

WebSphere Enterprise Service Bus lab 2: Add custom mediation, database lookup, message filter and XSL transformation

What this exercise is about	2
Lab requirements	2
What you should be able to do	2
Introduction	3
Exercise instructions	5
Part 1: Prepare environment for lab 2.....	6
Part 2: Prepare CustomerRoutingMediationModule for lab 2.....	8
Part 3: Add a custom mediation primitive to mediation flow	12
Part 4: Add a database lookup mediation primitive to mediation flow	15
Part 5: Add a message filter mediation primitive to mediation flow	22
Part 6: Add XSL transformation mediation primitives to mediation flow	26
Part 7: Test the CustomerServiceExtended backend.....	34
Part 8: Save work and clean up server.....	36
What you did in this exercise	37
Solution instructions	38
Task: Adding remote server to WebSphere Integration Developer test environment	39

What this exercise is about

The objective of this lab is to create another backend, CustomerServiceExtended, and use mediations to help decide which backend to use. Read the introduction below for a bigger picture of what this lab is about.

Lab requirements

List of system and software required for the student to complete the lab.

- WebSphere Integration Developer V6.1 with the WebSphere Enterprise Service Bus test server option installed.

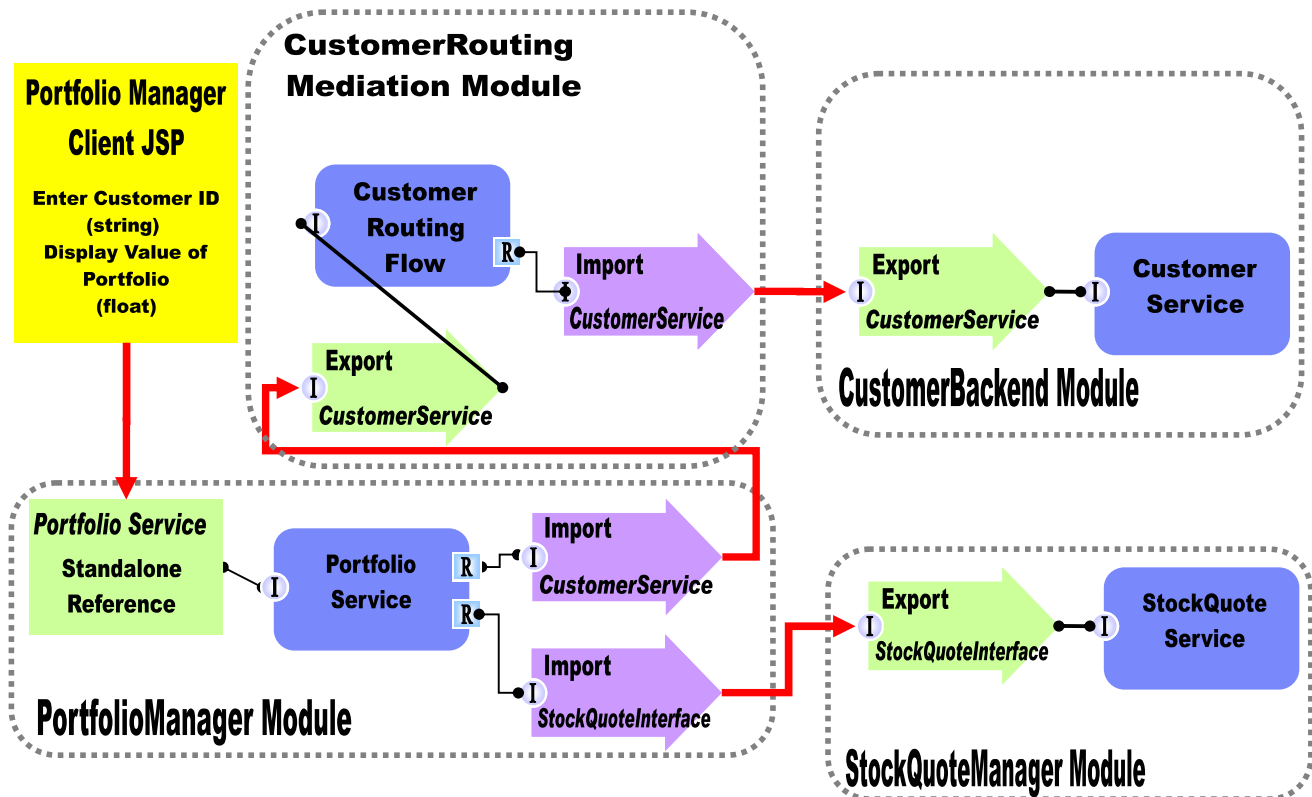
What you should be able to do

At the end of this lab you should be able to:

- Import project interchange files into the WebSphere Integration Developer V6.1 development environment.
- Know how to create Custom Mediation primitives and understand what they do.
- Know how to create Database Lookup primitives and understand what they do.
- Know how to create Message Filter primitives and understand what they do.
- Know how to create XSL Transformation primitives and understand what they do.
- Navigate the Properties View for mediation information.

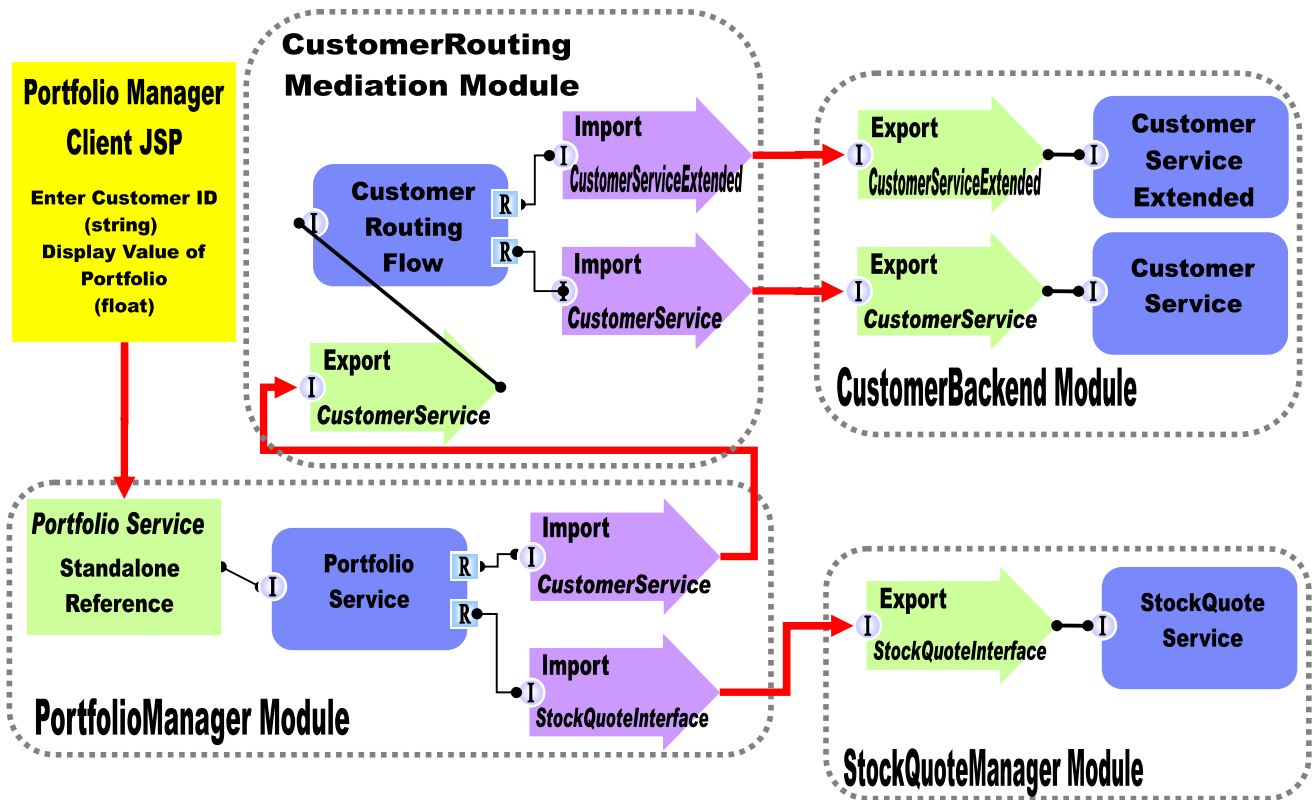
Introduction

In Lab-1 of this lab series, you changed an SCA application to one that uses a mediation module as described in the diagram below. You added the CustomerRoutingMediationModule between the PortfolioManager service and the CustomerBackend service to allow you access to message information being passed between them. You then created a mediation flow called CustomerRoutingFlow that uses a Message Logger mediation to store message information in a derby database.



In Lab2, you will add to the Message Logger mediation primitive with a couple others available in WebSphere Integration Developer V6.1. The main goal for Lab-2 is to create another backend, **CustomerServiceExtended**, and use mediations to help decide which backend to use. You will first add a Custom Mediation. The Custom Mediation is a mediation primitive that is used when you want some custom functionality from the mediation. For this custom mediation, it will use a Java snippet that will extract a two digit prefix from the customer ID and place 2 digit it in the transient context (a business object). Think of transient context as a scratchpad, a place for information to be stored in a message as it passes through a flow. You will then add a Database Lookup primitive that uses that two digit prefix from transient context as key to lookup a backend identifier to place into transient context. The Customer ID prefixes with 11, 22, 33, and 44 will go to the **CustomerService** backend. Customer ID prefixes with 55, 66, 77, 88, and 99 will go to a **CustomerServiceExtended** backend. To determine the message routing based on backend identifier, you will add a Message Filter primitive. The old backend will go directly to the callout for **CustomerService** and the new backend will go to the XSL Transformation. The XSL Transformation will transform the **CustomerService** business object into the **CustomerServiceExtended** business object, and then pass the newly formed message to the callout for **CustomerServiceExtendedPartner** instead of the **CustomerServicePartner**. Notice that this is just the request. There is also a flow for the response. On the response side, the **CustomerServicePartner** callout response will go directly to the **CustomerService** input response. However, the **CustomerServiceExtended** callout response will need to go back through XSL Transformation mediation in order to transform the response body from the **CustomerServiceExtended** business object back to the **CustomerService** business object. Feeding the

CustomerServiceExtended business object to application will end in error because the application only knows how to work with the CustomerService business object. This is the reason why you need an XSL Transformation mediation primitive; to map one business object to another. The diagram will now look like the one below.



In Lab-3 of this lab series, you will use the Visual Debugger to step through the application in order to see what is happening behind the scenes.

In Lab-4, you are going to learn how to create EAR files for mediation modules, install EAR files to the WebSphere ESB Server profile, and edit/connect SCA Binding information without using WebSphere Integration Developer. You will use the administrative Console for the WebSphere ESB Server. Throughout the lab you will learn what WebSphere Integration Developer does behind the scenes by manually exporting EAR files, installing EAR files, starting/stopping the ESB server, and using the administrative console to edit SCA Binding information. This is helpful to know how to take these actions outside of the development environment and to understand what WebSphere Integration Developer is doing for you when inside the development environment.

Exercise instructions

Some instructions in this lab are Windows operating system specific. If you plan on running WebSphere Integration Developer on a Linux operating system you will need to run the appropriate commands and use appropriate files for Linux. The directory locations are specified in the lab instructions using symbolic references, as follows:

Reference Variable	Windows Location	AIX®/UNIX® Location
<WID_HOME>	C:\Program Files\IBM\WID61	/opt/IBM/WID61
<LAB_FILES>	C:\Labfiles61\WESB\Lab2	/tmp/Labfiles61/WESB/Lab2
<WORKSPACE>	C:\Labfiles61\WESB\Lab2\workspace	/tmp/Labfiles61/WESB/Lab2/workspace
<SOLUTION>	C:\Labfiles61\WESB\Lab2\solution	/tmp/Labfiles61/WESB/Lab2/solution

Windows users note: When directory locations are passed as parameters to a Java program such as EJBdeploy or wsadmin, it is necessary to replace the backslashes with forward slashes to follow the Java convention. For example, C:\LabFiles61\ is replaced by C:/LabFiles61/

Note that the previous table is relative to where you are running WebSphere Integration Developer. The following table is related to where you are running remote test environment:

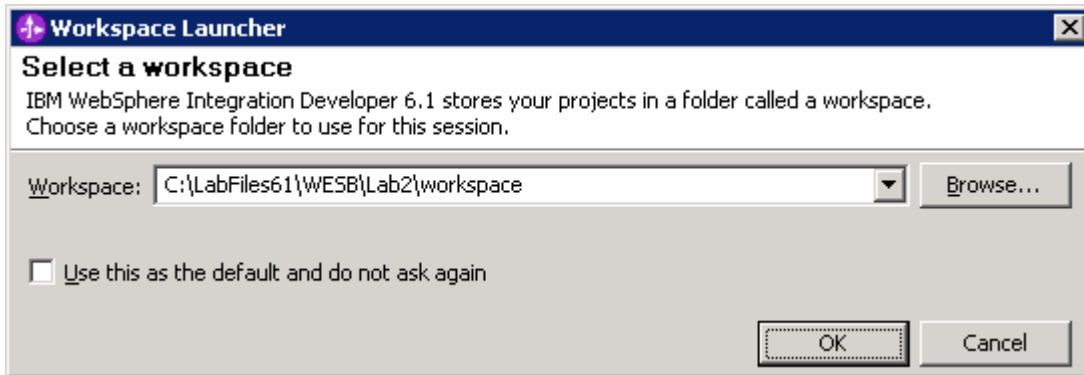
Reference variable	Example: Remote Windows test server location	Example: Remote z/OS® test server location	Input your values for the remote location of the test server
<SERVER_NAME>	server1	sssr011	
<WAS_HOME>	C:\Program Files\IBM\WebSphere\AppServer	/etc/sscell/AppServer	
<HOSTNAME>	localhost	mvsxxx.rtp.raleigh.ibm.com	
<SOAP_PORT>	8880	8880	
<TELNET_PORT>	N/A	1023	
<PROFILE_NAME>	AppSrv01	default	
<USERID>	N/A	ssadmin	
<PASSWORD>	N/A	fr1day	

Instructions for using a remote testing environment, such as z/OS, AIX or Solaris, can be found at the end of this document, in the section [“Task: Adding Remote Server to WebSphere Integration Developer Test Environment”](#).

Part 1: Prepare environment for lab 2

In this section of the lab, you will import all projects inside the **WPIv61_ESB_StartLab2_PI.zip** project interchange file into your workspace. Remember this is the sample SCA application that you are going to add ESB specific mediations to.

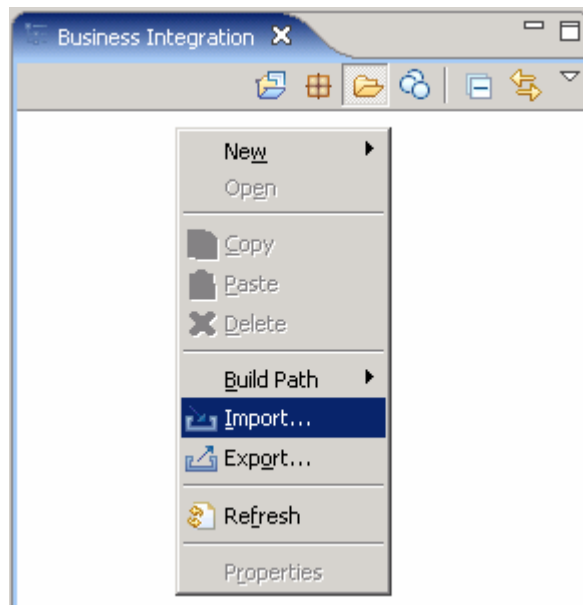
- ___ 1. Start WebSphere Integration Developer V6.1 with a workspace location of **<WORKSPACE>**.



