

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.....	7
Part 2: Prepare CustomerRoutingMediationModule for lab 2.....	9
Part 3: Add a custom mediation primitive to mediation flow	13
Part 4: Add a database lookup mediation primitive to mediation flow	19
Part 5: Add a message filter mediation primitive to mediation flow	25
Part 6: Add XSL transformation mediation primitives to mediation flow	29
Part 7: Test the CustomerServiceExtended backend.....	36
Part 8: Save work and clean up server.....	39
What you did in this exercise	40
Solution instructions	41
Task: Adding remote server to WebSphere Integration Developer test environment	42

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 V6.0.2 with the WebSphere Enterprise Service Bus test server option installed.

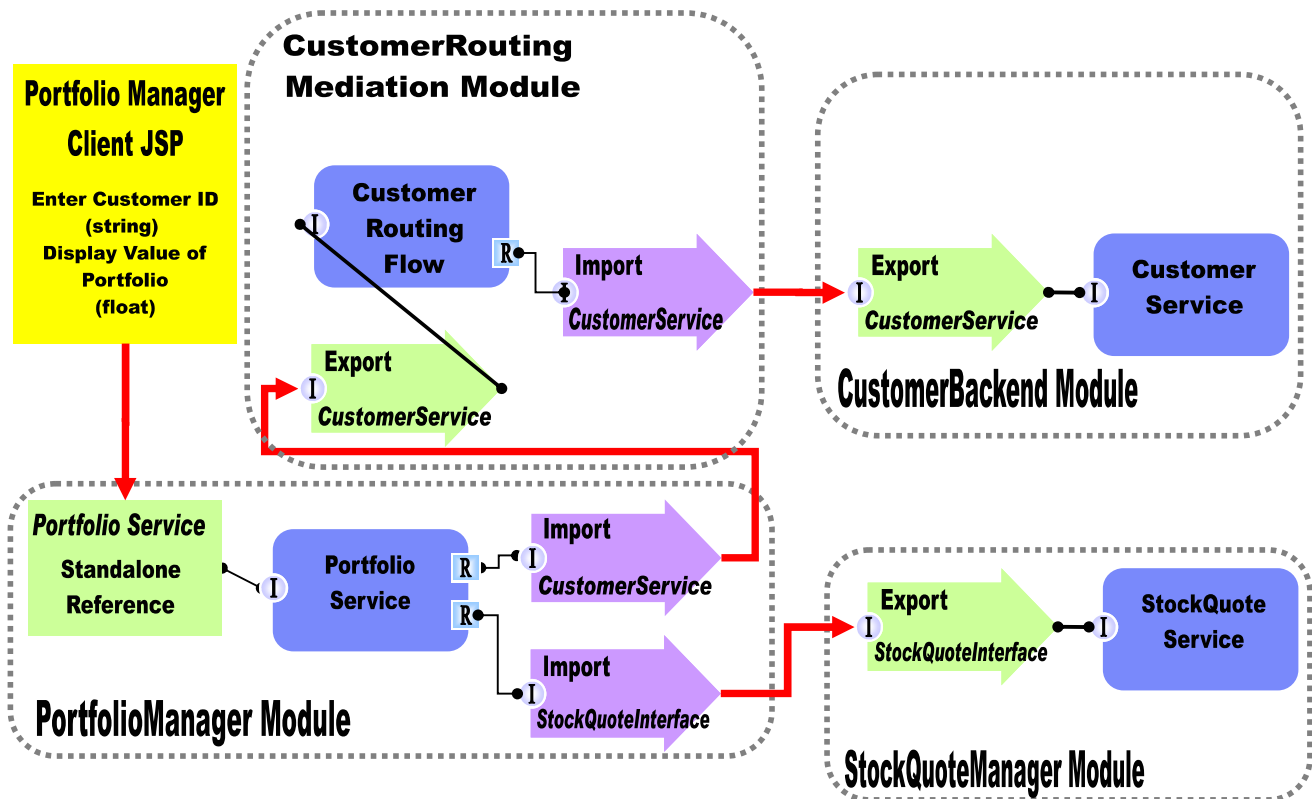
What you should be able to do

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

- Import project interchange files into the WebSphere Integration Developer V6.0.2 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.

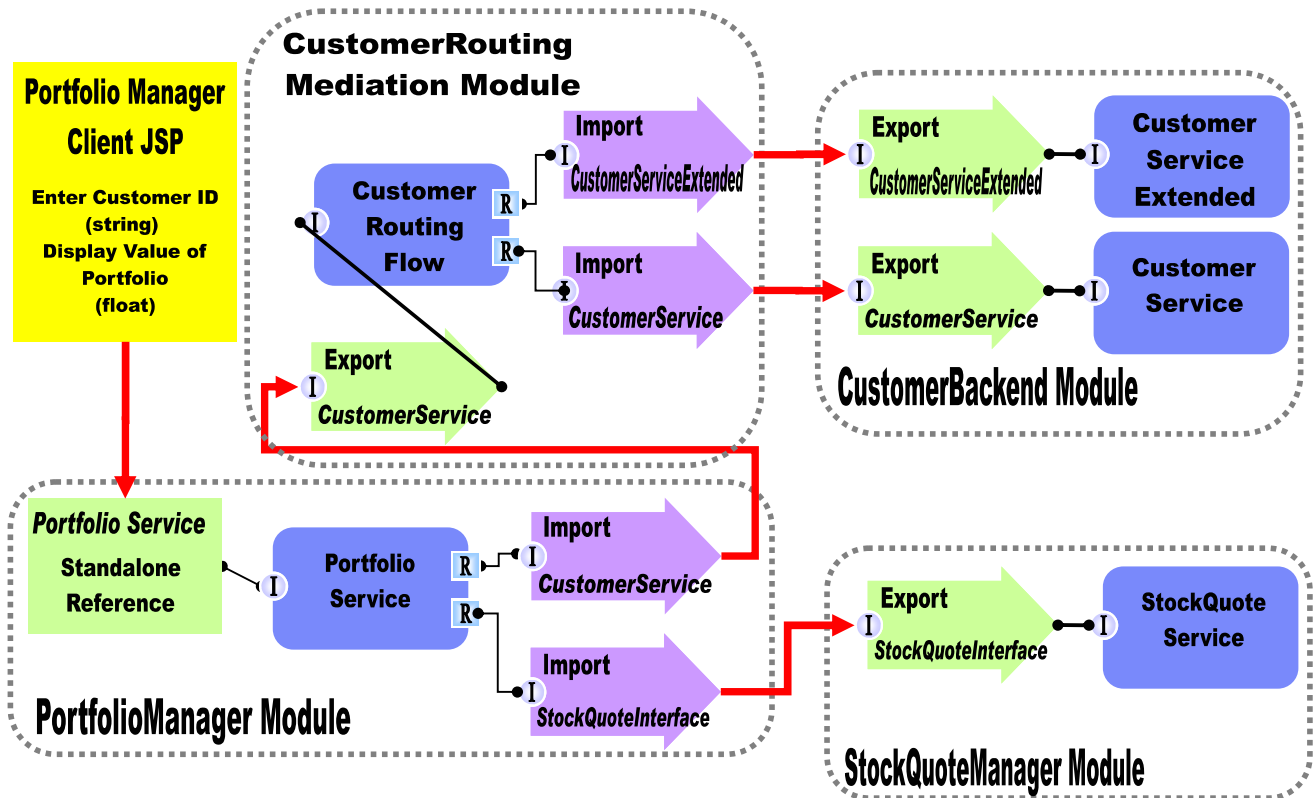
Introduction

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 cloudscape database.



