Application Server. Check the value of the HTTP port.

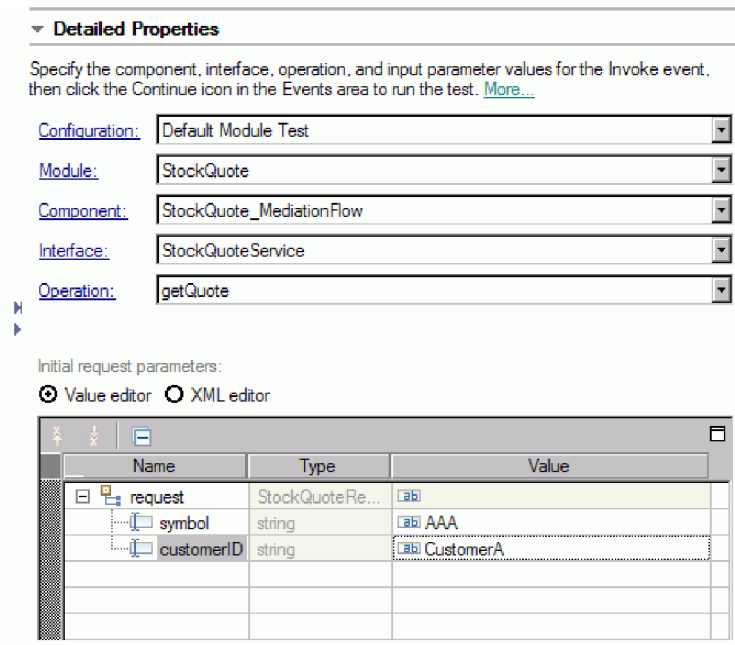



If the port is different from 9080 you have to make sure that the Stock Quote application looks for the web service on the correct port. This will affect the imports. To change the port:

- 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/StockQuoteProviderWeb/sca/DelayedServicePortTypeExport1`.
 - d. "Overwrite "9080" with the value of the HTTP port displayed from the Server Properties.
 - e. Follow the previous steps for the RealtimeService import and then save your module.
2. To add the StockQuoteApp project to the server, right-click 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.



3. Open the Assembly Diagram of the StockQuote mediation module. Then right-click the StockQuote_MediationFlow component and select **Test Component**. The Events page of the integrated test client opens.
4. In the Events page, you can select the modules, components, interfaces and operations that you want to test. For this sample, ensure the Detailed Properties are:
 - Configuration: **Default Module Test**
 - Module: **StockQuote**
 - Component: **StockQuote_MediationFlow**
 - Interface: **StockQuoteService**
 - 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.



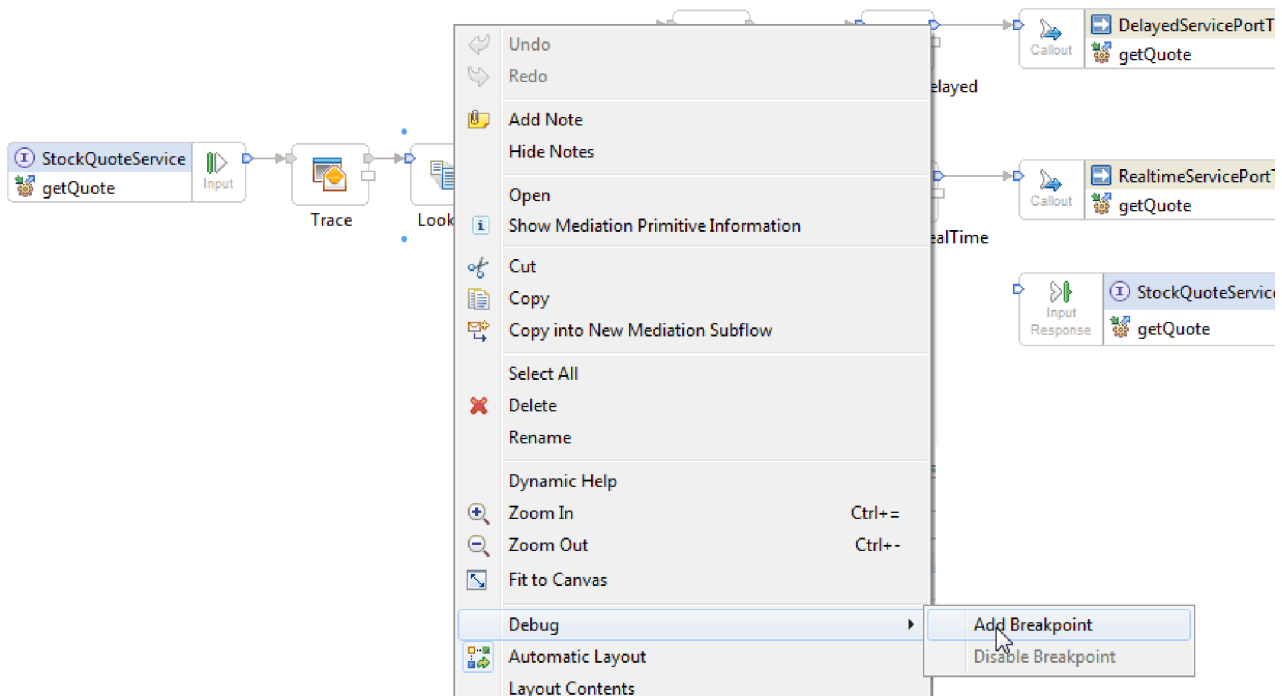
6. Next, select the **Continue** button  to invoke the getQuote operation. The Deployment Location window opens.
7. Select a server and click **Finish**. Enter the user ID and password for your server. The default is admin/admin.
You will see the resulting values for qualityOfService and value in the return parameters.
8. If the application does not run successfully, try the following:
 - If you see a service runtime exception, check that the web service application is started. In the Servers view, select the server, right click and select **Run Administration Console** and log in. In the list of running applications under **Applications > Enterprise Applications**, check the status of the SQSample application and start the application if necessary.
 - If you see an error indicating that the module StockQuote cannot be found, make sure the StockQuoteApp application is running, by following the instructions provided above.