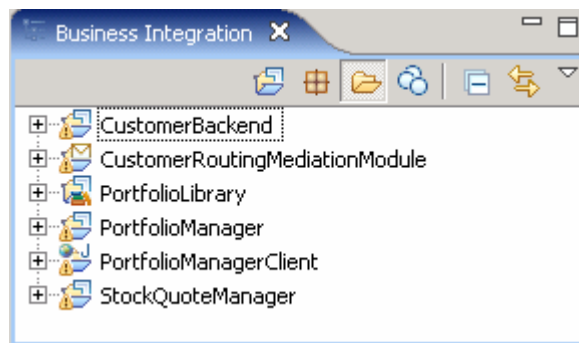
- ___ 2. On the Welcome window, click the curved arrow at top right to **go to workbench**.



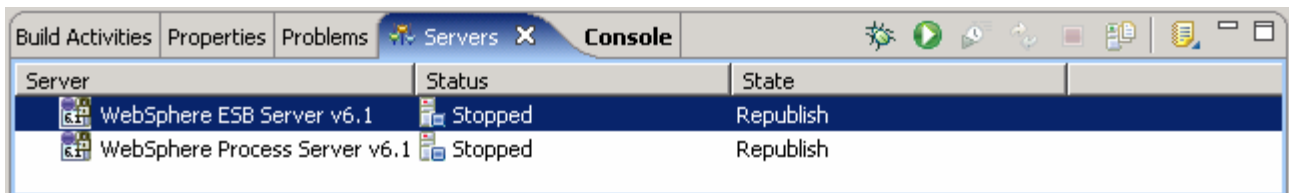
- ___ 3. Import Project Interchange file, **WPIv61_ESB_StartLab2_PI.zip**, into a new workspace.
 - ___ a. Right-click inside **Business Integration View** (top left view in the Business Integration Perspective).




- ___ b. Select **Import** from the pop-up menu. Expand '**Other**' and select **Project Interchange** from the Import panel. Click **Next**.
- ___ c. Click the top **Browse** button for '**From zip file:**'
- ___ d. Navigate to **<LAB_FILES>/import/WPIv61_ESB_StartLab2_P1.zip**.
- ___ e. Click the **Select All** button to select the all check boxes for the projects listed. The projects that should be listed are **CustomerBackend**, **CustomerRoutingMediationModule**, **PortfolioLibrary**, **PortfolioManager**, **PortfolioManagerClient** and **StockQuoteManager**
- ___ f. Click **Finish** button (projects are imported and auto-build will run).
- ___ g. Verify you have **CustomerBackend**, **CustomerRoutingMediationModule**, **PortfolioLibrary**, **PortfolioManager**, **PortfolioManagerClient** and **StockQuoteManager** modules listed in the Business Integration view.

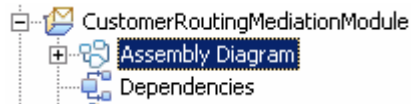


- ___ h. Verify you have WebSphere ESB Server v6.1 listed in your Servers view.

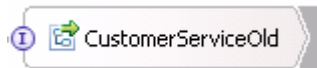


Part 2: Prepare CustomerRoutingMediationModule for lab 2

- ___ 1. Expand **CustomerRoutingMediationModule** in the Business Integration view and double click on **Assembly Diagram** ( **Assembly Diagram**) to open it in the Assembly Diagram editor




- ___ 2. Change the name of the import **CustomerServiceOut** to **CustomerServiceOld**.



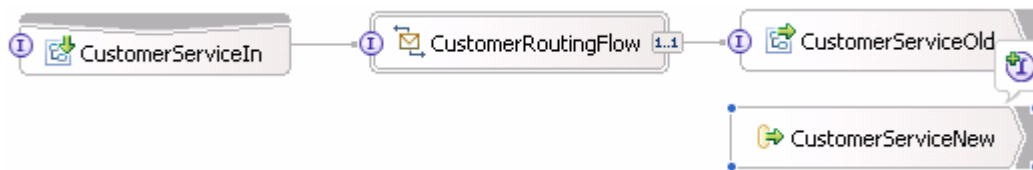
- ___ 3. Add a new Import, **CustomerServiceNew** to the Assembly Diagram of CustomerRoutingMediationModule.

___ a. Drag and drop an import on the right side of CustomerRoutingFlow.

___ b. Click on **Import** icon from Assembly Diagram tray ().

___ c. Click or drag import to the right side of CustomerRoutingFlow.

___ d. Change default import name from Import1 to **CustomerServiceNew**.



___ e. Hover over the import or click on the import to make the add interface icon appear.

Click "I" icon () to add an interface.

1) Select **CustomerServiceExtended** from the list of matching interfaces. Click **OK**.

2) Wire CustomerRoutingFlow and CustomerServiceNew together by clicking on **CustomerRoutingFlow** and drag a wire to **CustomerServiceNew**.
Click **OK** for any pop-up windows.

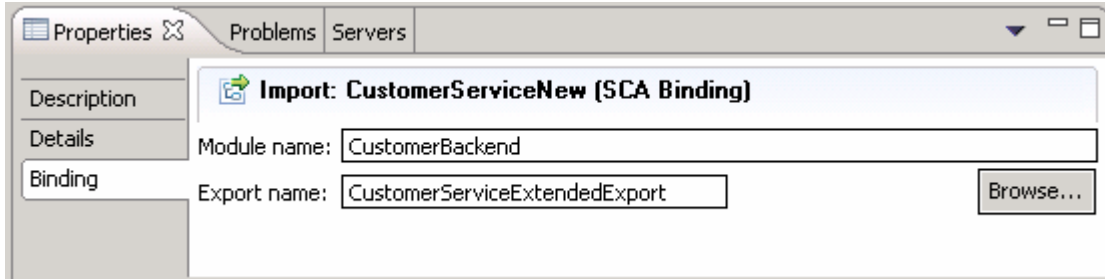


3) Update SCA binding for **CustomerServiceNew**.

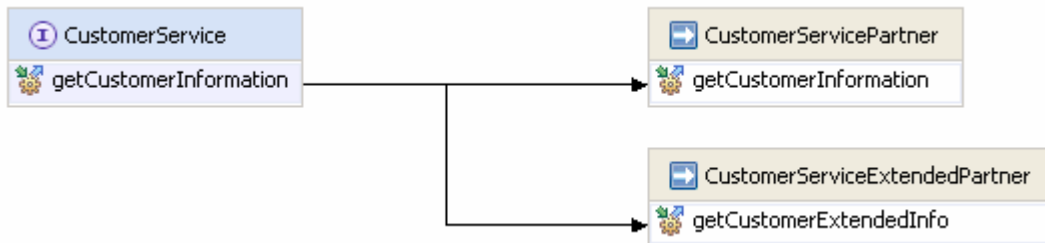
a) Right-click on CustomerServiceNew and select **Generate Binding → SCA Binding** from the pop-up menu.

b) Select the Import **CustomerServiceNew** on the Assembly Diagram.

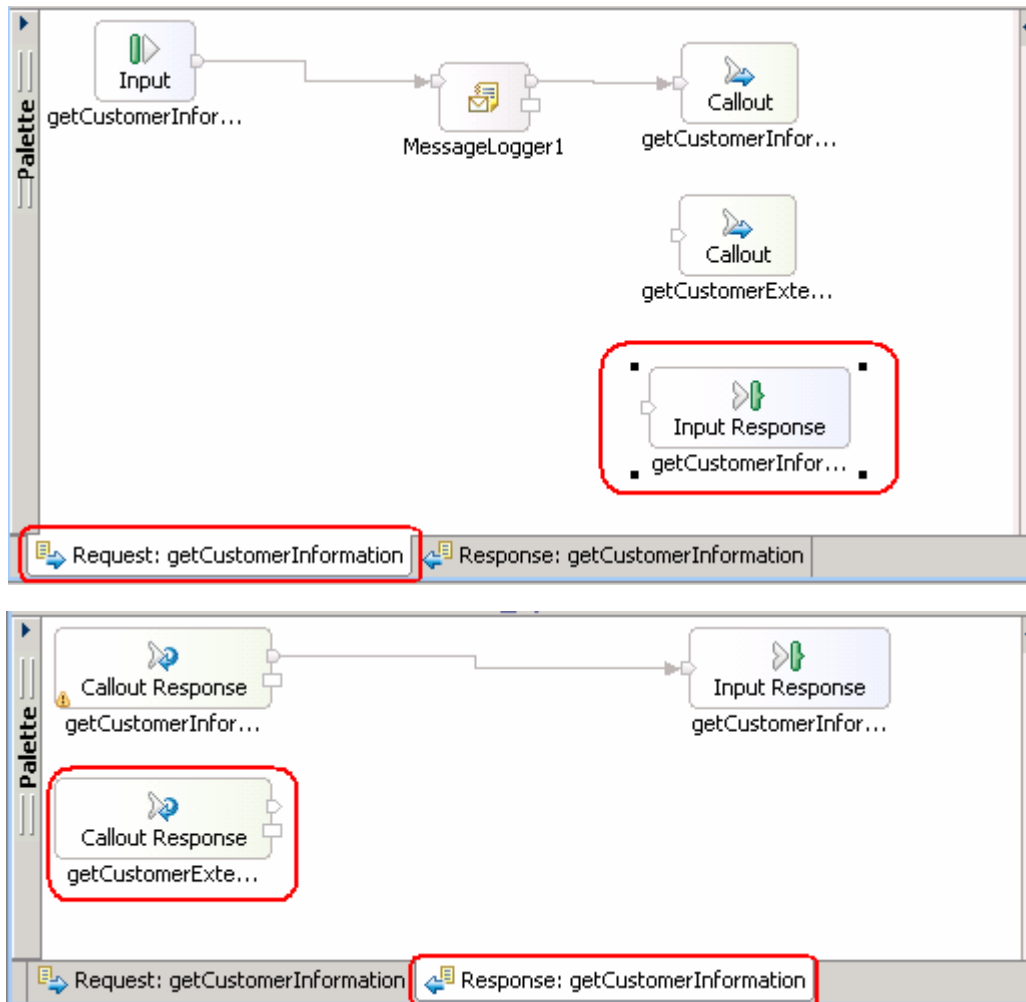
- c) Open the **Binding** tab of **Properties View**.
- d) Click **Browse...** button and select **CustomerServiceExtendedExport** from the SCA Export Selection panel. Click **OK**.
- e) Here is what your bindings tab should look like now.



- f) Save all work by **File → Save All** or **Ctrl + Shift + S**.
 - g) After saving the Assembly Diagram, the new implementation must be synchronized from the Assembly Diagram to the Mediation flow.
- 4) Synchronize implementation from Assembly Diagram to Mediation Flow to reflect the new input added to the CustomerRoutingMediationModule.
- a) Right click on the **CustomerRoutingFlow** in the Assembly Diagram.
 - b) Select **Synchronize Interfaces and References → to Implementation** from the pop-up menu. Wait for the workspace to get auto built.
 - c) Now double click **CustomerRoutingFlow** in the Assembly Diagram to open it in the Mediation Flow Editor.
 - d) You should see that the **CustomerServiceExtendedPartner** is added to the Operation connections view (top-most view of Mediation Flow Editor).
 - e) Drag a wire from the CustomerService operation on left side of the Operation connections view to the CustomerServiceExtendedPartner.
 - f) Click on the wire in the Operation connections view in the Mediation Flow Editor to see the Mediation Flow view (middle view).



- g) Notice also in the Mediation Flow view that an **Input Response** (for Request tab - 1) and a **Callout Response** (for Response tab - 2) nodes has been added for CustomerServiceExtendedPartner.



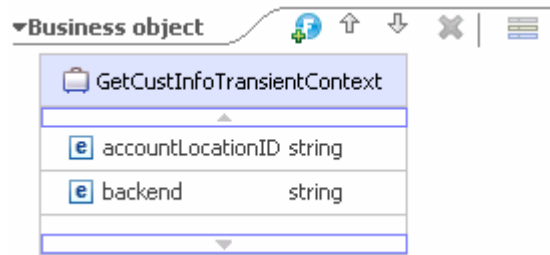
h) Save the Mediation Flow editor by selecting **File** → **Save** or **Ctrl + S**.

5) Create a transient context business object.

In this lab, the custom mediation primitive you create in the next section uses Java code that will extract a two digit prefix from the customer ID and places those 2 digits in the transient context. Think of the transient context as a scratchpad, somewhere to store information in a message. It is defined as a business object (or DataType in the Business Integration view) and part of the SMO (Service Message Object). These following steps show you how to create a transient context business object.

- a) In the **Business Integration** view, expand **CustomerRoutingMediationModule** to select the entry '**Data Types**'.
- b) Right-click on **Data Types** and select **New** → **Business Object** from the pop-up menu.
- c) In the New Business Object panel, enter **GetCustInfoTransientContext** for the Name field. Click **Finish**.
- d) The Business object editor opens. Click add a field to business object icon (+) twice to add two fields to the business object.

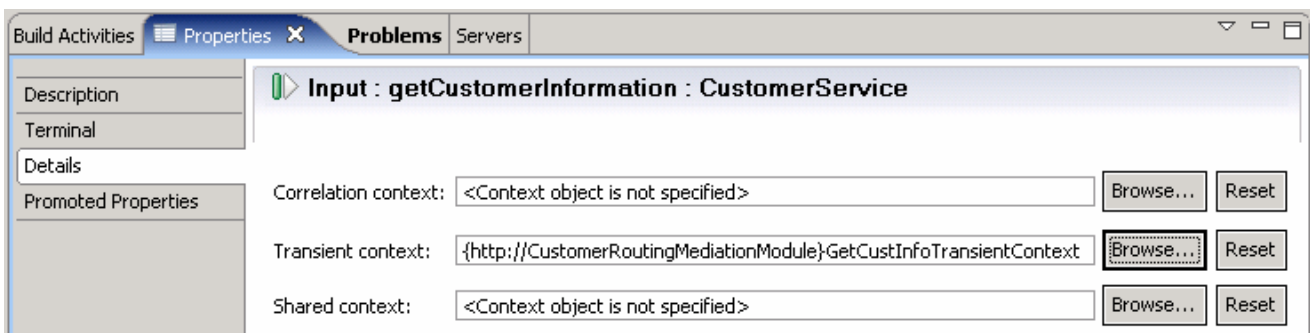
- e) Change the text field1 to **accountLocationID**.
- f) Change the text field2 to **backend**. The business object should look like picture shown below:



- g) Save and close the **GetCustInfoTransientContext** business object editor.
- 6) Add **GetCustInfoTransientContext** to mediation flow input.
- a) Open the **CustomerRoutingFlow** Mediation Flow Diagram from the Business Integration view by double-clicking the entry **CustomerRoutingMediationModule** → **Mediation Logic** → **Flows** → **CustomerRoutingFlow** if not already open.
 - b) Click on wire connecting CustomerService to CustomerServicePartner and CustomerServiceExtendedPartner in the operations connections view.
 - c) Select the only input under the request tab of the Mediation Flow view, **getCustomerInformation: CustomerService** (middle view).



- d) Go to the **Details** tab of the **Properties** view with the input selected.
- e) Click the **Browse...** button in the same row as **Transient Context**.
- f) Select **GetCustInfoTransientContext** from the Data Type Selection panel. Click **OK**.
- g) The **Details** tab will now look like this:



- h) Save changes by selecting **File** → **Save** or **Ctrl+S**.
- i) You are now ready to start creating mediation primitives.