In Lab2, you will add to the Message Logger mediation primitive with a couple others available in WebSphere Integration Developer V6.0.2. The main goal for Lab2 will be 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 (BO) 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 would 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 be using the Administration 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 may be Windows® operating-system specific. If you plan on running the lab on an operating-system other than Windows, you will need to run the appropriate commands, and use appropriate files (.sh vs. .bat) for your operating system. The directory locations are specified in the lab instructions using symbolic references, as follows:

Reference Variable	Windows Location	AIX®/UNIX® Location
<LAB_NAME>	Lab2	Lab2
<WID_HOME>	C:\Program Files\IBM\ID\6.0	/opt/IBM/WebSphere/ID/6.0
<LAB_FILES>	C:\Labfiles602	/tmp/Labfiles602
<WORKSPACE>	C:\Labfiles602\WESB\Lab2\workspace	/tmp/Labfiles602/WESB/Lab2/workspace
<TEMP>	C:\temp	/tmp
<SOLUTION>	C:\Labfiles602\WESB\Lab2\solution	/tmp/Labfiles602/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:\LabFiles602\ would be replaced by C:/LabFiles602//

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:

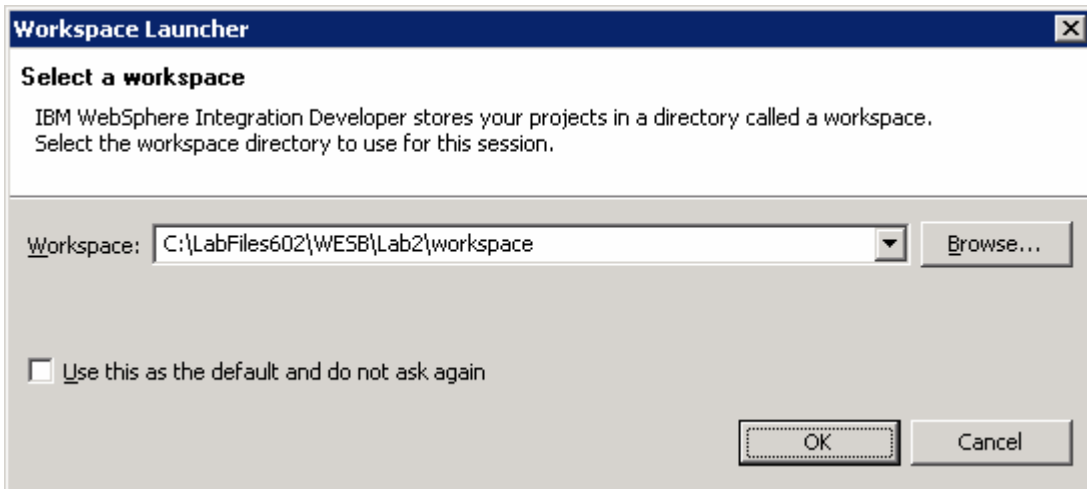
Reference Variable	Example: Remote Windows test server location	Example: Remote z/OS® test server location	Input your values for the remote location of the test server
<SERVER_NAME>	server1	cl1sr01	
<WAS_HOME>	C:\Program Files\IBM\WebSphere\AppServer	/etc/cl1cell/AppServerNode1	
<HOSTNAME>	localhost	mvsxxx.rtp.raleigh.ibm.com	
<BOOTSTRAP_PORT>	2809	2809	
<TELNET_PORT>	N/A	1023	
<PROFILE_NAME>	AppSrv01	default	
<USERID>	N/A	cl1admin	
<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 be importing all projects inside the **WPIv602_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 V6.0.2 with a workspace location of **C:\LabFiles602\WESB\Lab2\workspace**.

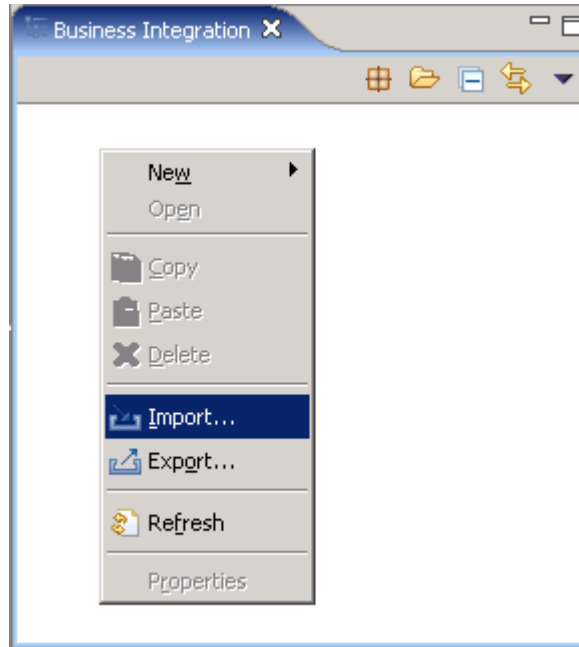


2. On the Welcome window, click the curved arrow at top right to **go to workbench**.

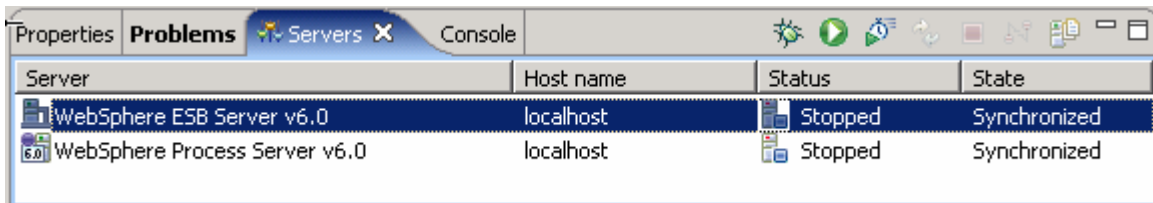


Import Project Interchange file, **WPIv602_ESB_StartLab2_Pi.zip**, into a new workspace.


- a. Right-click inside **Business Integration View** (top left view in the Business Integration Perspective)
- b. Select **Import** from the context menu.

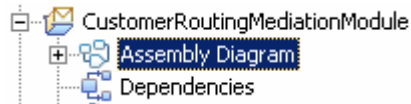


- ___ c. Select **Project Interchange** listed from the Import dialog. Click **Next**.
- ___ d. Click the top **Browse** button for **From zip file**:
- ___ e. Navigate to **C:/LabFiles602/WESB/Lab2/import/WPIv602_ESB_StartLab2_PI.zip**.
- ___ f. Click the **Select All** button to select the all checkboxes for the projects listed. The project that must be listed are **CustomerBackend, CustomerRoutingMediationModule, PortfolioLibrary, PortfolioManager, PortfolioManagerClient** and **StockQuoteManager**
- ___ g. Verify you have **CustomerBackend, CustomerRoutingMediationModule, PortfolioLibrary, PortfolioManager** and **StockQuoteManager** modules listed in the Business Integration view.
- ___ h. Click **Finish** button (projects will be imported and auto-build will run).
- ___ i. Verify you have WebSphere ESB Server v6.0 listed in your Servers view.

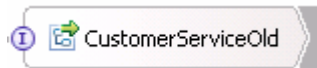


Part 2: Prepare CustomerRoutingMediationModule for lab 2

- ___ 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**.