Debug the mediation flow

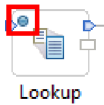
Advanced

Debug the StockQuote_MediationFlow component in the unit test environment by using the integration debugger.

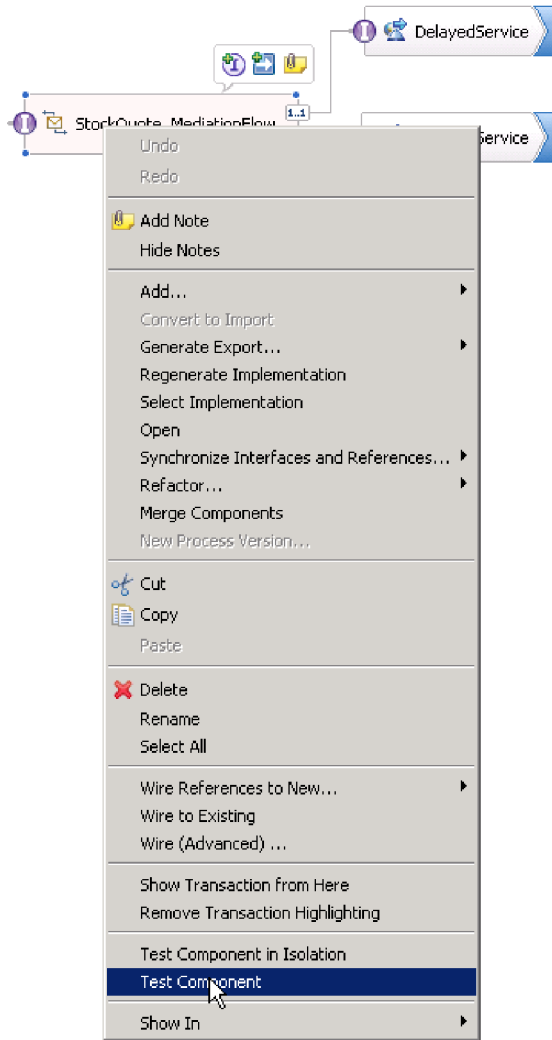
1. To add a breakpoint to a mediation primitive, right-click the primitive node in the request or response flow canvas, and select **Debug > Add Breakpoint**.



Notice that a small blue icon is added to the top left corner of the primitive, indicating that a breakpoint has been added to it.