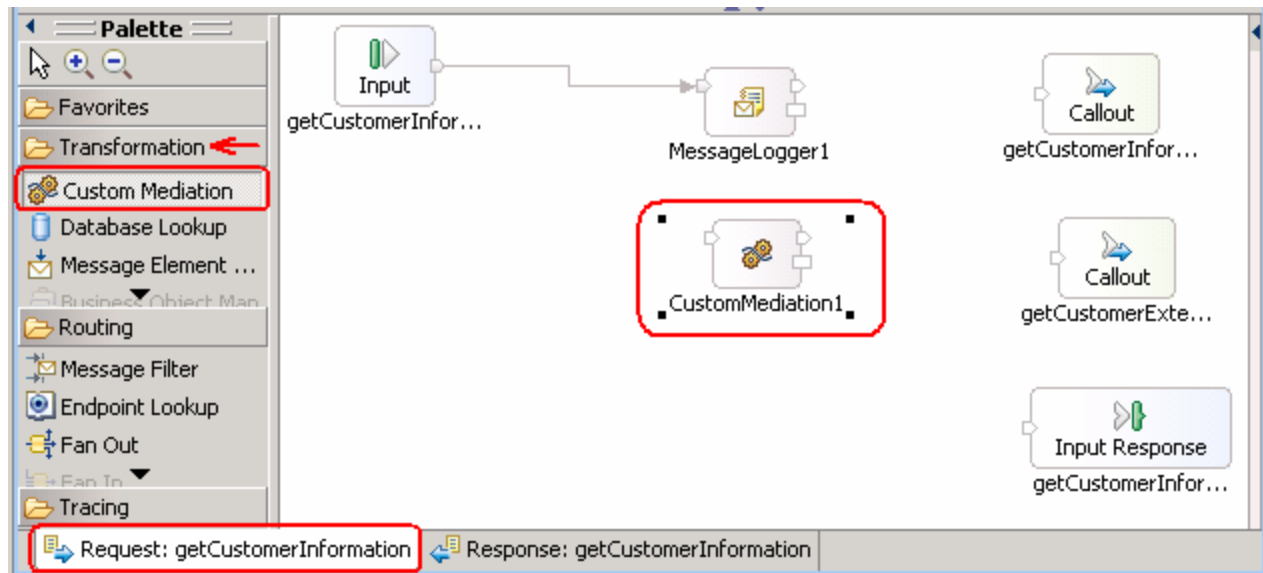
Part 3: Add a custom mediation primitive to mediation flow

In this section of the lab, you will add a custom mediation primitive to the mediation flow. The custom mediation uses Java code which is going to be provided for you in a snippet that will extract a two digit prefix from the customer ID and place it in the transient context.

- ___ 1. Delete the wire from MessageLogger1 to CustomerServicePartner Callout in order to connect message logger primitive to a new custom mediation primitive.
 - ___ a. In the Mediation Flow view of the Mediation Flow Editor, click on the **wire** connecting **MessageLogger1** to **CustomerServicePartner** callout.



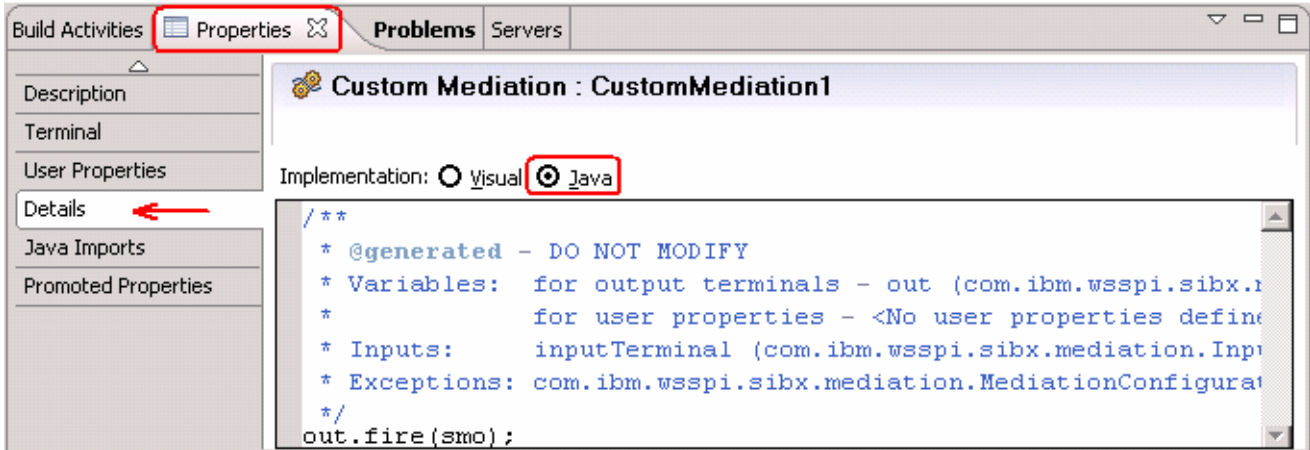
- ___ b. Press the Delete button on the keyboard or right click on wire and select **Delete**.
- ___ c. From the pallet tray to the left, click to expand '**Transformation**' and then click the custom mediation icon (Custom Mediation).
- ___ d. Now click under the **MessageLogger1**. A new custom mediation, '**CustomMediation1**' appears as shown below:



- ___ e. Now wire the output terminal of **MessageLogger1** to the input terminal of **CustomMediation1**. Click on output terminal of **MessageLogger1** and drag a wire to the input terminal of **CustomMediation1**.



__ f. Define a Service Operation for CustomMediation1. Click on **CustomMediation1** and navigate to the **Details** tab of **Properties View**.



__ g. Ensure the radio button for **Java** is selected.

__ h. Add snippet to mediate method and organize imports for code

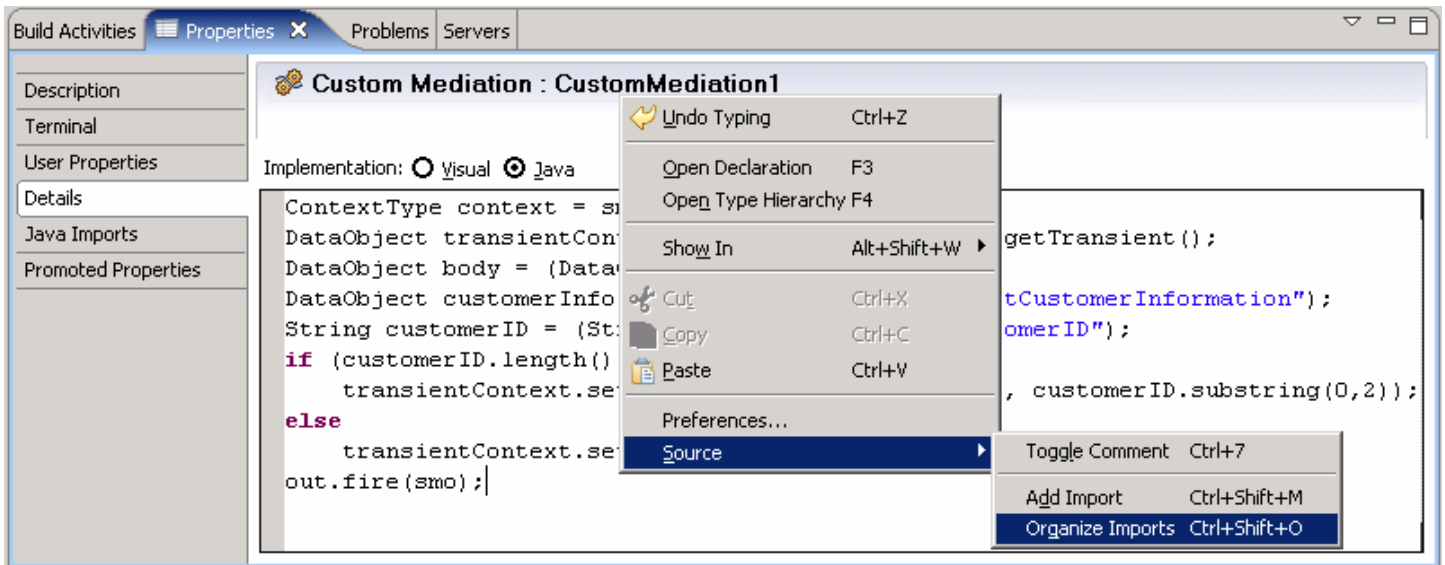
- 1) Outside of WebSphere Integration Developer, navigate to **C:/Labfiles61/WESB/Lab2/snippet/snippet1.txt** and open in a text editor.
- 2) Copy contents and paste to overwrite the existing contents in the text area as shown below:

Implementation: Visual Java

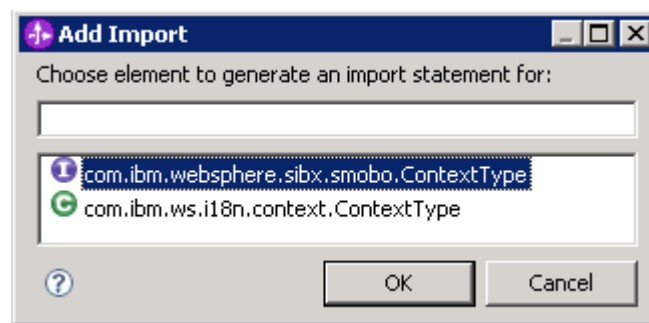
```

ContextType context = smo.getContext();
DataObject transientContext = (DataObject)context.getTransient();
DataObject body = (DataObject)smo.getBody();
DataObject customerInfo = (DataObject)body.get("getCustomerInformation");
String customerID = (String)customerInfo.get("customerID");
if (customerID.length() > 1)
    transientContext.setString("accountLocationID", customerID.substring(0,2));
else
    transientContext.setString("accountLocationID", "00");
out.fire(smo);
    
```

__ i. Now right click any where on the implementation text area and select **'Source → Organize Imports'** (or press **Ctrl + Shift + O**)

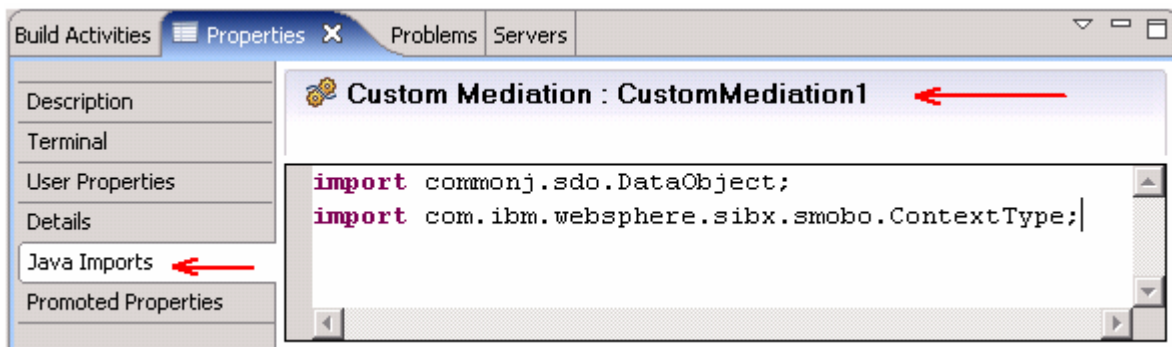


__ j. The 'Add Imports' panel opens. Select '**com.ibm.websphere.sibx.smobo.ContextType**' from the list



__ k. Click **OK**

__ l. Now select the properties, '**Java Imports**' tab and you should see the imports populated as shown below:



__ f. Save changes by selecting **File → Save** or **Ctrl+S**.

__ m. You have now fully created a custom mediation primitive with working Java code to support it.

Part 4: Add a database lookup mediation primitive to mediation flow

This section is two fold. In the first part of this section, you will learn how to set up a data source that the database lookup mediation primitive will use. A simple database is supplied for this lab, however walking through the steps to set up the data source allows you to view the Administration Console and what it takes to use a database lookup mediation primitive. In the second part of this section, you will add a Database Lookup primitive that uses that two digit prefix from transient context as key to lookup a backend identifier to also place into transient context. The Customer ID prefixes with 11, 22, 33, and 44 will go to the CustomerService backend. Customer ID prefixes with 55, 66, 77, 88, and 99 will go to a CustomerServiceExtended backend.

___ 1. Add a data source in the administration console for the database lookup primitive to use.

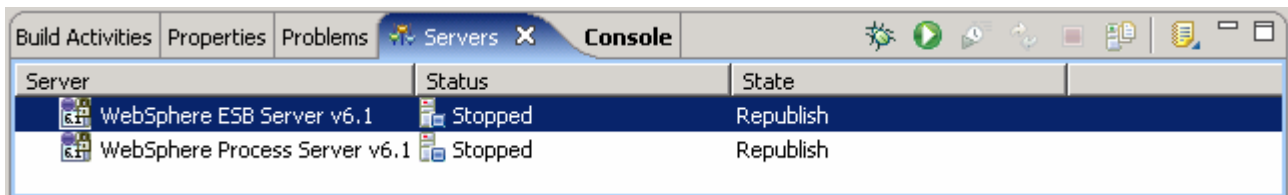
Start **WebSphere ESB Server v6.1**.

If using a remote testing environment, follow the instructions in [Task: Adding Remote Server to WebSphere Integration Developer Test Environment](#) at the end of this document, to start the remote server.