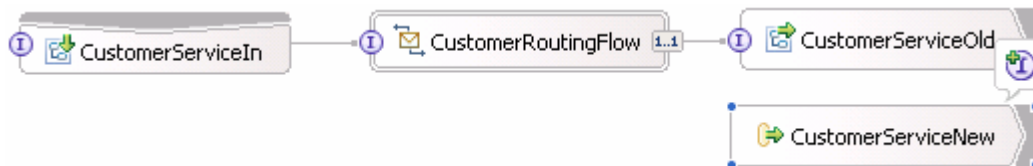
- ___ 3. Add a new Import, **CustomerServiceNew** to the Assembly Diagram of CustomerRoutingMediationModule.

- ___ a. Drag and drop an import on the right-hand 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.

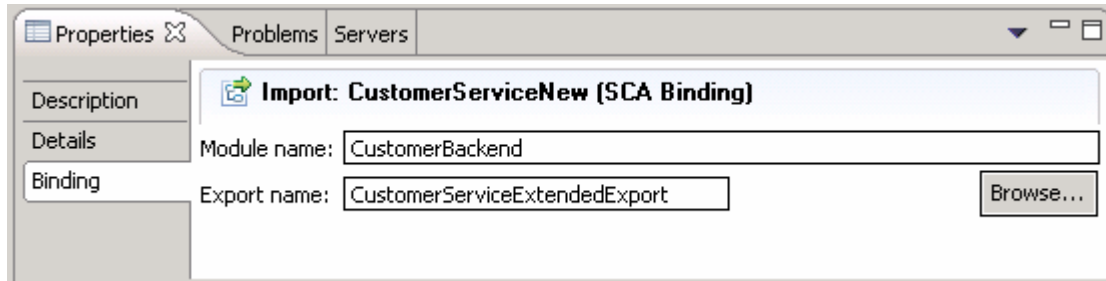


- 3) Update SCA binding for **CustomerServiceNew**.

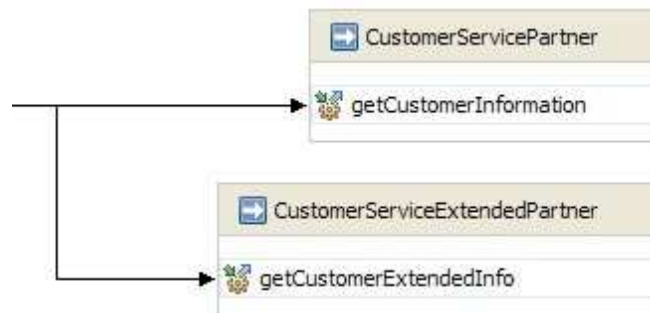
- a) Right-click on CustomerServiceNew and select **Generate Binding → SCA Binding** from the context menu.

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

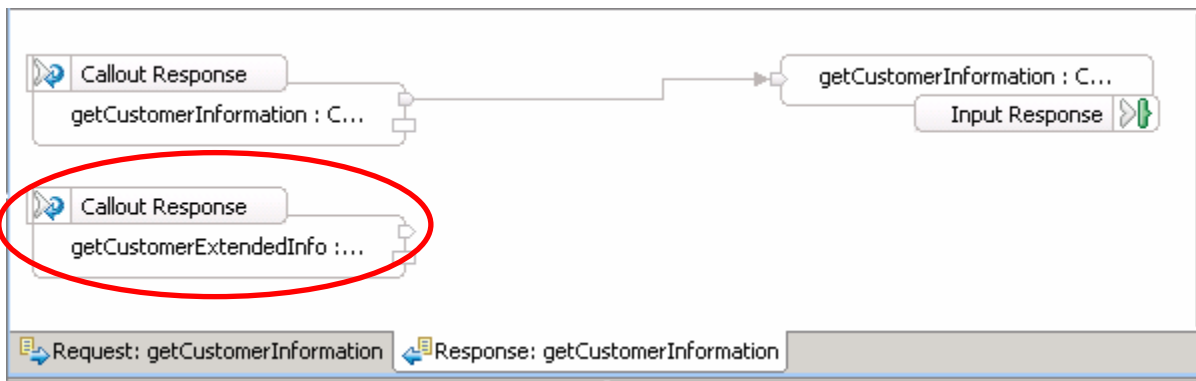
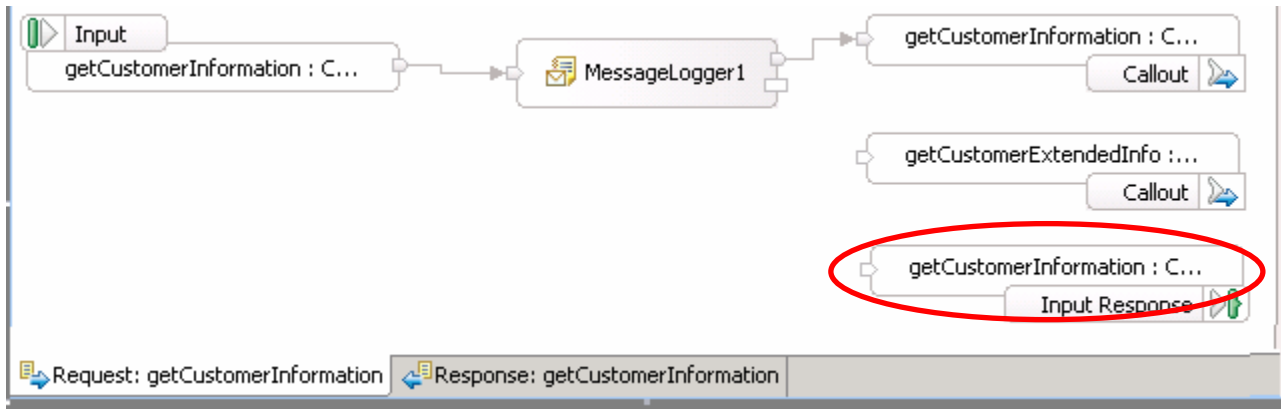
- c) Open the **Binding** tab of **Properties View**.
- d) Click **Browse...** button and select **CustomerServiceExtendedExport** from the SCA Export Selection panel. Click **OK**.
- e) Here is what your bindings tab should look like now.



- 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 context 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 must 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-hand 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.

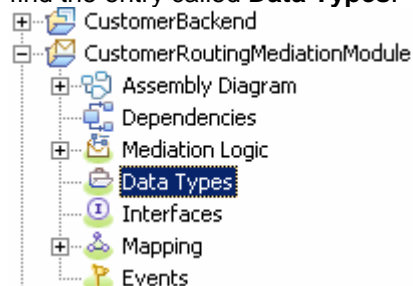


h) Save the Mediation Flow editor by selecting **File → Save** or **Ctrl + S**.

5) Create a transient context business object.

In this lab, the custom mediation primitive you will be creating 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.

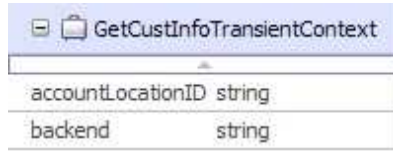
a) In the **Business Integration** view, expand **CustomerRoutingMediationModule** to find the entry called **Data Types**.



b) Right-click on **Data Types** and select **New → Business Object** from the context menu.

c) In the New Business Object panel, enter **GetCustInfoTransientContext** for the Name field. Click **Finish**.