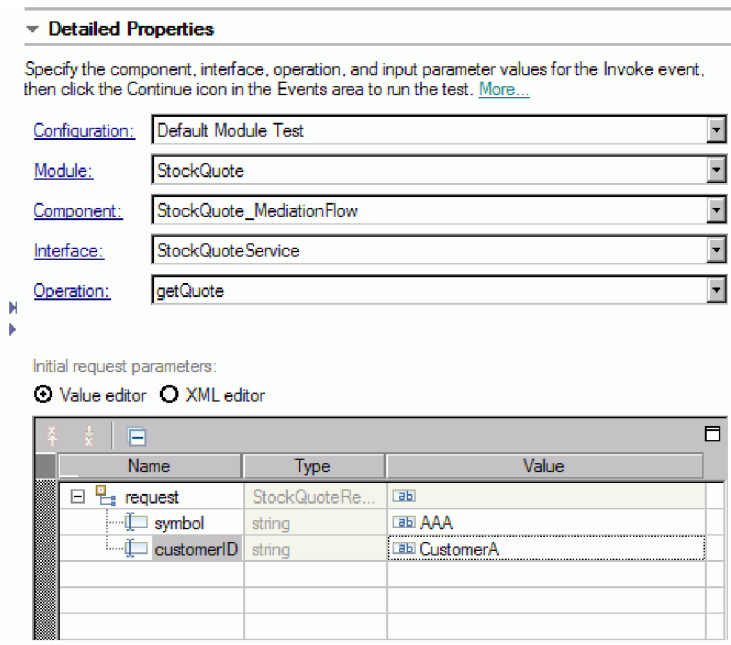




2. Select a server, right-click and click it and select **Restart in Debug** to restart the server in Debug mode. Note that if the server is already started, you must stop and start the server again in Debug mode.
3. When the server has started, open the Assembly Diagram of the StockQuote mediation module. Then right-click 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 that you want 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, click the **Continue** button . The Deployment Location window 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 .

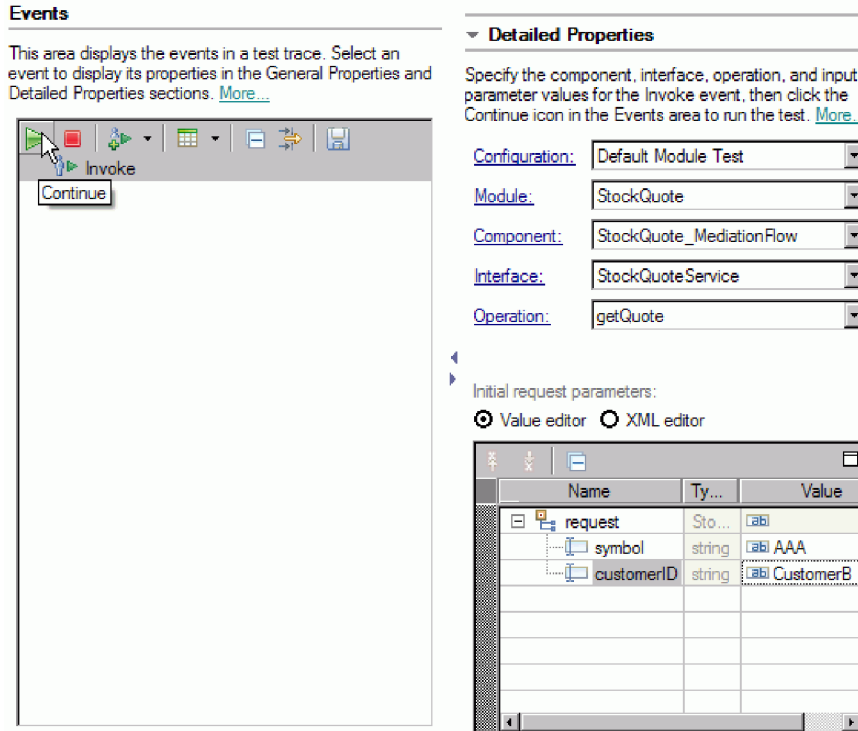
Change the quality of service at run time

Advanced

When you built the sample, you promoted the Filter pattern property. This lets you change the value of the property at run time. Now, you will change the value of the Filter pattern property in the unit test environment's administrative console, which causes the request to be sent to a different service. You will then view the result in the test client emulator.