If using a local testing environment:

___ a. Open **Servers View**.

___ b. Highlight WebSphere ESB Server v6.1 and click **Start button** ().



___ c. This will take some time. Wait for the server status to say **Started**.

___ d. Right-click on the '**WebSphere ESB Server v6.1**' in the Servers view and select **Run administration console** from the context menu

___ e. Log-in to the administration console.

___ f. In the left navigation menu, expand **Resources** → **JDBC** and then click the '**Data sources**' link

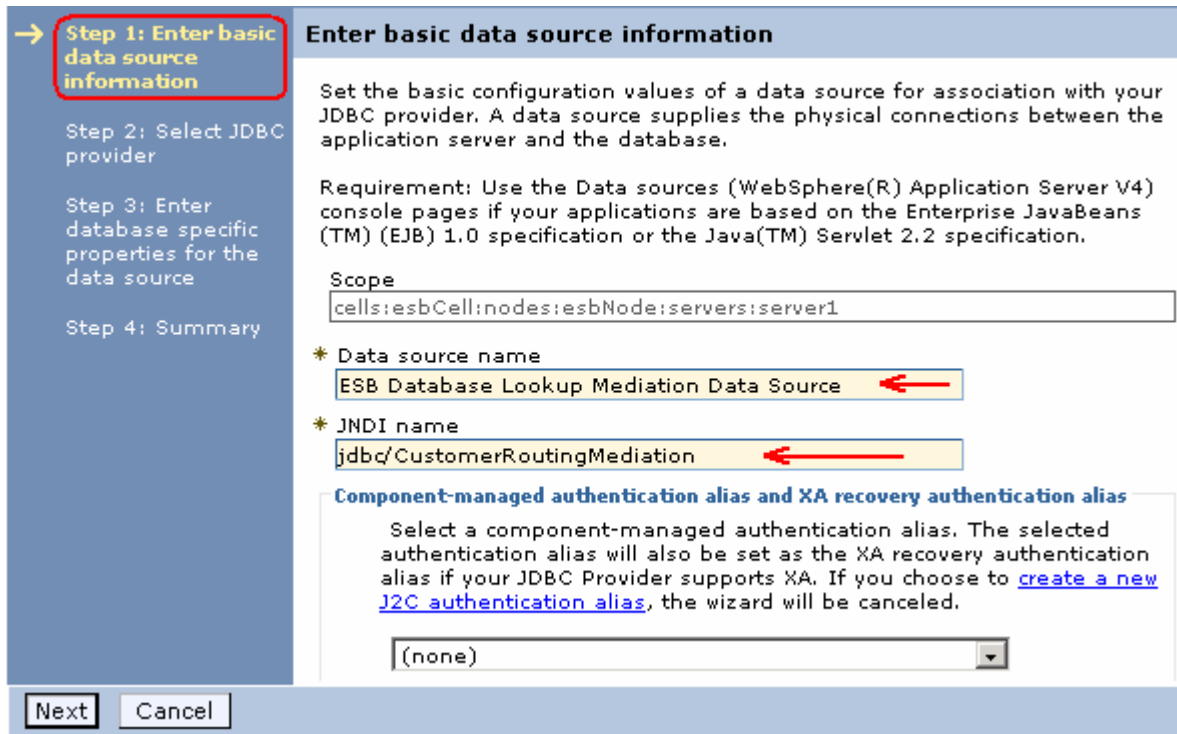
___ g. In the next panel, set the '**Scope**' to server by selecting from the drop down list

Node=esbNode, Server=server1

___ h. Click the **New** button at top-right of the data sources table to create a new data source. The '**Create a data source**' wizard is launched

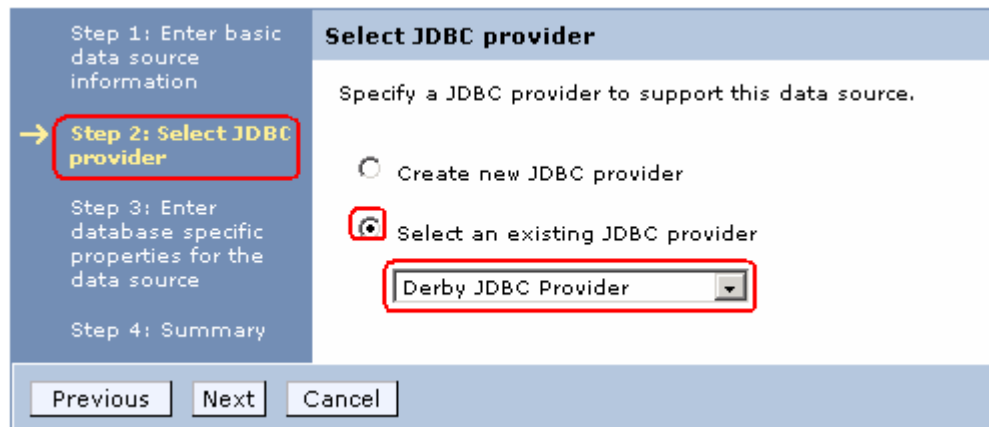
___ i. In the '**Step 1: Enter basic data source information**' panel, enter the parameters listed below:

- Data source name : **ESB Database Lookup Mediation Data Source**
- JNDI name : **jdbc/CustomerRoutingMediation**



__ j. Click **Next**

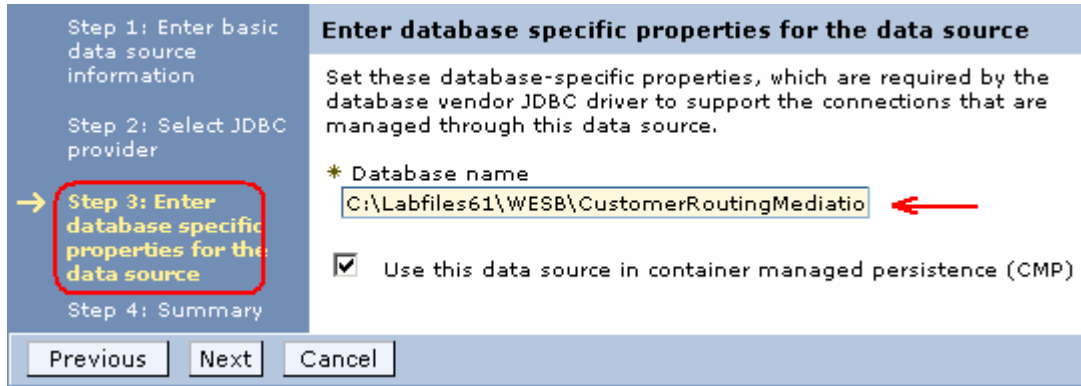
__ k. In the '**Step 2: Select JDBC Provider**' panel, select the radio button for '**Select an existing JDBC provider**' and then select '**Derby JDBC Provider**' from the drop down list



__ l. Click **Next**

__ m. In the '**Step 3: Enter database specific properties for the data source**' panel, enter parameter listed below:

- **C:\Labfiles61\WESB\CustomRoutingMediation**



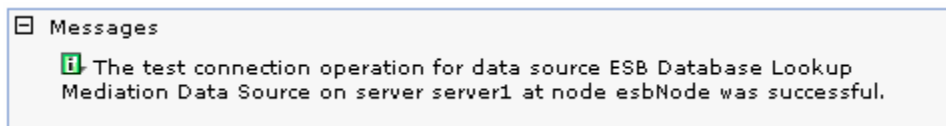
Note: Ensure that the database directory **CustomerRoutingMediation** is existing under **C:\Labfiles61\WESB**. If not use the correct path of where the **CustomerRoutingMediation** is located.

Note: If using a remote system for testing, you will need to copy the database to the remote system by uploading it in binary format. This entails uploading the entire database folder. Once on your remote system, upload just the service.properties file in ASCII. You will then need to substitute the directory where you placed the database, such as **<LAB_FILES>/CustomerRoutingMediation**. Make sure the permissions are set up to allow the server to access the database.


- ___ n. Click **Next**.
- ___ o. In the '**Step 4: Summary**' panel, review the '**Summary of actions**' and click the '**Finish**' button.
- ___ p. Save to the master configuration.
- ___ q. After the changes are saved, you will see the data source table is updated with the new data source.

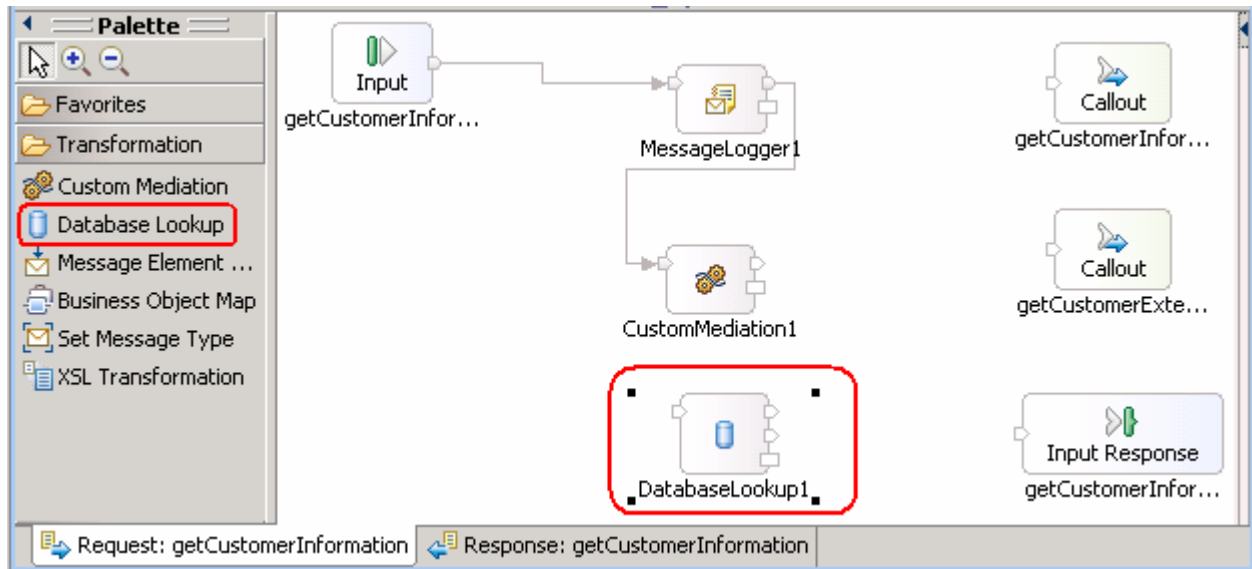
<input type="checkbox"/>	ESB Database Lookup Mediation Data Source	jdbc/CustomerRoutingMediation	Node=esbNode,Server=server1	Derby JDBC Provider
--------------------------	---	-------------------------------	-----------------------------	---------------------

- ___ r. Select the check box next to **ESB Database Lookup Mediation Data Source** and click the '**Test Connection**' button at the top of the data sources table. Ensure the database connection is successful with the message shown below:

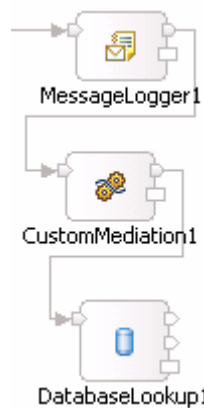


- ___ s. At the top-left of the administrative console page, click on **Logout**. Close the **administrative console**. You have now built the data source and are ready to build the Database Lookup mediation primitive in your workspace.
- ___ 2. Open the **CustomerRoutingFlow** Mediation Flow diagram from the Business Integration view by double-clicking the entry **CustomerRoutingMediationModule → Mediation Logic → Flows → CustomerRoutingFlow** if not already open.
- ___ 3. Drop a database lookup primitive on to the Mediation Flow to create a new database lookup primitive.

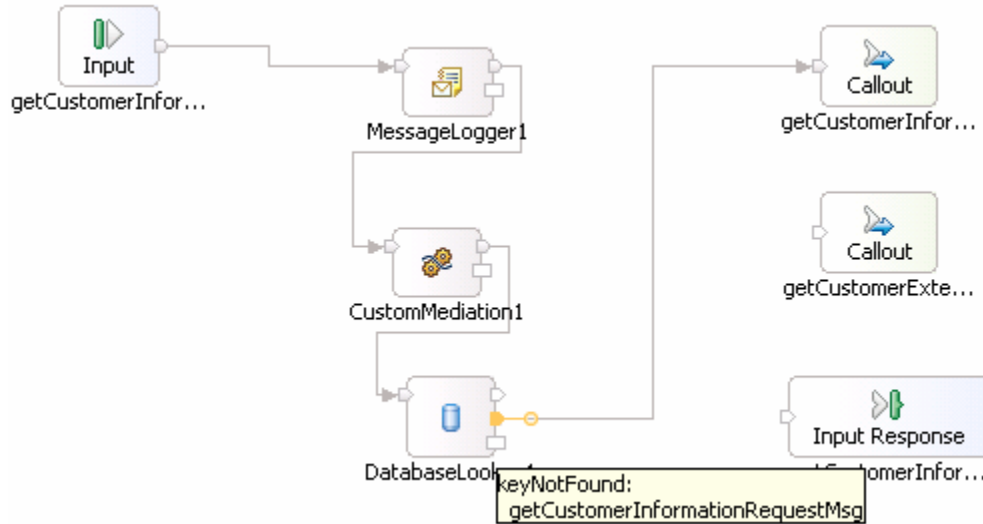
- ___ a. Click the database lookup icon ( Database Lookup) in the Transformation palette (left-hand side) and then drop it under CustomMediation1. The DatabaseLookup1 primitive will appear.



- ___ b. Wire the output terminal of **CustomMediation1** to the input terminal of **DatabaseLookup1**. Click on output terminal of CustomMediation1 and drag a wire to the input terminal of DatabaseLookup1.



- ___ c. Connect the **keyNotFound** output terminal of DatabaseLookup1 with the **getCustomerInformation: CustomerServicePartner** Callout node.
- ___ d. Click the **keyNotFound** output terminal of DatabaseLookup1 and drag to CustomerServicePartner callout

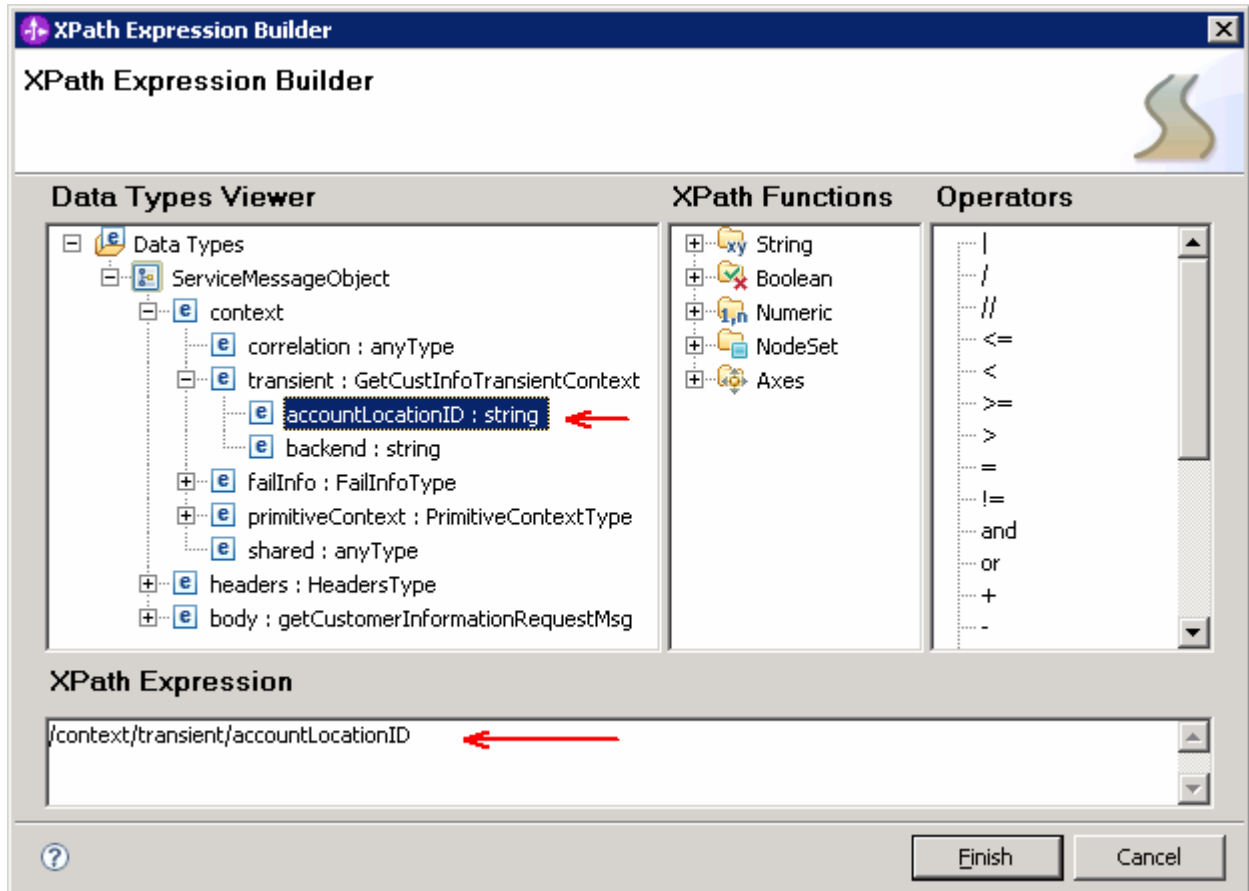


Note: The middle output on DatabaseLookup1 is called the **keyNotFound** output. If a key is not found when a lookup is performed on the database, the message is sent to this output terminal. If a key is not found, you can send that message to the **CustomerServiceOld** backend

- ___ e. In the above image, notice that the bottom node on the right side, **getCustomerInformation: CustomerService InputResponse** node is not wired in this example. InputResponse stops the message and sends it as-is back as the response (in this case not going to the backend module and back through the response side of the mediation flow). Since you do want your message to go to the backend module, you will send the message through the two callout nodes.
- ___ f. Define the data source and database lookup information in the details tab of Properties view. Click on **DatabaseLookup1** and navigate to the **Details tab** of **Properties View**.
- ___ g. Enter the **Database lookup** details as shown below:

Note: You have defined a database with a data source in the administrative console of the WebSphere Enterprise Service Bus server. In this sub-step, you are filling out the information needed to allow this application to use this data source and database.