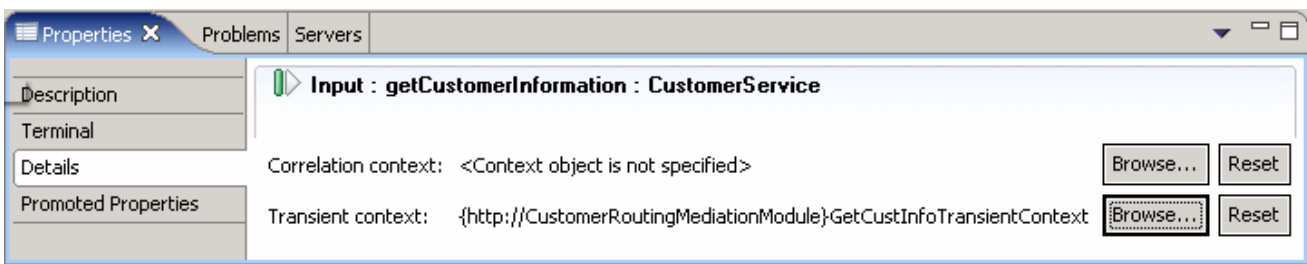
- d) The Business object editor opens. Click the add attribute icon (+) twice to add two attributes to the business object.
- e) Change the text attribute1 to **accountLocationID**.
- f) Change the text attribute2 to **backend**. The result should look like below picture.



- 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/Highlight 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 be adding 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.



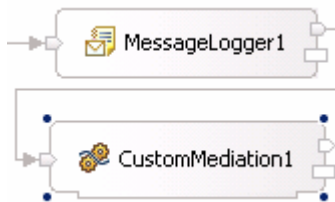
- ___ b. Press the Delete button on the keyboard or right click on wire and select **Delete**.

Drag and drop a custom mediation primitive on the Mediation Flow to create a new custom mediation primitive.

- ___ c. Click the custom mediation icon () in the mediation flow palette (left-hand side) and then click under the MessageLogger1. The CustomMediation1 will appear.

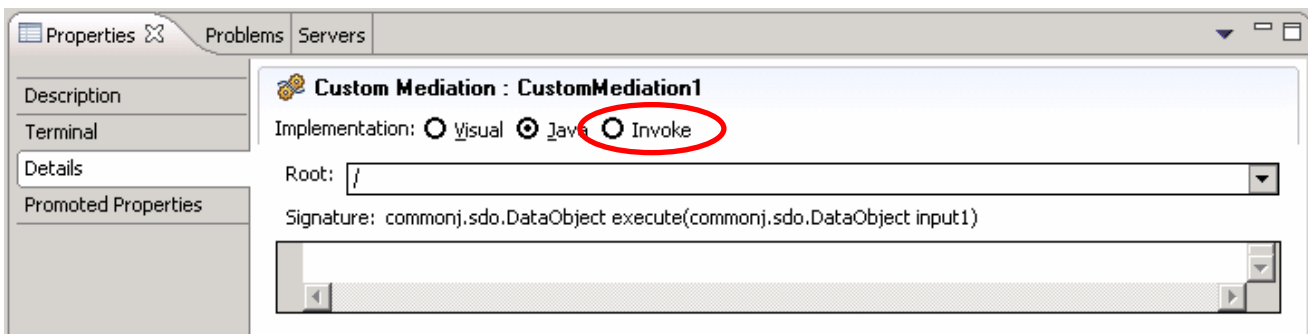
Wire the output terminal of MessageLogger1 to the input terminal of CustomMediation1.

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

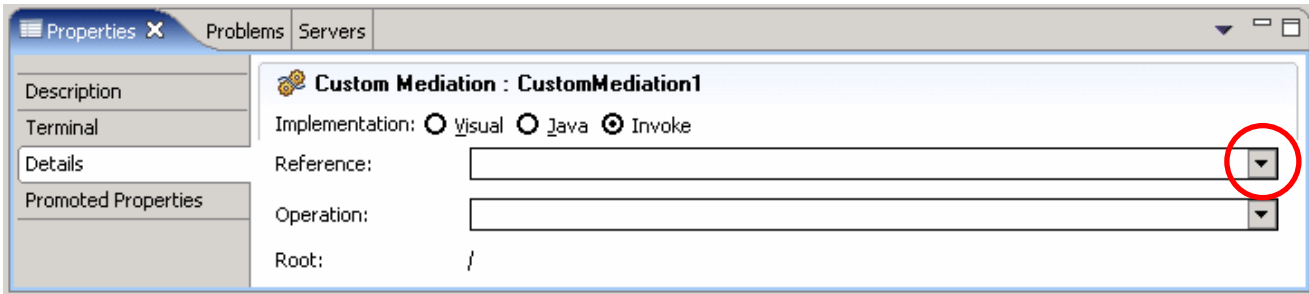


Define a Service Operation for CustomMediation1.

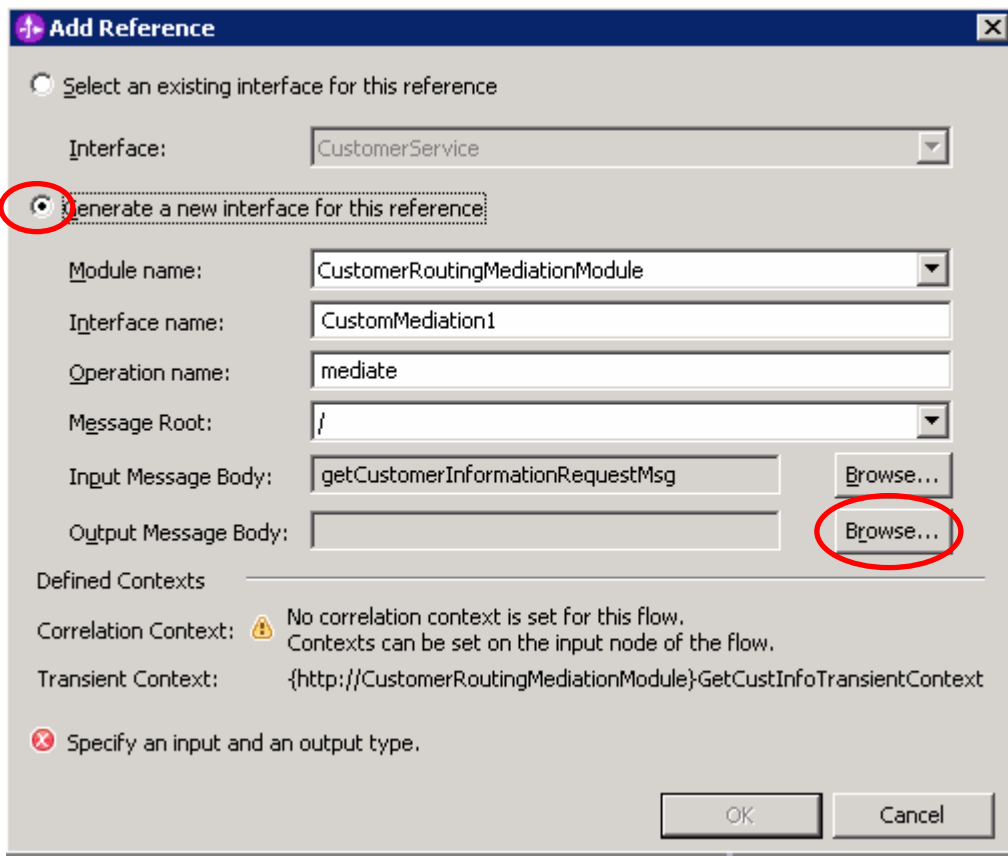
- ___ e. Click on CustomMediation1 and navigate to the **Details tab** of **Properties View**.



- ___ f. Select the radio button next to **Invoke**. Say **Yes** to any pop-up Question dialog.