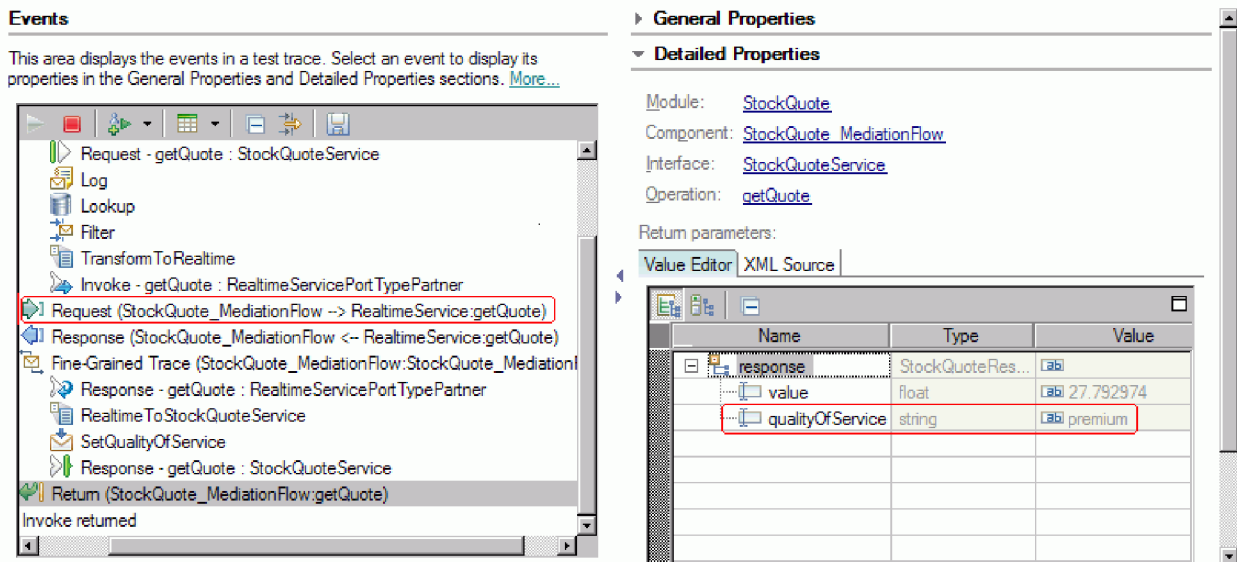
Open the StockQuote assembly diagram, and follow these steps to test the promoted property of the Filter primitive:

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.

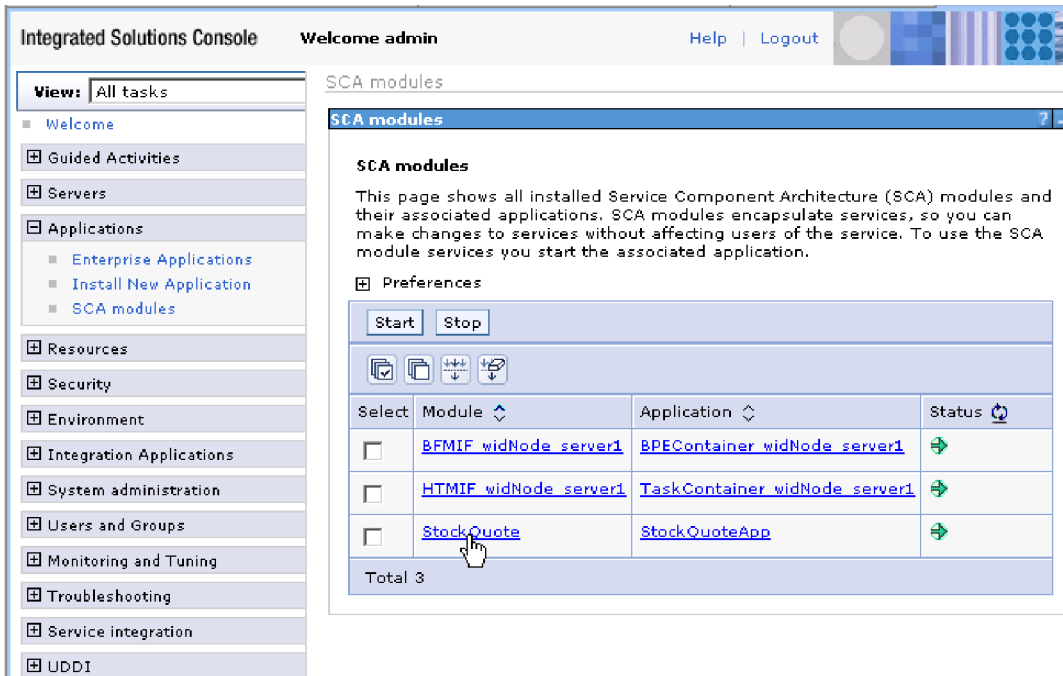


3. Select the server for the unit test environment. Click **Finish** and enter the username and password to login to the server. The default is admin\admin.