- Data source name: **jdbc/CustomerRoutingMediation**
- Table Name : **BACKEND_LOCATIONS**
- Key column name: **ACCT_NO_PREFIX**
- Key path: Click the 'Edit' button to define which element to use. The '**XPath Expression Builder**' opens.
 - In the '**Data Types Viewer**' section, expand **Data Types** → **ServiceMessageObject** → **context** → **transient** and then double click **accountLocationID** to add to the **XPath Expression** text area as shown below:



➤ Click **Finish**

- In the Data Elements table, enter the backend element from the transient context as your message element you want to lookup.

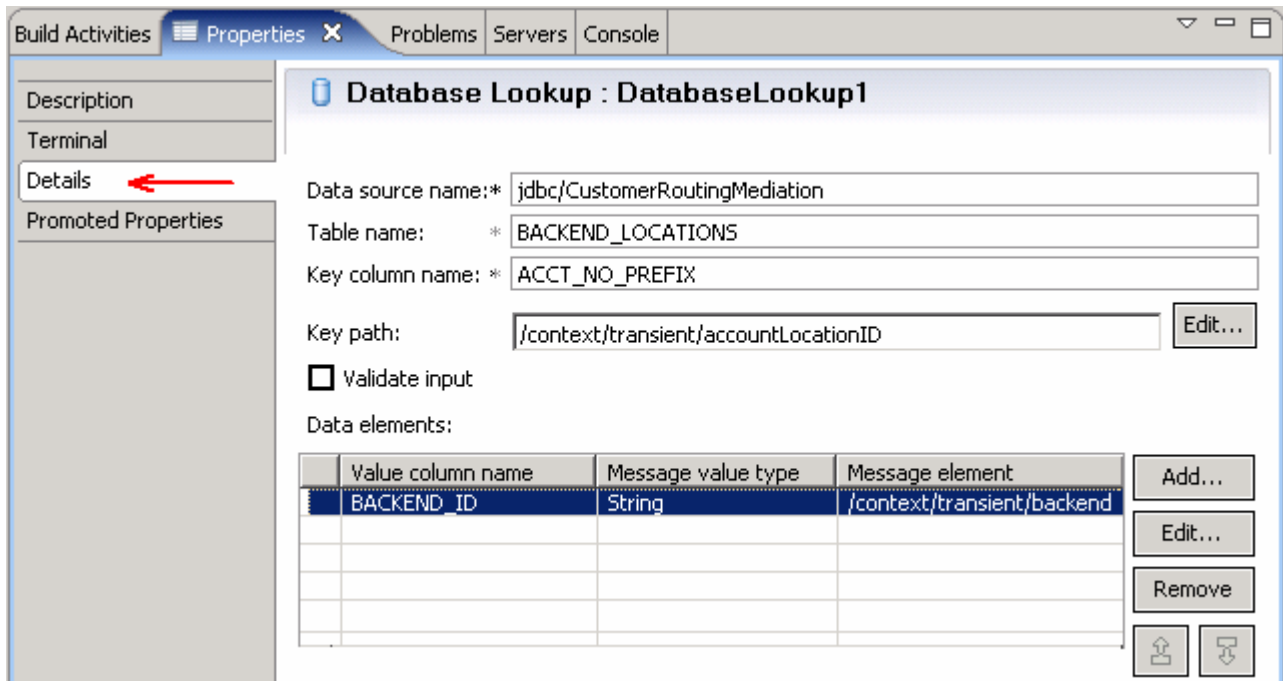
➤ Click the **Add** button next to **Data Elements** table.

➤ In the '**Add/Edit properties**' panel, enter the values listed below:

- Value column name: **BACKEND_ID**
- Message value type: **String** (default)
- Message element: Click the '**Edit**' button to launch the **XPath Expression Builder**
- In the '**Data Types Viewer**' section expand **Data Types** → **ServiceMessageObject** → **context** → **transient** and then double click **backend** to add to the **XPath Expression** text area.
- Click **Finish**.

➤ Click **Finish** over the **Add/Edit properties** panel.

___ h. The final details properties must look like the picture shown below:



__ i. Save your work (**File → Save** or **Ctrl + S**)

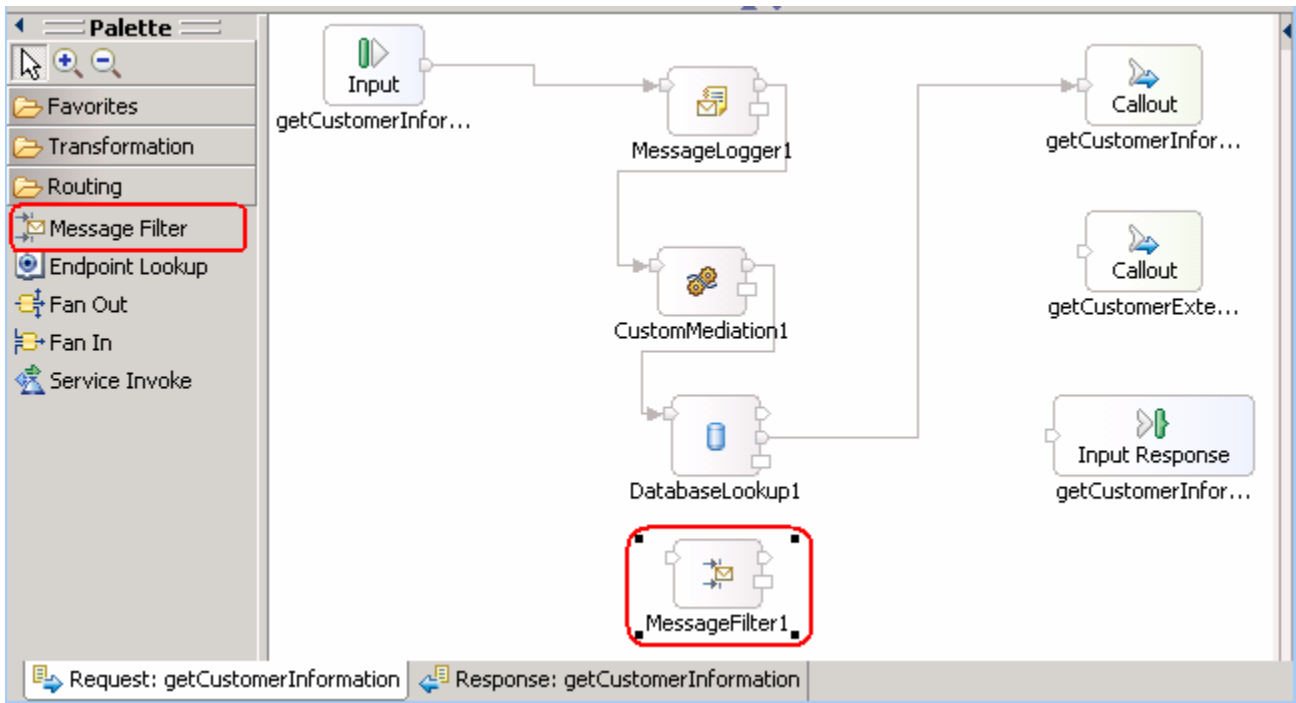
__ j. You finished defining a Database lookup mediation primitive that looks up the backend identifier.

Part 5: Add a message filter mediation primitive to mediation flow

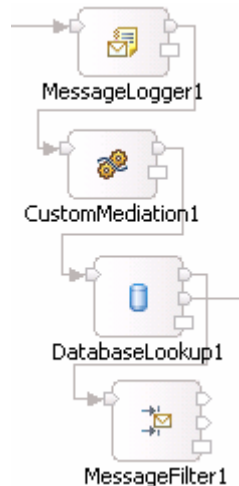
In this section you will determine the message routing based on backend identifier by adding a Message Filter primitive to the Mediation Flow. The old backend will go directly to the callout for CustomerService and the new backend will go to XSL Transformation mediation primitive (XSL Transformation is explained in next section).

___ 1. Drop a Message Filter mediation primitive on the Mediation Flow to create a new message filter.

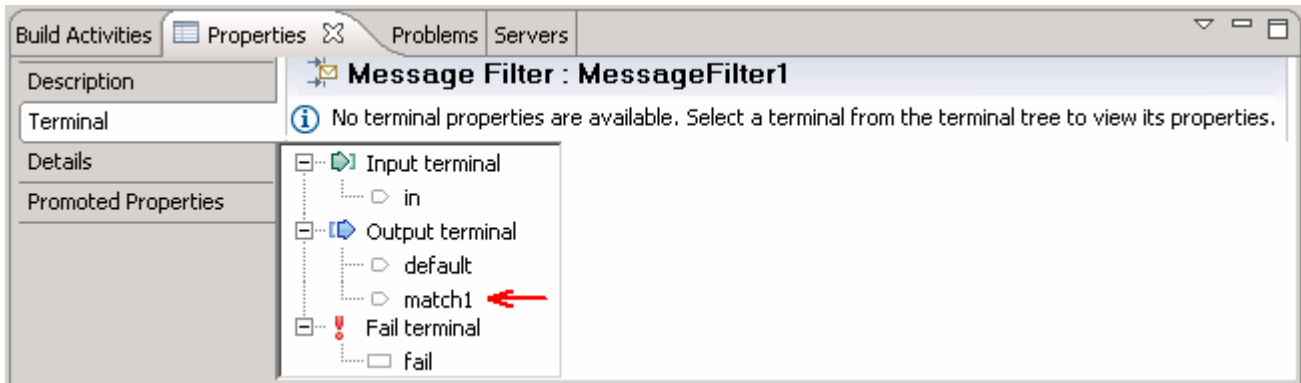
___ a. Click the message filter icon ( Message Filter) from the routing palette (left-hand side) and then click under DatabaseLookup1. The **MessageFilter1** primitive will appear.



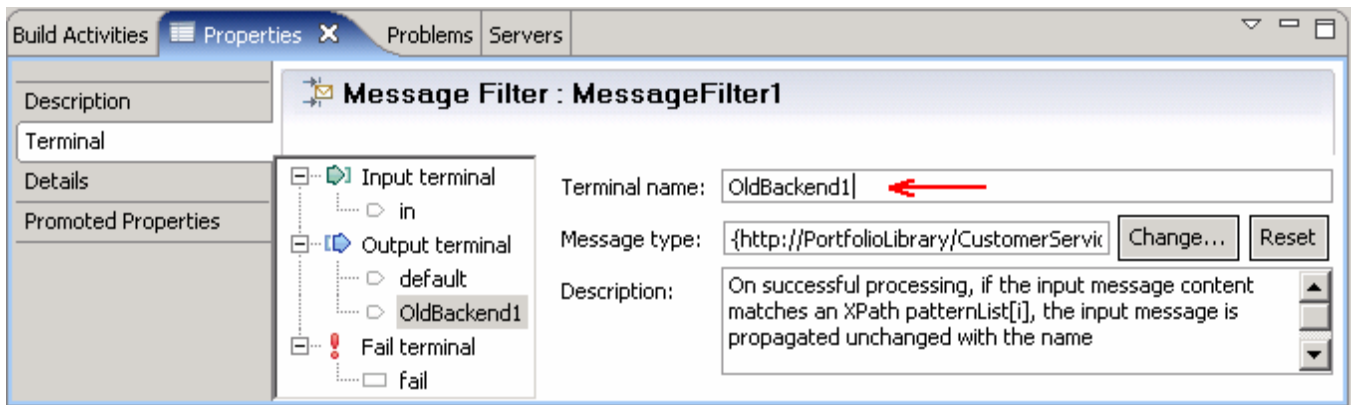
___ b. Wire the output terminal of **DatabaseLookup1** to the input terminal of **MessageFilter1**. Click on **output terminal** of DatabaseLookup1 and drag a wire to the **input terminal** of MessageFilter1.



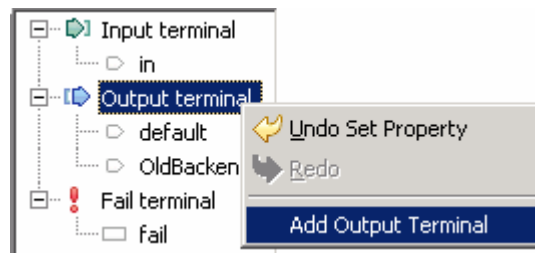
- ___ c. Add **OldBackend1** and **NewBackend1** output terminals to MessageFilter1. Click on MessageFilter1 and open the **Terminal** tab of the **Properties** View.



- ___ d. Select the 'match1' terminal and change the name to **OldBackend1**

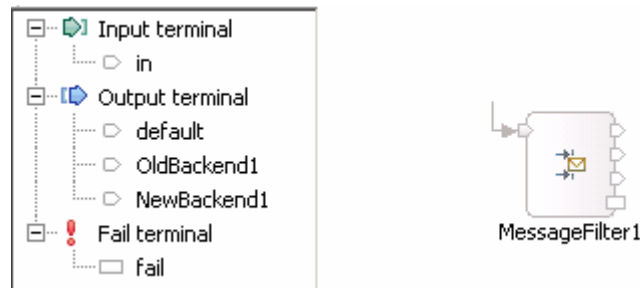


- ___ e. Now, right-click on **Output terminal** and select **Add Output Terminal** from the pop-up menu



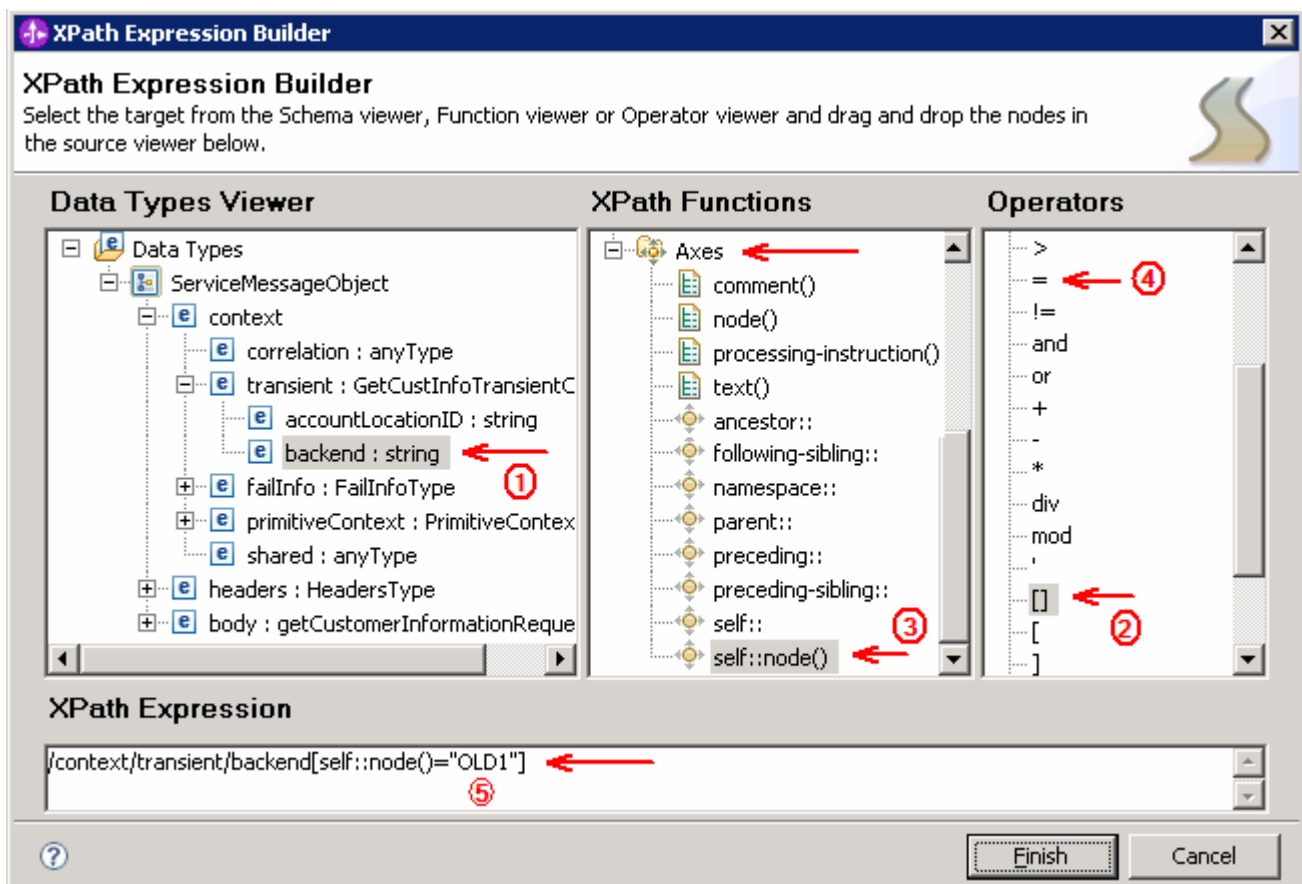
- ___ f. Accept terminal type as 'match', but change the **Terminal Name** to **NewBackend1**. Click **OK**.