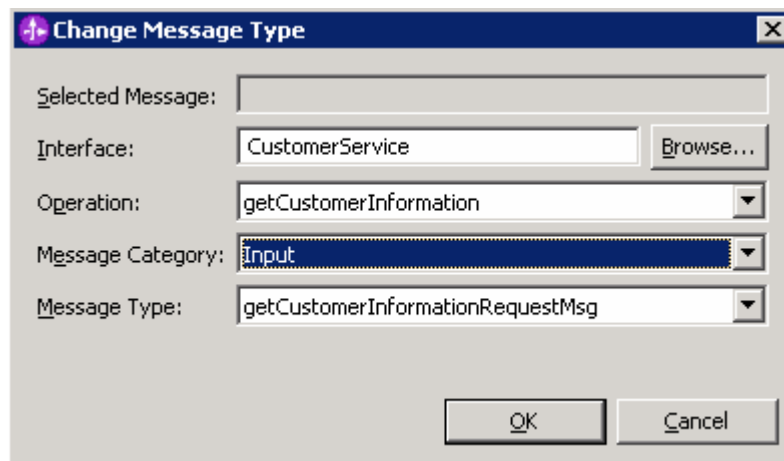
- ___ g. Click to drop-down list for the **Reference** field and select **Add a new reference** from the list. The **Add New Reference** panel pops-up.
- ___ h. In the **Add New Reference** panel, select the radio button next to **Generate a new interface for this reference**.



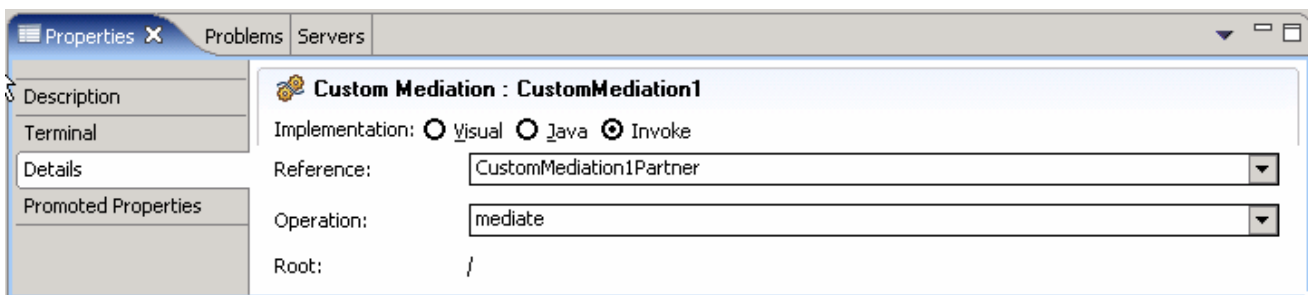
- ___ i. Ensure the following information is populated to the following fields:
 - 1) **Module Name** : CustomerRoutingMediationModule
 - 2) **Interface name** : CustomMediation1
 - 3) **Operation name** : mediate
 - 4) **Message Root** : /
 - 5) **Input Message Body** : getCustomerInformationRequestMsg

Note: Since you wired the MessageLogger1 output to the input of CustomMediation1, the input message body has been automatically defined. However, since you have not wired the output to any other object, the output message has not been defined.

- 6) Click the **Browse...** button next to **Output Message Body**. The Change Message Type panel opens.
 - a) On the Change Message Type panel, click the **Browse** button for **Interface**. Select **CustomerService** listed from the Interface Selection panel and click **OK**.
 - b) For **Operation** filed, click the drop down list and select **getCustomerInformation** operation.
 - c) For **Message Category** filed, use the drop down list and select **Input**.
 - d) The **Message Type** filed gets populated after choosing input or output (request or response) in this case **getCustomerInformationRequestMsg**. The Change Message Type panel must look like the picture shown below:




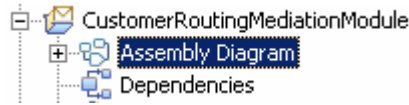
- e) Click **OK**.
- 7) The **Output Message Body** filed must be populated with the value, **getCustomerInformationRequestMsg**
- 8) Finally click **OK** over the Add Reference panel.
- 9) The **Details tab** will now look like this:



- ___ j. Save all the work by selecting **File → Save All** or **Ctrl + Shift + S**.

Synchronize Implementation from the Mediation Flow with Assembly Diagram.

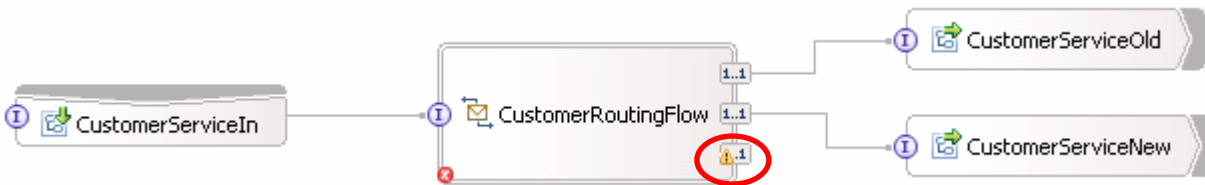
- ___ f. Expand **CustomerRoutingMediationModule** in the Business Integration view and double click on **Assembly Diagram** ( **Assembly Diagram**) to open it in the Assembly Diagram editor



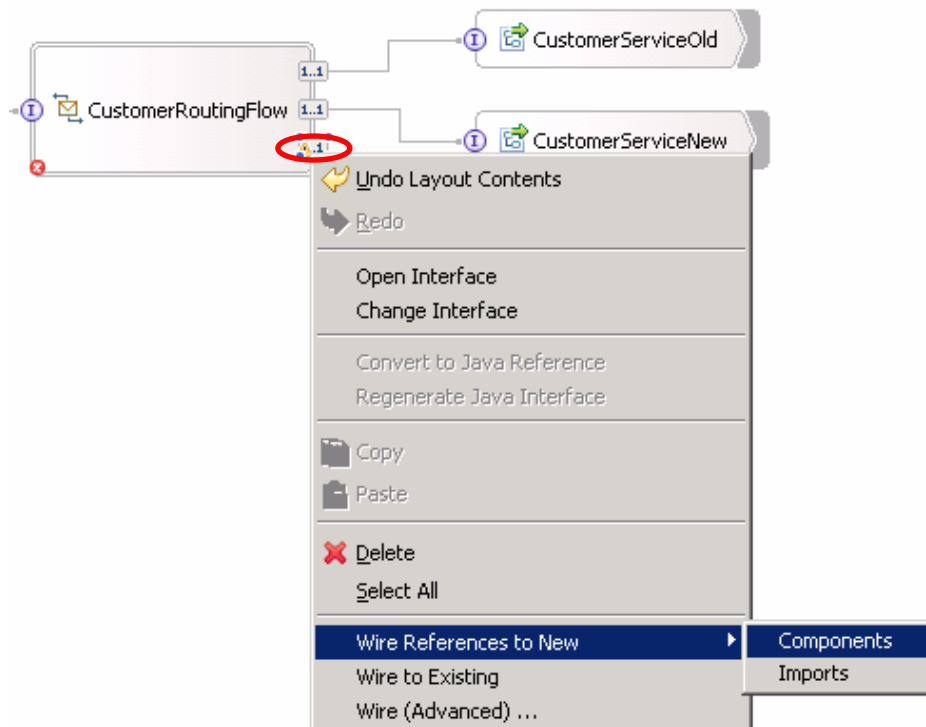
- ___ g. Right click on the **CustomerRoutingFlow** in the Assembly Diagram.
- ___ h. Select **Synchronize Interfaces and References → from Implementation** from the context menu. Wait for the workspace to auto build.