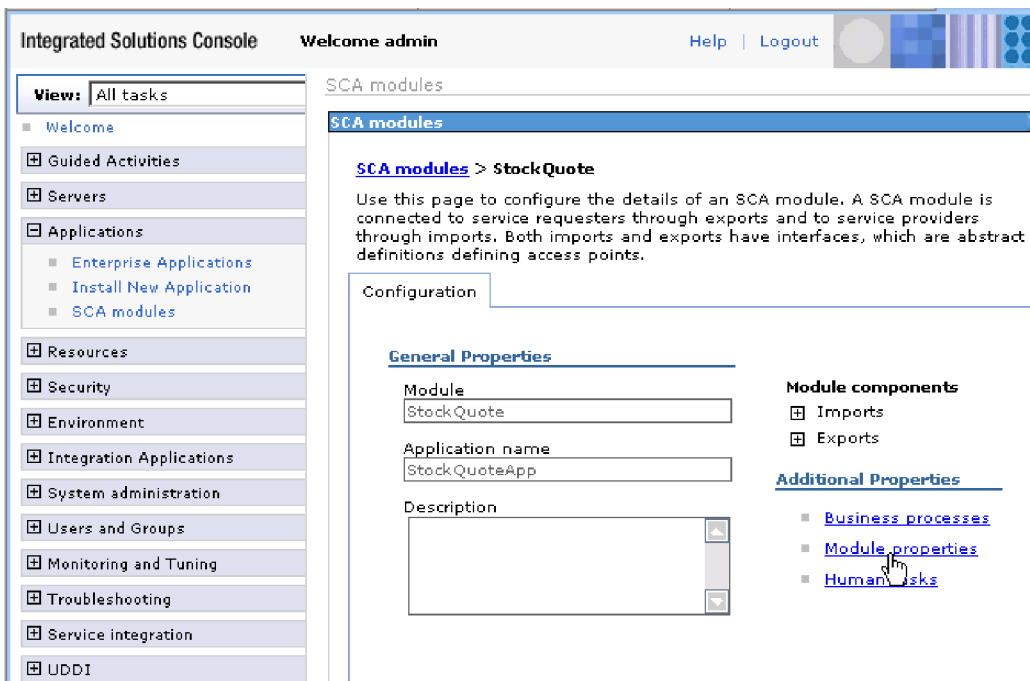
The results in the emulator show that the service invoked is RealtimeService. You can also see the value of the qualityOfService string which tells you that this customer's service level is premium.



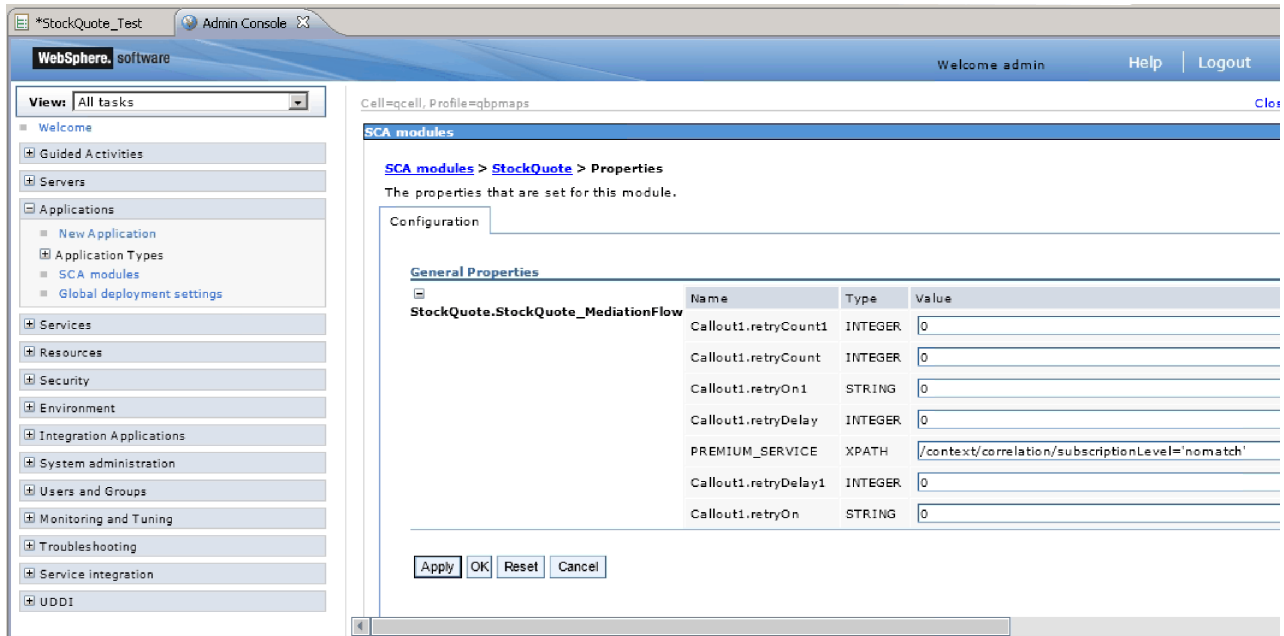
4. Switch to the Servers view. Right-click the server, and select **Administration > Run administrative console**. In the log in window, enter your userid and password (the default is admin/admin). Click **Log in**.
5. In the administrative console, expand **Applications** and click **SCA modules**.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