- ___ g. You should now have **three** output terminals on MessageFilter1 as shown below:



- ___ h. Add two filter options on Details tab to determine the message routing based on the backend identifier/pattern. Click on the **Details tab** of the **Properties View** while the MessageFilter1 is selected. Here is where you provide the filter options for the message filter to make decisions on which output to send the message to, NewBackend1 or OldBackend1.
- ___ i. Click on the **Add button** next to the Filters table.
- ___ j. In the 'Add/Edit properties' panel, click the 'Edit' button to launch the **XPath Expression Builder**
- ___ k. In the 'Data Types Viewer' section, expand **Data Types** → **ServiceMessageObject** → **context** → **transient** and then double click **backend** to add to the **XPath Expression** text area.
- ___ l. In the **XPath Expression** text area, append the condition, **[self::node()="OLD1"]** to the **/context/transient/backend** expression

Note: You can use **XPath Functions** and **Operators** to build the condition. Alternatively, you can use content assist by placing the cursor at the end of the expression and then pressing 'Ctrl + Space bar' keys from the key board.

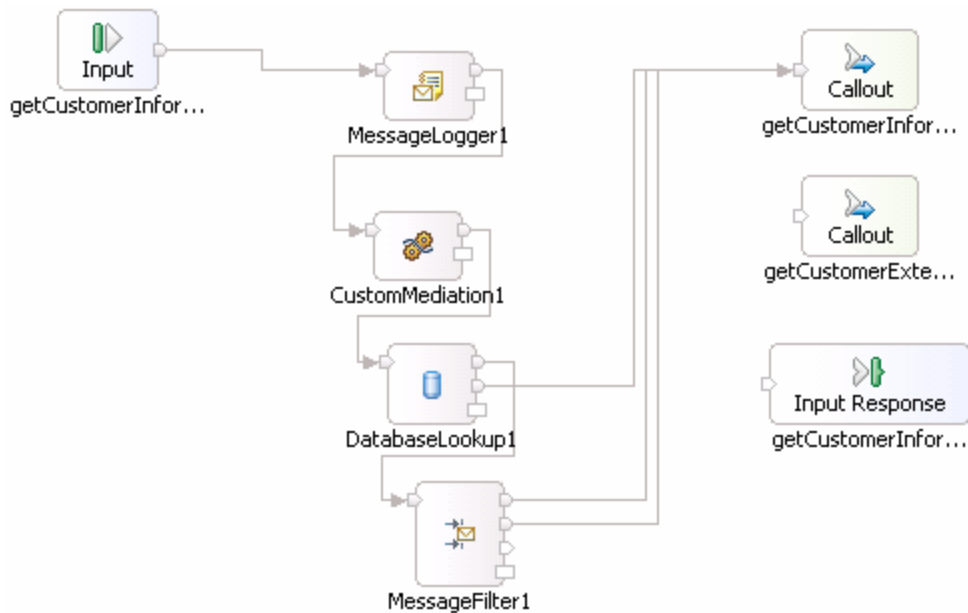


- ___ m. Click **Finish**.
- ___ n. Select **OldBackend1** for Terminal Name.
- ___ o. Click **Finish**.

- ___ p. Repeat this again to add a filter option for “**NEW1**”. Click on the **Add** button on next to the Filters table.
- ___ q. In the ‘**Add/Edit properties**’ panel, click on the ‘**Edit**’ button to launch the **XPath Expression Builder**
- ___ r. In the ‘**Data Types Viewer**’ section, expand **Data Types → ServiceMessageObject → context → transient** and then double click **backend** to add to the **XPath Expression** text area.
- ___ s. In the **XPath Expression** text area, append the condition, **[self::node()=“NEW1”]** to the **/context/transient/backend** expression.
- ___ t. Click the drop-down box for ‘Terminal Name’ and change it to **NewBackend1**.
- ___ u. Click **Finish**. The filters table is as shown below:

Pattern	Terminal name
/context/transient/backend[self::node()="OLD1"]	OldBackend1
/context/transient/backend[self::node()="NEW1"]	NewBackend1

- ___ v. Save your work (**File → Save** or **Ctrl + S**).
- ___ w. Wire the **default** and **OldBackend1** output terminals to **CustomerServicePartner**. In the mediation flow diagram, click on the **default output terminal** of MessageFilter1 and drag it to the **CustomerServicePartner Callout**.
- ___ x. Click on the **OldBackend1** output terminal of MessageFilter1 and drag it to the **CustomerServicePartner Callout**.



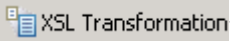
- ___ y. The **NewBackend1** output terminal is now reserved to go to an XSL Transformation since you need to change the CustomerService business object to a CustomerServiceExtended business object.

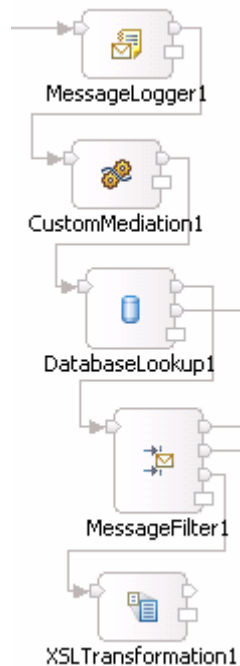
___ 2. Save all changes. (**File → Save All** or **Ctrl +Shift + S**)

Part 6: Add XSL transformation mediation primitives to mediation flow

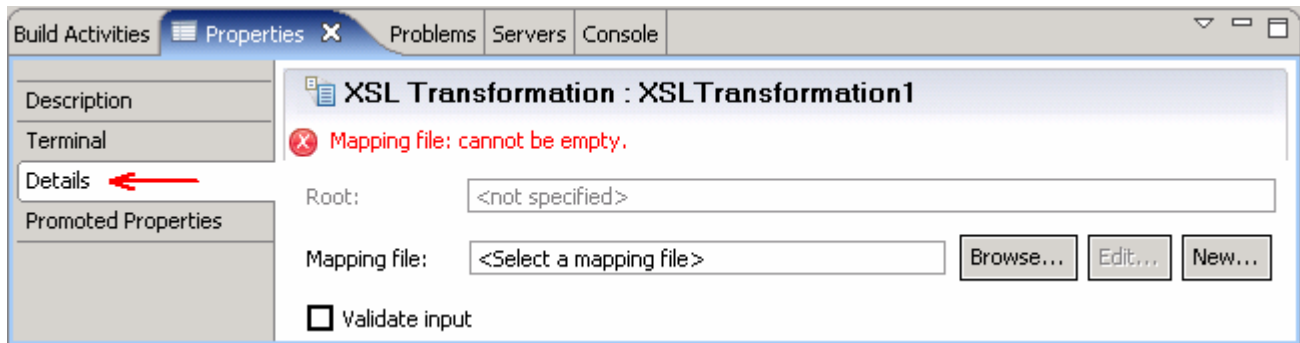
In this section you will create two XSL Transformations that will transform the CustomerService business object into the CustomerServiceExtended business object, then pass the newly formed message to the callout for CustomerServiceExtendedPartner instead of the CustomerServicePartner. Notice that this is just the request. There is also a flow for the response. On the response side, the CustomerServicePartner callout response will go directly to the CustomerService input response. However, the CustomerServiceExtended callout response will need to go back through XSL Transformation mediation in order to transform the response body from the CustomerServiceExtended business object back to the CustomerService business object. Feeding the CustomerServiceExtended business object to application ends in an error because the application only knows how to work with the CustomerService business object. This is the reason why you need an XSL Transformation mediation primitive; to map one business object to another.

___ 1. Drop an XSL Transformation mediation primitive on the Mediation Flow to create a new XSL Transformation.

- ___ a. Click the XSL Transformation icon () from the transformation palette (left-hand side) and then click under MessageFilter1. The XSLTransformation1 primitive will appear.
- ___ b. Wire the output terminal of MessageFilter1 to the input terminal of XSLTransformation1. Click on **output terminal** (NewBackend1) of MessageFilter1 and drag a wire to the **input terminal** of XSLTransformation1.



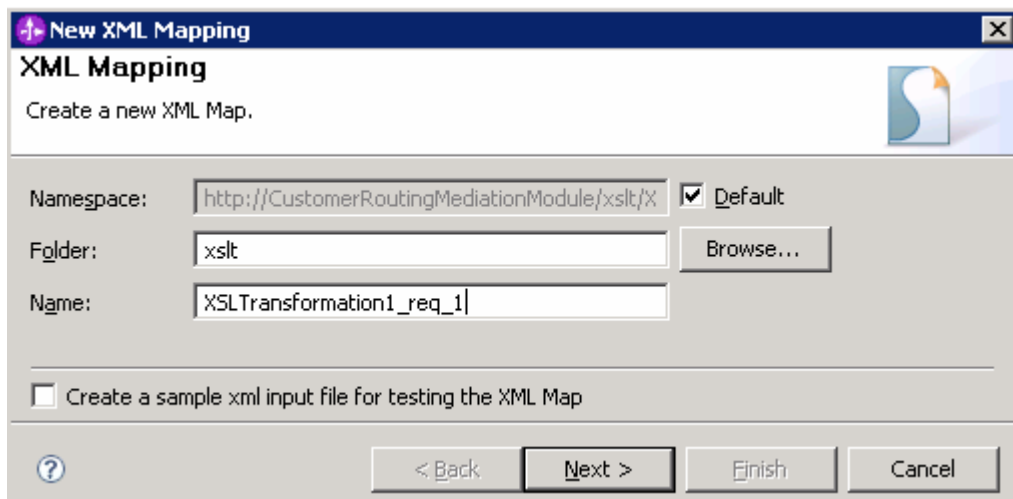
- ___ c. Create a new mapping for this XSL Transformation. Select **XSLTransformation1** and open the **Details tab** of the **Properties View**.



__ d. Click the **'New...'** button on the Mapping File row to launch the **'XML Mapping'** wizard

1) Enter the parameters in the **'XML Mapping'** panel

- Folder : **xslt** (default)
- Name : **XSLTransformation1_req_1** (default)

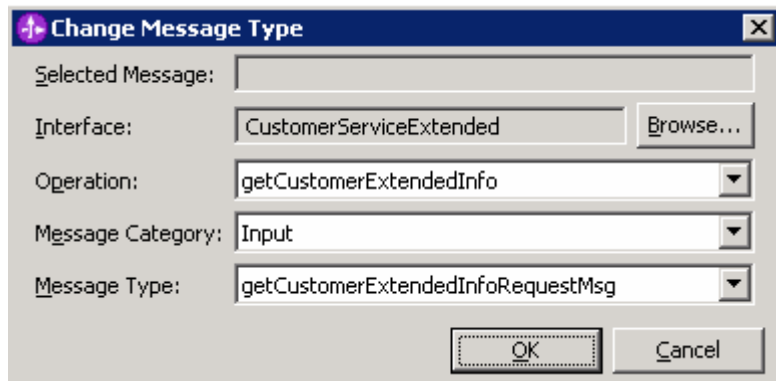


2) Click **Next**

3) In the **'Specify Message Types'** panel, enter the parameters listed below:

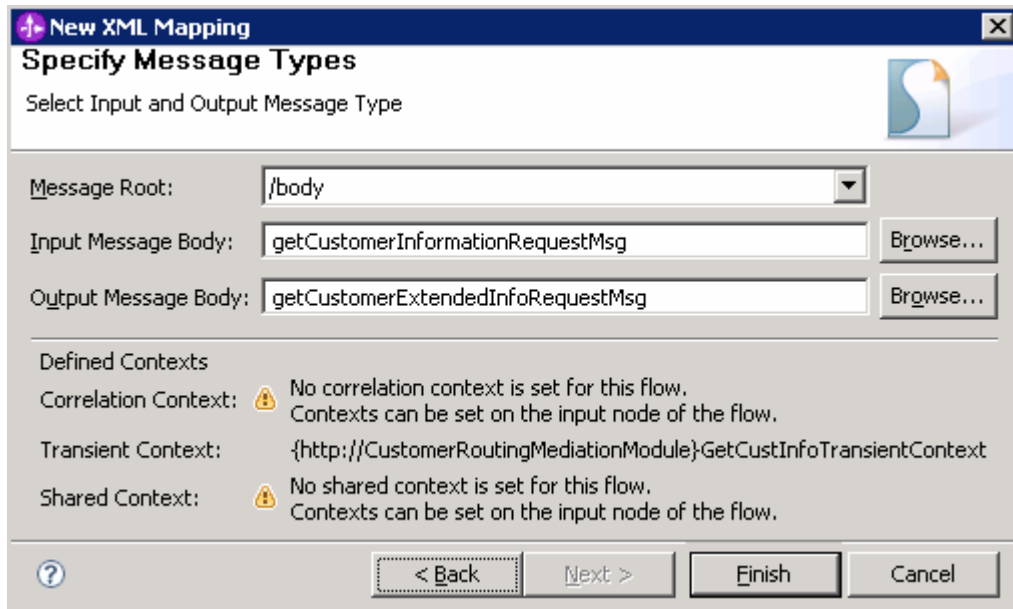
- Select **/body** for the Message Root field
- Since you wired the MessageFilter1 to XSLTransformation1, the input message body is already assigned **getCustomerInformationRequestMsg**
- Click the **'Browse...'** button next to 'Output Message Body'. The **'Change Message Type'** panel opens.
- On the 'Change Message Type' panel, click the **'Browse...'** button to open a list of interfaces. Select **CustomerServiceExtended** and click the **OK** button.
- For 'Operation', click the drop-down list to and select **getCustomerExtendedInfo** operation.
- For 'Message Category', click the drop-down list and select **Input** since you are still working on the request side.