Note: Ignore any errors reflected in the problems views at this time.

- ___ i. Notice that a new Reference node is added to the **CustomerRoutingFlow** module as shown below:

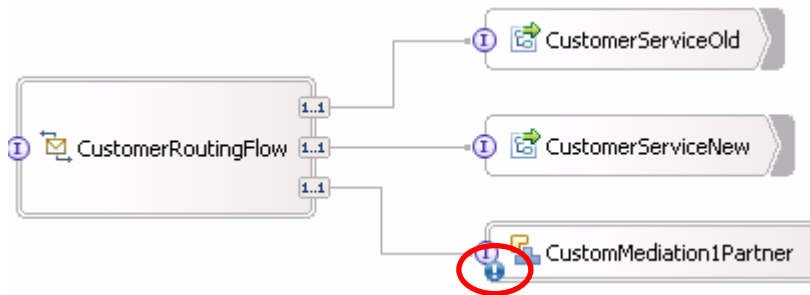


- ___ k. Right click over this new Reference node and select **Wire References to New → Components** from the context menu.



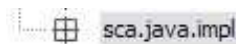
- ___ l. Notice that the **CustomMediation1Partner** Java component has been created on the Assembly Diagram. Save the update (**File → Save or key in Ctrl + S**) and there must not be any errors in the problems view after project has built.

Note: If the errors do not disappear from the problems view, clean and build all the projects and re-build all the projects again. (Project → Clean, Project → Build All)



Generate implementation code for the Java component and copy in snippet code.

- ___ m. The blue exclamation icon on **CustomMediation1Partner** states that it does not have any implementation code. Therefore, right-click on **CustomMediation1Partner** and select **Generate Implementation → Java**.
- ___ n. Click the **New Package...** button over the Generate Implementation panel. Type in **sca.java.impl** for the package, and select it from the generated list. Click **OK**.



- ___ o. CustomMediationPartner1_1Impl.java will be created and opened in a Java editor. Scroll down to find the public method... **public DataObject mediate(DataObject input1) {**
- ___ p. Add snippet to mediate method and organize imports for code.

- 1) Outside of WebSphere Integration Developer, navigate to **C:/Labfiles602/WESB/Lab2/snippet/snippet1.txt** and open in a text editor.
- 2) Copy contents and paste inside mediate method between the { } symbols where currently there is...

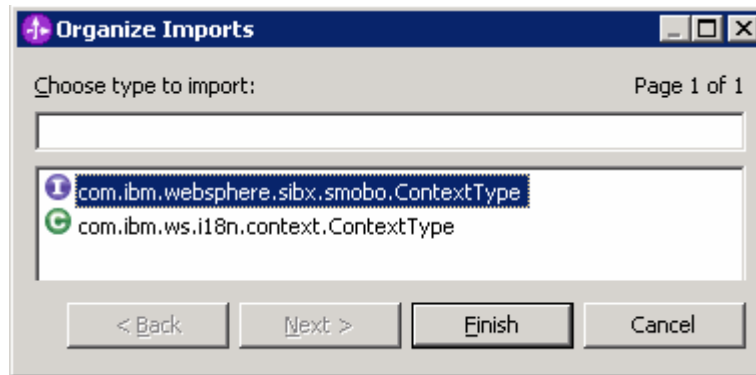
```
//TODO Needs to be implemented.
return null;
```

The code to replace this with is...

```
ServiceMessageObject smo = (ServiceMessageObject)input1;
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");  
return (DataObject)smo;
```

- 3) Notice there will be errors. You will need to organize imports in order for this code to work. Right-click in the Java editor and select **Source → Organize Imports** or **Ctrl + Shift + O**.
- 4) Select the first interface that displays **com.ibm.websphere.sibx.smobo.ContextType**.



- 5) Click **Finish**.
 - 6) There must be no errors at this time. Save work by navigating to **File → Save** or **Ctrl + S**.
 - 7) Close Java Editor.
- ___ q. Save Assembly Editor navigating to **File → Save** or **Ctrl + S**.
- ___ r. 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.

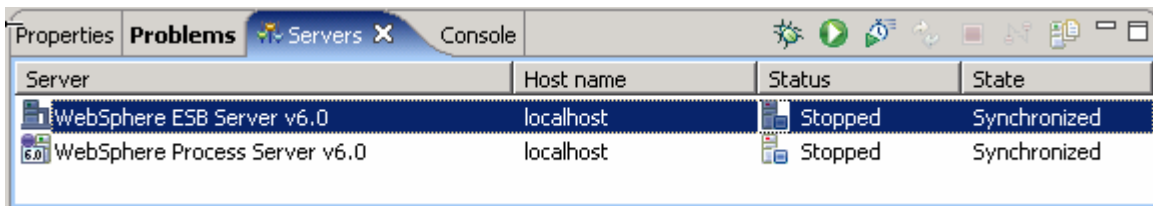
Start **WebSphere ESB Server v6.0**.

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 v6.0 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 v6.0 in the Servers view and select **Run administration console**.

___ e. Log-in to the administration console with any user ID you like.

___ f. Under the Task Filtering Selector section on the first page of the administration console, click on **Server and Bus**, then the **Apply** Button.

___ g. In the left navigation menu, find **Resources** and click to open.

___ h. Click on **JDBC Providers**.

___ i. There are 3 radio button choices when you first come to the JDBC Providers page. Click the radio button next to **Server: <SERVER_NAME>** and click the **Apply** button.

☐ Scope: Cell=**esbCell**, Node=**esbNode**, Server=**server1**

Cell : esbCell Scope specifies the level at which the resource definition is visible. For detailed information on what scope is and how it works, [see the scope settings help](#)

Node : esbNode [scope settings help](#)

Server : server1

- ___ j. A list of JDBC providers that the server knows of is listed. Click on **Cloudscape JDBC Provider (XA)** link.
- ___ k. In the following page, click on the **Data sources** link under **Additional Properties** on the right hand side of the page.
- ___ l. Click the **New** button at top-right of the data sources table to create a new data source.
- ___ m. Replace the default name **Cloudscape JDBC Driver XA Data Source** with **ESB Database Lookup Mediation Data Source** in the Name filed.
- ___ n. Replace the default JNDI Name **jdbc/Cloudscape JDBC Driver XA Data Source** with **jdbc/CustomRoutingMediation** in the JNDI Name field.

* Name

JNDI name

- ___ o. Scroll down to the bottom of the page and enter **C:\Labfiles602\WESB\CustomRoutingMediation** for **Database Name**.

Note: Ensure that the database directory **CustomRoutingMediation** is exiting under **C:\Labfiles602\WESB**. If not use the correct path of where the **CustomRoutingMediation** is.

If using a remote system for testing, you will need to put the database on your remote system by uploading it in binary. 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 **<Labfiles602>/CustomRoutingMediation**. Make sure the permissions are set up to allow the server to access the database.

Cloudscape data source properties

* Database name

- ___ p. Click the **Apply Button**.
- ___ q. After the page refreshes, click on the **save** link at the top of the page.

Messages

⚠ Changes have been made to your local configuration. Click [Save](#) to apply changes to the master configuration.

i The server may need to be restarted for these changes to take effect.

__ r. In the following page, click the **Save button**.

⊕ Total changed documents: 1

Save Discard Cancel

__ s. After the changes are saved, you will see the JDBC Providers table is updated with your new data source.

<input type="checkbox"/>	ESB Database Lookup Mediation Data Source	jdbc/CustomRoutingMediation
--------------------------	---	-----------------------------

__ t. 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 following message:

Messages


i Test connection for data source ESB Logger Mediation Data Source on server server1 at node esbNode was successful.