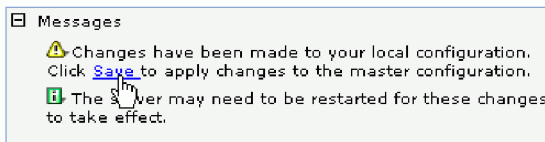
7. Click **Module Properties**.



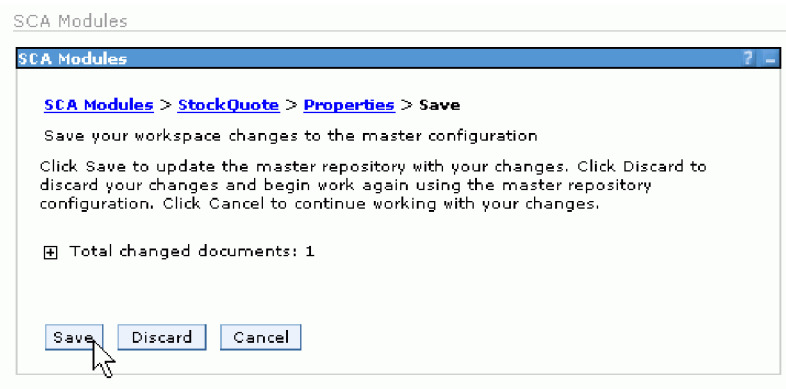
8. The property that you promoted earlier is displayed, showing the alias PREMIUM_SERVICE. Click the value field of PREMIUM_SERVICE, and change "premium" to "nomatch". Click **Apply**. This means that when the lookup returns with the subscriptionLevel set to premium, it won't match to the realtime terminal, and therefore will take the default path.



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



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



11. Switch to the Events page of the test client. Click Invoke  on the upper left side of the page.

12. 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.

Integration Test Client: StockQuote_Test

Events

This area displays the events in a test trace. Select an event to display its properties in the General Properties and Detailed Properties sections. [More...](#)

General Properties

Detailed Properties

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

Return parameters:

Value Editor: XML Source

Name	Type	Value
response	StockQuoteResponse	[ab]
value	string	[ab] 87.69361
qualityOfService	string	[ab] regular

Notices

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

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

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

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

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

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

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

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

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

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:

*Intellectual Property Dept. for IBM Integration
Designer
IBM Canada Ltd.
8200 Warden Avenue
Markham, Ontario
L6G 1C7 Canada*

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

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

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

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

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

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.

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.

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. 2000, 2009. All rights reserved.

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

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

IBM, IBM Logo, WebSphere, Rational, DB2, Universal Database DB2, Tivoli, Lotus, Passport Advantage, developerWorks, Redbooks, CICS, z/OS, and IMS are trademarks or registered trademarks of International Business Machines Corporation in the United States or other countries or both.

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

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

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

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

Adobe is either a registered trademark or trademark of Adobe Systems Incorporated in the United States, other countries, or both.

Other company, product and 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, 2012. All Rights Reserved.

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

IBM Integration Designer
Stock Quote Sample
Version 8 Release 5

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.

Email 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
IBM Integration Designer
8200 Warden Avenue
Markham, Ontario
Canada L6G 1C7

Fold and Tape

Please do not staple

Fold and Tape

Cut or Fold
Along Line