- The 'Message Type' gets populated when **input** or **output** (request or response) is selected from the message category. The Change Message Type panel should look like in the picture below:



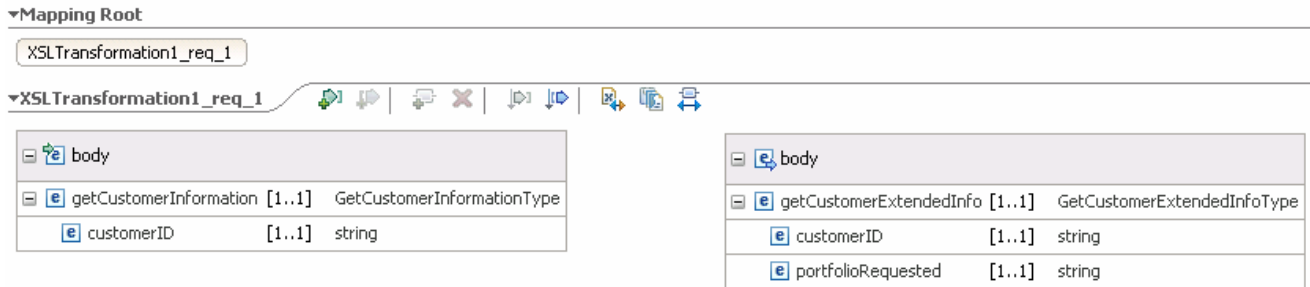
- Click **OK**

- 4) Now you should see **getCustomerInformationRequestMsg** populated for the Input Message Body and **getCustomerExtendedInfoRequestMsg** for the Output Message Body.

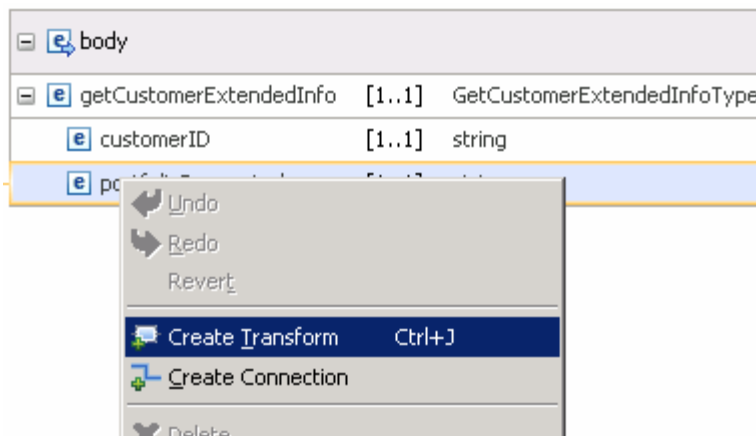


- 5) Click **Finish**.

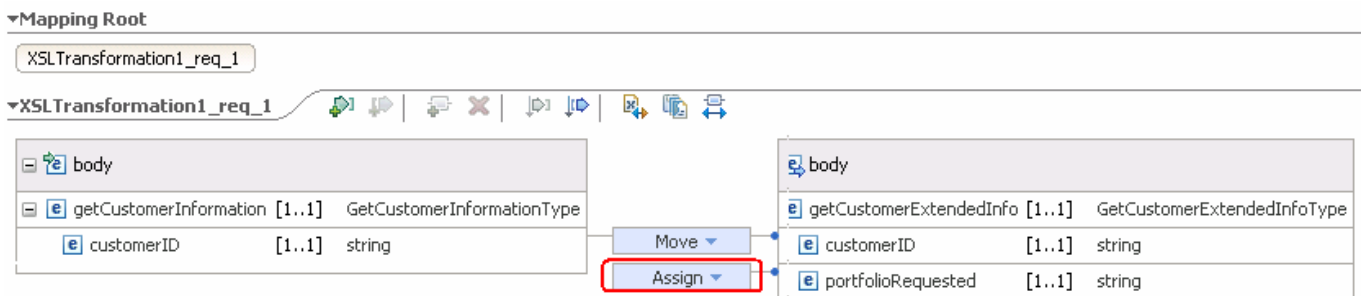
- e. The mapping file, **XSLTransformation1_req_1.map** is created and opened in the editor. Completely expand both the **Source** and **Target objects** as shown below:



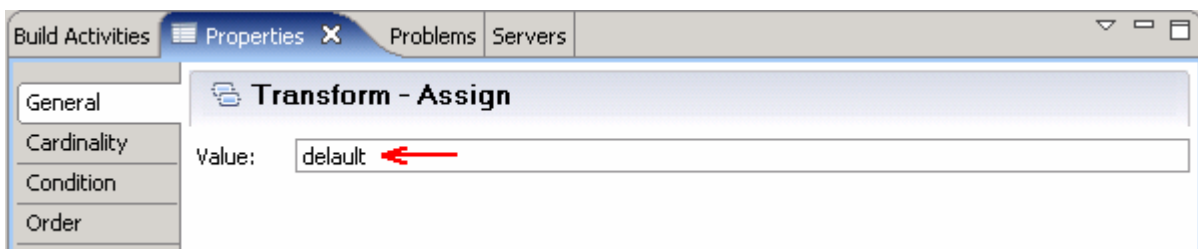
- ___ f. Click on **customerID** from the **Source** side and drag a wire to the **customerID** on to the **Target** side.
- ___ g. Since **portfolioRequested** is a new element for the target, you are going to assign a default value to fill **portfolioRequested** with something. You are going to put the string “default” in for this value. Right-click on **portfolioRequested** and select **Create Transform** from the pop-up menu as shown below:



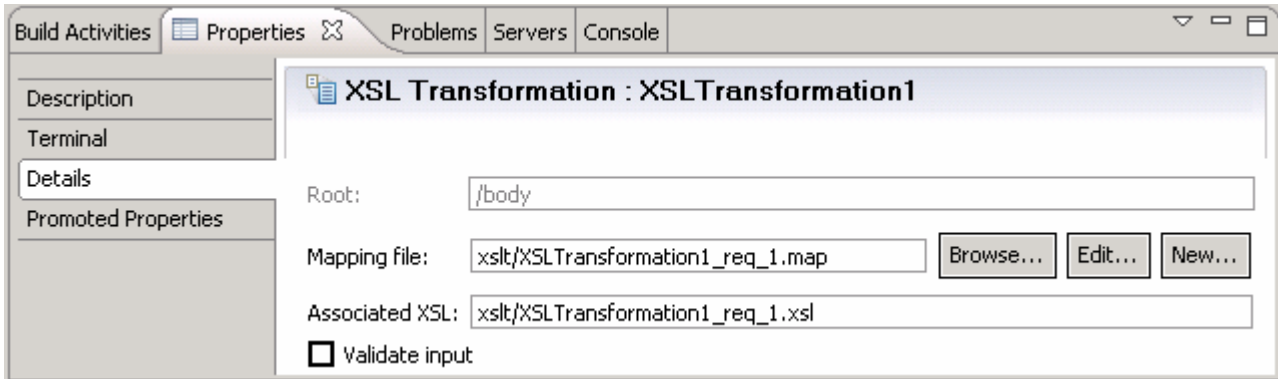
- ___ h. This creates an ‘Assign’ transform as shown below:



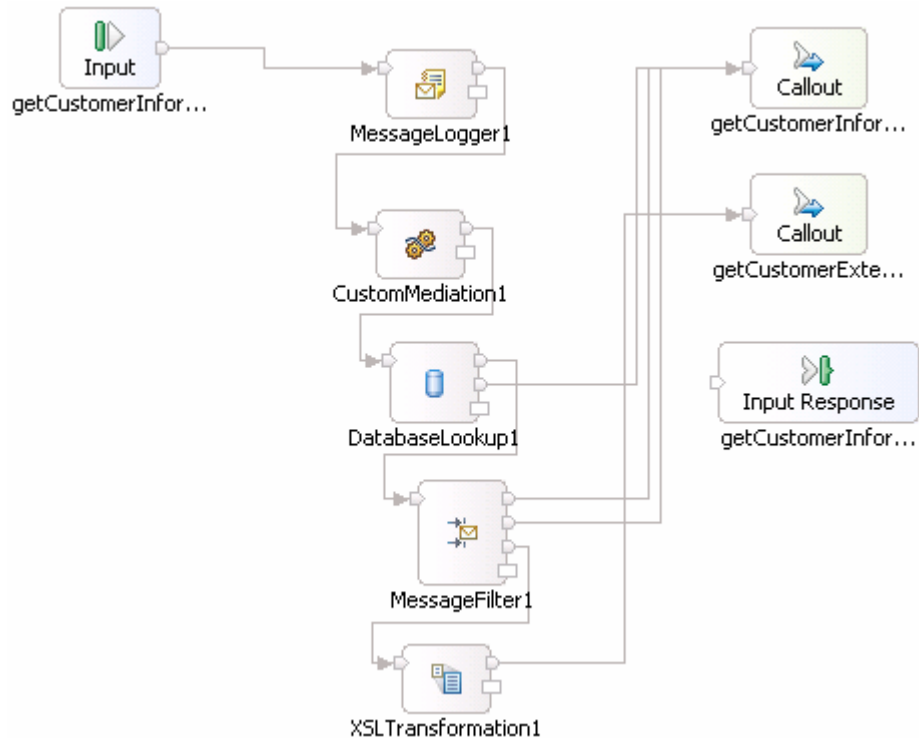
- ___ i. Select the ‘Properties → General tab for the ‘Assign’ transform



- __ j. Enter **'default'** for Value
- __ k. Save the mapping (**File → Save** or **Ctrl + S**). **Close** mapping file.
- __ l. The details for the XSL Transformation, XSLTransformation1 should look like in the picture shown below:



- __ m. Save the Mediation Flow (**File → Save** or hit **Ctrl + S**).
- __ n. Wire the output terminal of XSLTransformation1 to the **getCustomerExtendedInfo: CustomerServiceExtendedPartner** Callout node. Click on output terminal of **XSLTransformation1** and drag to the **getCustomerExtendedInfo: CustomerServiceExtendedPartner** Callout node.




- __ o. Save the Mediation Flow (**File → Save** or hit **Ctrl + S**).

- ___ 2. Add an XSL Transformation to the response side of Mediation flow. This action turns the CustomerServiceExtended business object back into a CustomerService business object that the application knows how to work with. Click on the **Response tab** of mediation flow.

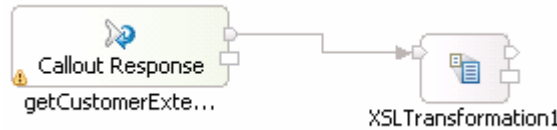


- ___ a. Add an XSL Transformation mediation primitive on the Mediation Flow to create a new XSL Transformation.

- 1) Click the XSL Transformation icon ( XSL Transformation) from the transformation palette (left-hand side) and then click on the flow editor canvas. The XSLTransformation1 primitive will appear.

- ___ b. Wire the **callout** response for **CustomerServiceExtendedPartner** to the **input** terminal of **XSLTransformation1**.

- 1) Click on **output terminal** of CustomerServiceExtendedPartner and drag a wire to the **input terminal** of XSLTransformation1.

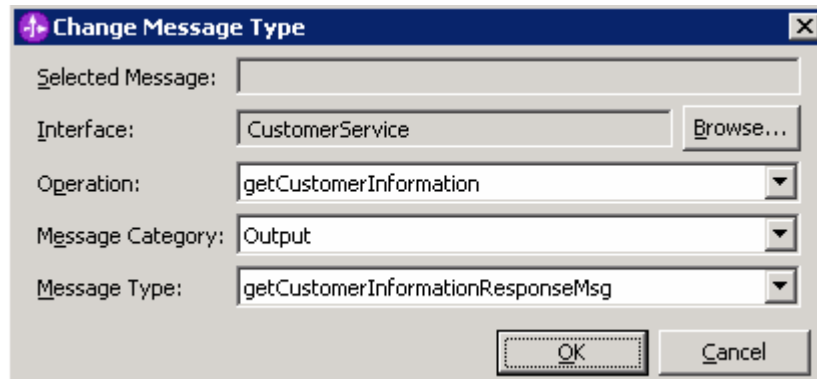


- ___ c. Create a new mapping for this XSL Transformation. Select **XSLTransformation1** and open the **Details tab** of the **Properties View**.

- ___ d. Click the **'New...'** button on the Mapping File row to launch the **'XML Mapping'** wizard

- 1) Enter the parameters in the **'XML Mapping'** panel
 - Folder : **xslt** (default)
 - Name : **XSLTransformation1_res_1** (default)
- 2) Click **Next**
- 3) In the **'Specify Message Types'** panel, enter the parameters listed below:
 - Select **/body** for the Message Root filed
 - Since you wired the MessageFilter1 to XSLTransformation1, the input message body is already assigned **getCustomerExtendedInfoResponseMsg**
 - Click the **'Browse...'** button next to 'Output Message Body'. The **'Change Message Type'** panel opens.
 - On the 'Change Message Type' panel, click the **'Browse...'** button to open a list of interfaces. Select **CustomerService** and click the **OK** button.
 - For 'Operation', click the drop-down list to and select **getCustomerInformation** operation.
 - For 'Message Category', click the drop-down list and select **Output** since you are working on the response side.

- The 'Message Type' gets populated when **input** or **output** (request or response) is selected from the message category. The Change Message Type panel should look like in the picture below:



- Click **OK**

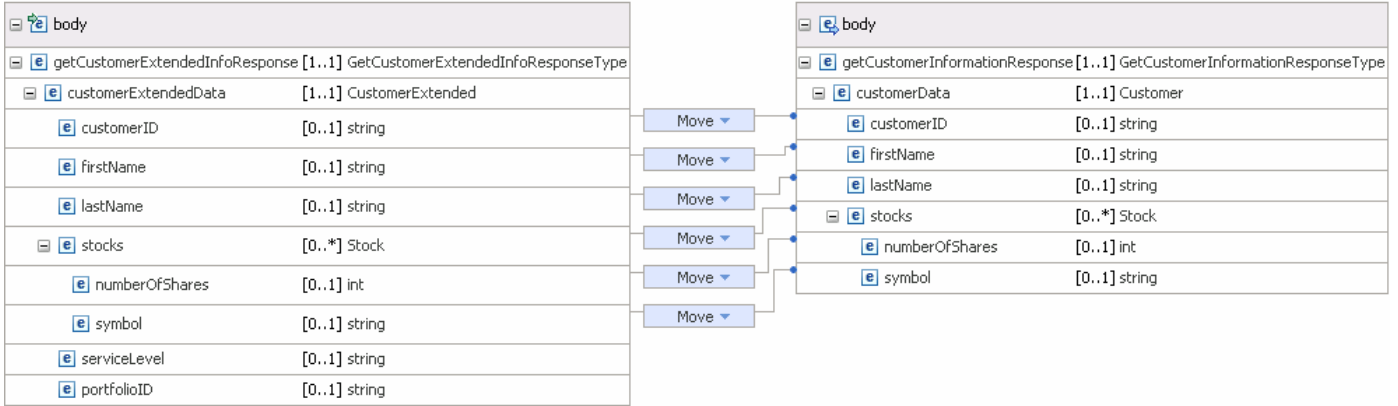
- Now you should see **getCustomerExtendedInfoResponseMsg** populated for the Input Message Body and **getCustomerInformationResponseMsg** for the Output Message Body.
- Click **Finish**.