__ u. At the top-left of the Administration Console page, click on **Logout**. Close **Administrative Console**.

You have now built the data source and are ready to build the Database Lookup mediation primitive in your workspace.

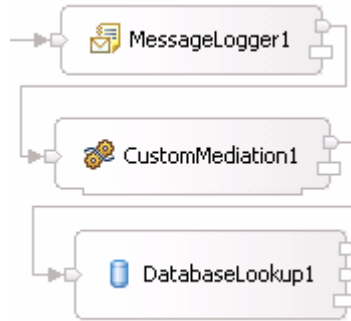
___ 2. Open the **CustomerRoutingFlow** Mediation Flow diagram from the Business Integration view by double-clicking the entry **CustomerRoutingMediationModule → Mediation Logic → Flows → CustomerRoutingFlow** if not already open.

___ 3. Drag and drop a database lookup primitive on to the Mediation Flow to create a new database lookup primitive.

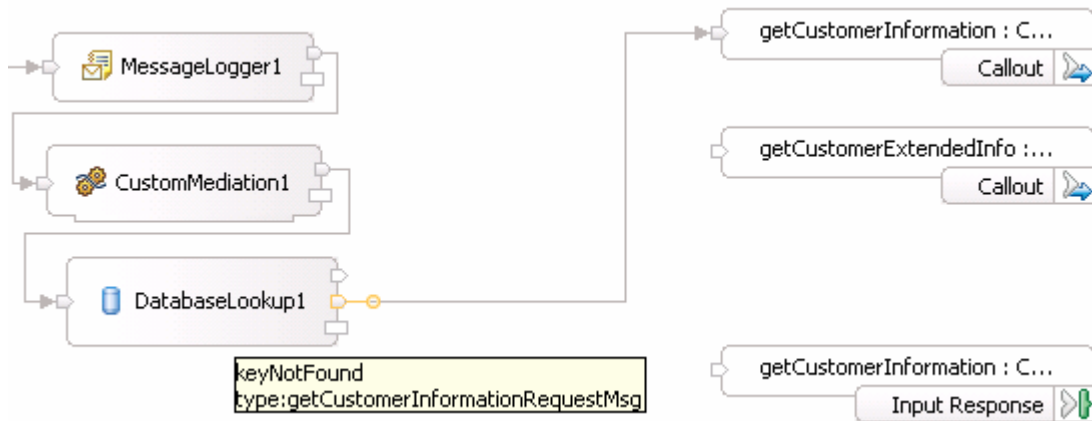
__ a. Click the database lookup icon () in the mediation flow palette (left-hand side) and then drop it under CustomMediation1. The DatabaseLookup1 primitive will appear.

Wire the output terminal of CustomMediation1 to the input terminal of DatabaseLookup1.

__ b. 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. As you would think, if a key is not found when a lookup is performed on the database, the message will be sent to this output terminal. If a key is not found, you would like to send that message to the **CustomerServiceOld** backend

- ___ e. In the above image, notice that the bottom node on the right-hand side, **getCustomerInformation: CustomerService InputResponse** node will not be 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.

Define the data source and database lookup information in the details tab of Properties view.

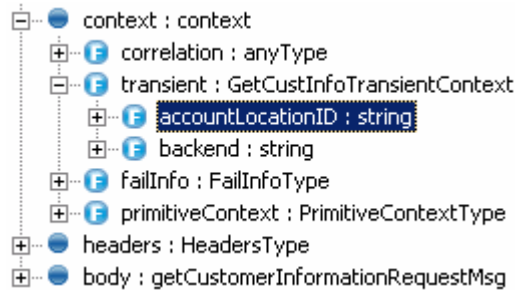
- ___ f. Click on DatabaseLookup1 and navigate to the **Details tab** of **Properties View**.
- ___ g. Enter the **Data source** information.
You have defined a database with a data source in the Administrative Console of the WebSphere Enterprise Service Bus server. In this sub-step, you are filling out the information needed to allow this application to use this data source and database.

- 1) Data source name: **jdbc/CustomerRoutingMediation**
- 2) Table Name: **BACKEND_LOCATIONS**

3) Key column name: **ACCT_NO_PREFIX**

4) Key path: Click on **Custom Xpath...** button to define which element to use.

a) Navigate the menu and click on **context → transient → accountLocationID** in the XPath Expression Builder panel



b) Click **OK**.

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

a) Click the **Add** button next to Data Elements table.

b) In the Add/Edit properties panel, enter the following values:

(1) Value column name: **BACKEND_ID**

(2) Message value type: **String** (default)

(3) Message element: Click on **Custom Xpath...** button and navigate to **context → transient → backend**.

(4) Click **OK**.

c) Click **Finish** over the Add/Edit properties panel.

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

Database Lookup : DatabaseLookup1

Data source name:*

Table name: *

Key column name: *

Key path: *

Validate input

Data elements:


Value column name	Message value type	Message element
BACKEND_ID	String	/context/transient

- ___ i. Save your work by navigating to **File → Save** or hitting **Ctrl + S** on your keyboard.
- ___ j. You are finished defining a Database lookup mediation primitive that looks up the backend identifier.

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

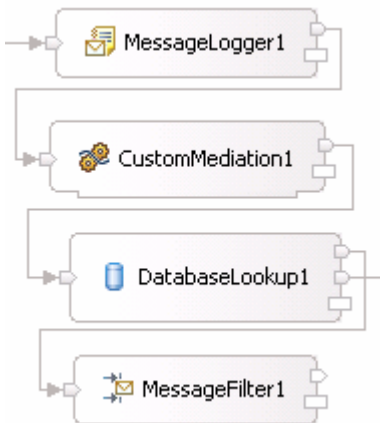
In this section you will determine the message routing based on backend identifier by adding a Message Filter primitive to the Mediation Flow. The old backend will go directly to the callout for CustomerService and the new backend will go to XSL Transformation mediation primitive (XSL Transformation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

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

___ a. Click the message filter icon () in the mediation flow palette (left-hand side) and then click under DatabaseLookup1. The MessageFilter1 primitive will appear.

Wire the output terminal of DatabaseLookup1 to the input terminal of MessageFilter1.

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



Add OldBackend1 and NewBackend1 output terminals to MessageFilter1

___ c. Click on MessageFilter1 and open the **Terminal tab** of the **Properties View**.

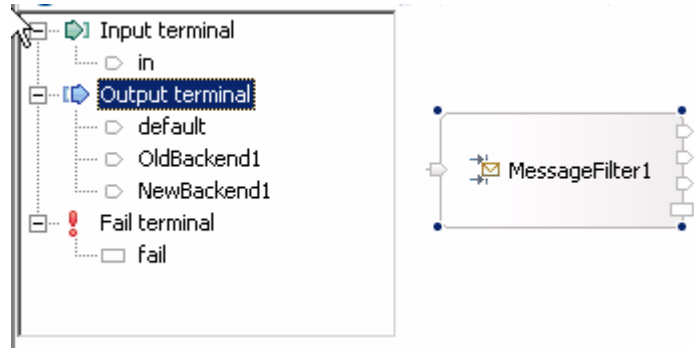
___ d. Right-click on **Output terminal** and select **Add Output Terminal** from the context menu.