___ e. The mapping file, **XSLTransformation1_res_1.map** file is created and opened. Completely expand both the **Source** and **Target** objects as shown below:

body	
getCustomerExtendedInfoResponse	[1..1] GetCustomerExtendedInfoResponseType
customerExtendedData	[1..1] CustomerExtended
customerID	[0..1] string
firstName	[0..1] string
lastName	[0..1] string
stocks	[0..*] Stock
numberOfShares	[0..1] int
symbol	[0..1] string
serviceLevel	[0..1] string
portfolioID	[0..1] string

body	
getCustomerInformationResponse	[1..1] GetCustomerInformationResponseType
customerData	[1..1] Customer
customerID	[0..1] string
firstName	[0..1] string
lastName	[0..1] string
stocks	[0..*] Stock
numberOfShares	[0..1] int
symbol	[0..1] string

- Click on **customerID** from the **Source** side and drag a wire to the **customerID** on the **Target** side.
- Similarly connect (from source to target) **firstName**, **lastName**, **stocks**, **numberOfShares**, and **symbol**. Since the node, **stocks** is an array, you have to map the array elements so that each array and array element is transformed.

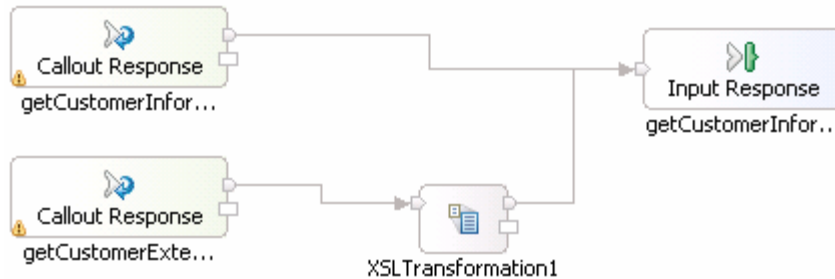


3) Save the mapping file (**File → Save** or **Ctrl + S**). **Close** the mapping file.

4) Save the Mediation Flow (**File → Save** or **Ctrl + S**).

___ f. Wire the output terminal of **XSLTransformation1** to the **getCustomerInformation: CustomerService** InputResponse.

1) Click on output terminal of XSLTransformation1 and drag to the **getCustomerInformation: CustomerService** InputResponse node.



___ g. Save the Mediation Flow (**File → Save** or **Ctrl + S**).


Part 7: Test the CustomerServiceExtended backend

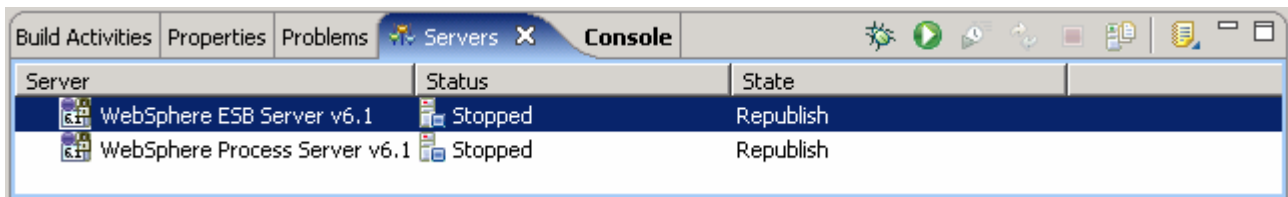
In this section you will run a JSP that enters a customerID to start off the process. However, for Lab-2 you are going to test the new CustomerServiceExtended backend instead of the old CustomerService backend.

- ___ 1. Start **WebSphere Enterprise Service Bus Server** and **add modules** to server

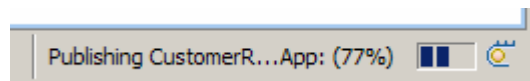
If using a remote testing environment, follow the instructions in [Task: Adding Remote Server to WebSphere Integration Developer Test Environment](#) at the end of this document, to start the remote server.

If using a local testing environment:

- ___ a. Open **Servers View**.
- ___ b. Highlight WebSphere ESB Server v6.1 and click **Start button** ().



- ___ c. This will take some time. Wait for the server to start
- ___ 3. Add **projects** to WebSphere Enterprise Service Bus Server (once the ESB Server is started, not before).
- ___ a. In Servers view, right-click on '**WebSphere ESB Server v6.1**' and select '**Add and Remove Projects...**'
- ___ b. Click **Add-All>>** button to move all projects to server and click **Finish** button
- ___ c. Wait for the deployment to finish. While the project is deploying you will see something like the following in the lower right corner of WebSphere Integration Developer.



- ___ d. Click the Web browser icon in the WebSphere Integration Developer tools panel to launch a browser



- ___ e. Enter [http:// <HOSTNAME>:<PORT>/PortfolioManagerClient/index.jsp](http://<HOSTNAME>:<PORT>/PortfolioManagerClient/index.jsp) . Where hostname is the name of the system where the WebSphere Enterprise Service Bus server is located. Port is the **WC_defaulthost** port of the WebSphere Enterprise Service Bus profile.

Ex: <http://localhost:9080/PortfolioManagerClient/index.jsp>

Note: You can get the **WC_defaulthost** port by going to **serverindex.xml** file in <WID_HOME>\pf\esb\config\cells\esbCell\nodes\esbNode. Where WID_HOME is the location where WebSphere Integration Developer is installed


Ex: <ESB_PROFILE_HOME>\config\cells\esbCell\nodes\esbNode.

- __ f. Enter **7777777** (7 seven times) in text input box and click **Submit** to get a response displayed to the JSP and to the Console View of WebSphere Integration Developer. You should see the output as "The value is: 3410.0.0".

Portfolio Application

Enter customer ID:

The value is: 3410.0

- __ g. **Close** the browser.
- __ h. Stop 'WebSphere ESB Server v6.1' by highlighting the WebSphere ESB Server v6.1 in the Servers View and click **Stop** button ().

Part 8: Save work and clean up server

- ___ 1. Export project as Project Interchange file
 - ___ a. Navigate to **File → Export**.
 - ___ b. Select **Project Interchange**.
 - ___ c. Out of all the projects listed, you only need to add a check next to 6 projects.

CustomerBackend
CustomerRoutingMediationModule
PortfolioLibrary
PortfolioManager
PortfolioManagerClient
StockQuoteManager

All other projects are generated upon import of the project interchange into a workspace.
 - ___ d. Save in C:/LabFiles61/WESB/Lab2/
 - ___ e. Name the project interchange **WPIv61_ESB_FinishedLab2_PI.zip**.

If you are not continuing to Lab 3 right now, go ahead and clean up the **ESB Server**.

1. Start **WebSphere ESB Server v6.1** from the Servers view of the Business Integration perspective.
2. Right-click on WebSphere ESB Server v6.1 (once started) and select **Add and Remove projects...**
3. Select **Remove-All** and click **Finish**.
4. After remove is done, **stop** the WebSphere ESB Server v6.1.

What you did in this exercise

In this lab you added to the Message Logger mediation primitive with a couple others available in WebSphere Integration Developer V6.1. The main goal for Lab2 was to create another backend, CustomerServiceExtended, and use mediations to help decide which backend to use.

You first added a Custom Mediation that was used to mine out a two digit prefix from the customer ID and place 2 digit it in the transient context (a business object) using a Java snippet. You then added a Database Lookup primitive that uses that two digit prefix from transient context as key to lookup a backend identifier to also place into transient context.

The Customer ID prefixes you used (77) went to a CustomerServiceExtended backend. To determine the message routing based on backend identifier, you added a Message Filter primitive. The old backend went directly to the callout for CustomerService and the new backend went to the XSL Transformation.

The XSL Transformation transformed the CustomerService business object into the CustomerServiceExtended business object, and then passed the newly formed message to the callout for CustomerServiceExtendedPartner instead of the CustomerServicePartner. On the response side, the CustomerServicePartner callout response went directly to the CustomerService input response. However, the CustomerServiceExtended callout response needed to go back through XSL Transformation mediation in order to transform the response body from the CustomerServiceExtended business object back to the CustomerService business object.

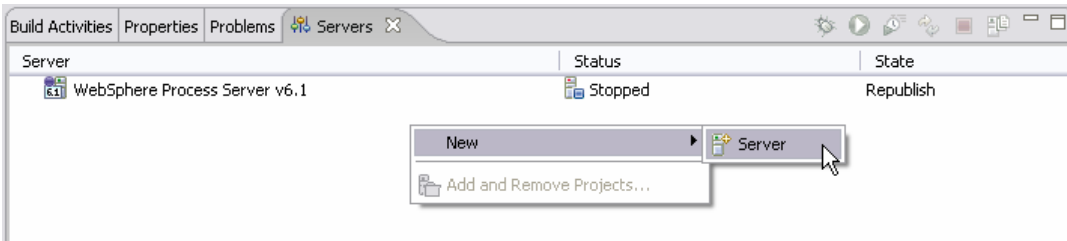
Solution instructions

- ___ 1. Import **Solution** Project Interchange file.
 - ___ a. With a blank workspace in WebSphere Integration Developer, Go to **File → Import → Project Interchange**.
 - ___ b. Click on top Browse button and navigate to **<LAB_FILES>WPIv61_ESB_Lab2_solution_Pi.zip**
 - ___ c. Click **Finish** button.
- ___ 2. Add the data source from the first section of **Part 4**, the Database Lookup.
- ___ 3. Continue past to **Part 7** to test the application.

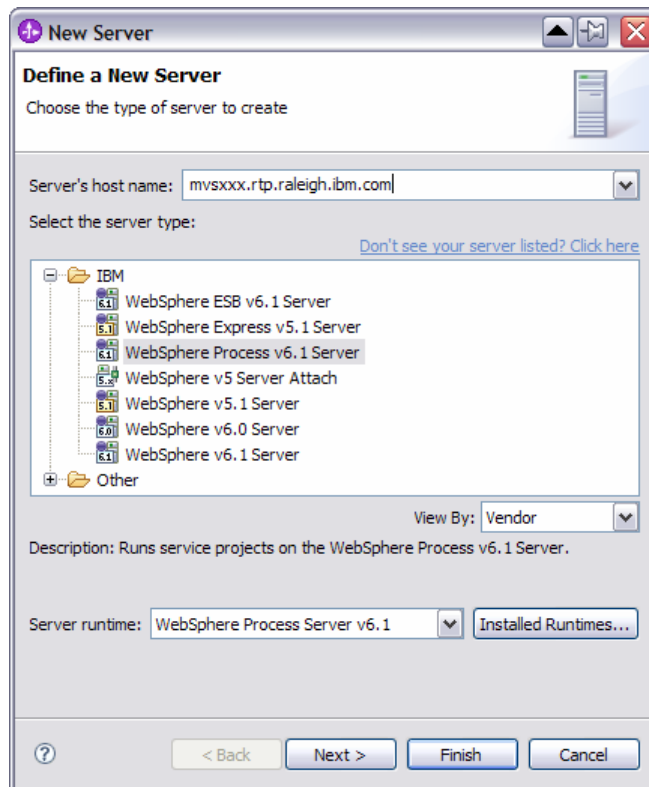
Task: Adding remote server to WebSphere Integration Developer test environment

This task describes how to add a remote server to the WebSphere Integration Developer test environment. This example uses a z/OS machine.

- ___ 1. Define a new remote server to WebSphere Integration Developer.
 - ___ a. Right click on the background of the **Servers** view to access the pop-up menu.
 - ___ b. Select **New → Server**.

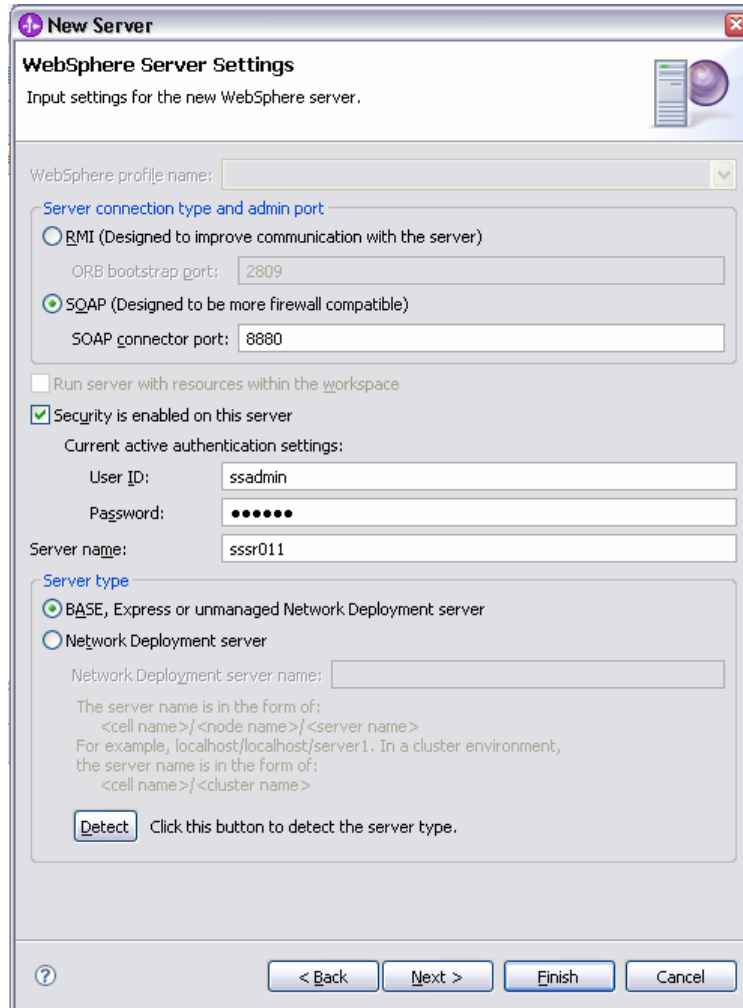


- ___ c. In the New Server dialog, specify the remote server's host name, **<HOSTNAME>**.
- ___ d. Ensure that the appropriate server type, **'WebSphere Process v6.1 Server'** or **'WebSphere ESB v6.1 Server'**, is highlighted in the server type list

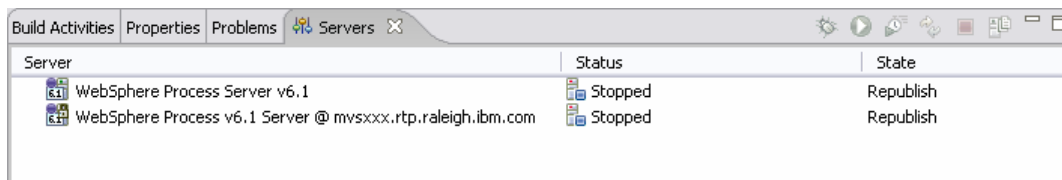


- ___ e. Click **Next**.

- ___ f. On the WebSphere Server Settings page, leave the radio button for **SOAP** selected, changing the **SOAP connector port** to the correct setting (<SOAP_PORT>). If security is on in your server, check the box for 'Security is enabled on this server' and input <USERID> for the user ID and <PASSWORD> for the password.



- ___ g. Click **Finish**.
- ___ h. The new server should be seen in the Server view.



- ___ 2. Start the remote server if it is not already started. WebSphere Integration Developer does not support starting remote servers from the Server View.
- ___ a. From a command prompt, telnet to the remote system if needed:

'telnet <HOSTNAME> <TELNET_PORT>'

User ID : **<USERID>**

Password : **<PASSWORD>**

__ b. Navigate to the bin directory for the profile being used:

cd <WAS_HOME>/profiles/<PROFILE_NAME>/bin

__ c. Run the command file to start the server: **./startServer.sh <SERVER_NAME>**

__ d. Wait for status message indicating server has started:

```
ADMU3200I: Server launched. Waiting for initialization status.
```

```
ADMU3000I: Server c11sr01 open for e-business; process id is 0000012000000002
```