___ e. Leave terminal type as "match", but change the **Terminal Name** to **OldBackend1**. Click **OK**.

___ f. Right-click on **Output terminal** and select **Add Output Terminal** from the context menu again.

___ g. Leave terminal type as "match", but change the **Terminal Name** to **NewBackend1**. Click **OK**.

___ h. You should now have 3 output terminals on MessageFilter1.



Add two filter options on Details tab to determine the message routing based on the backend identifier/pattern.

- ___ i. Click on the **Details tab** of the **Properties View** while the MessageFilter1 is selected. Here is where you will provide the filter options for the message filter to make decisions on which output to send the message to, NewBackend1 or OldBackend1.
- ___ j. Click on the **Add button** on next to the Filters table.
- ___ k. In the Add/Edit panel, click on the **Custom Xpath...** button.
- ___ l. Navigate to **context** → **transient** → **backend** and click on **backend**.
- ___ m. Below, next to **Condition:** click on the cell that currently has "Location" in it. Select the only **self:: () node : string** option.
- ___ n. For the value, click on "Value" in blue text, and then click on String. Enter the text **OLD1** and hit enter button on keyboard.

Schema Viewer:

- ServiceMessageObject : PARENT
- context : context
 - correlation : anyType
 - transient : GetCustInfoTransientContext
 - accountLocationID : string
 - backend : string
 - failInfo : FailInfoType
 - primitiveContext : PrimitiveContextType
- headers : HeadersType
- body : getCustomerInformationRequestMsg

XPath Location:	/context/transient/backend		
Condition:	self::node()	=	"OLD1"
Full XPath Expression:			
<input type="checkbox"/> Override			
/context/transient/backend[self::node()="OLD1"]			

- __ o. Click **OK**.
- __ p. Ensure the Terminal Name is **OldBackend1**.
- __ q. Click **Finish**.
- __ r. Now do this again to add a filter option for NEW1. Click on the **Add button** on next to the Filters table.
- __ s. In the Add/Edit panel, click on the **Custom Xpath** button.
- __ t. Navigate to **context → transient → backend** and click on **backend**.
- __ u. Below, next to **Condition**: click on the cell that currently has "Location" in it. Click the **self:: () node : string** option.
- __ v. For the value, click on "Value" in blue text, and then click on String. Now this time, enter the text **NEW1** and hit enter button on keyboard. Click **OK**.
- __ w. Click the drop-down box for Terminal Name and change it to **NewBackend1**.
- __ x. Click **Finish**. Here is what you should have in your filters table.

Filters:

Pattern	Terminal name	
/context/transient/backend[self::node()="OLD...	OldBackend1	
/context/transient/backend[self::node()="NE...	NewBackend1	

Add...

Edit...

Remove

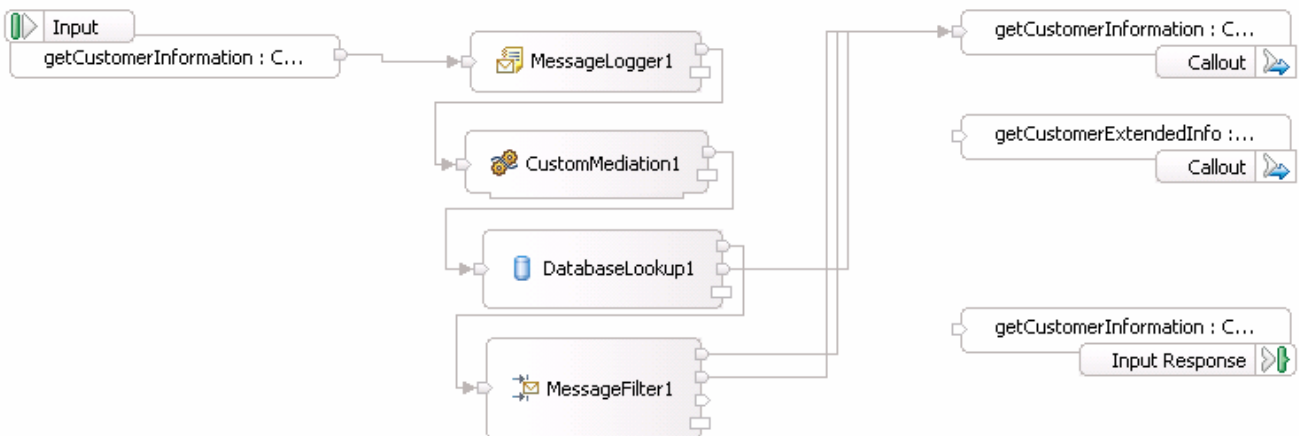
↑

↓

- __ y. Save your work by navigating to **File → Save** or hitting **Ctrl + S** on your keyboard.

Wire the default and OldBackend1 output terminals to CustomerServicePartner.

- __ z. In the mediation flow diagram, click on the **default output terminal** of MessageFilter1 and drag it to the **CustomerServicePartner Callout**.
- __ aa. Click on the **OldBackend1 output terminal** of MessageFilter1 and drag it to the **CustomerServicePartner Callout**.




___ bb. The NewBackend1 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.

Part 6: Add XSL transformation mediation primitives to mediation flow

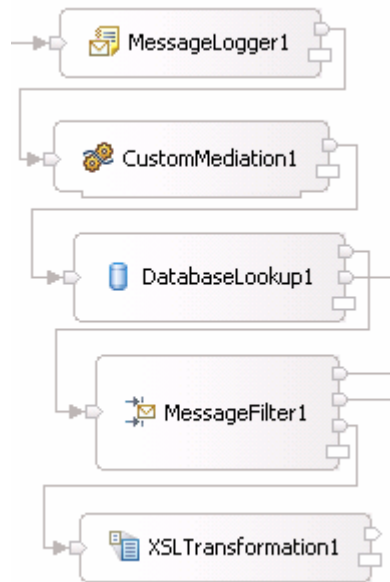
In this section you will create two XSL Transformations that will transform the CustomerService business object (BO) into the CustomerServiceExtended business object, then pass the newly formed message to the callout for CustomerServiceExtendedPartner instead of the CustomerServicePartner. Notice that 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 would 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.

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

___ a. Click the XSL Transformation icon () in the mediation flow palette (left-hand side) and then click under MessageFilter1. The XSLTransformation1 primitive will appear.

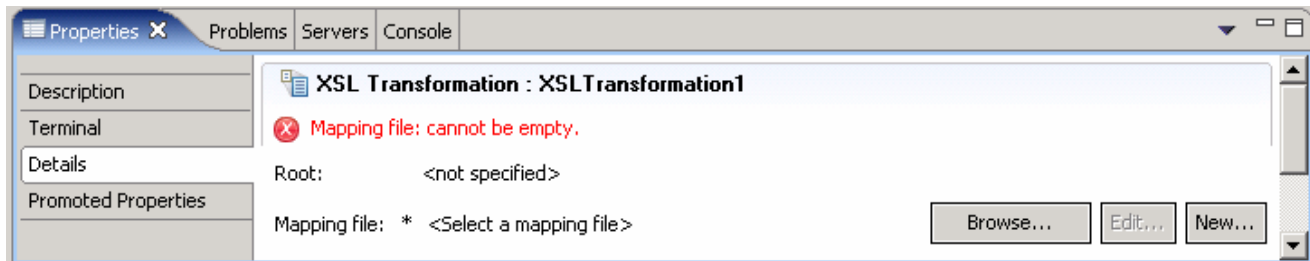
Wire the output terminal of MessageFilter1 to the input terminal of XSLTransformation1.

___ b. Click on **output terminal (NewBackend1)** of MessageFilter1 and drag a wire to the **input terminal** of XSLTransformation1.



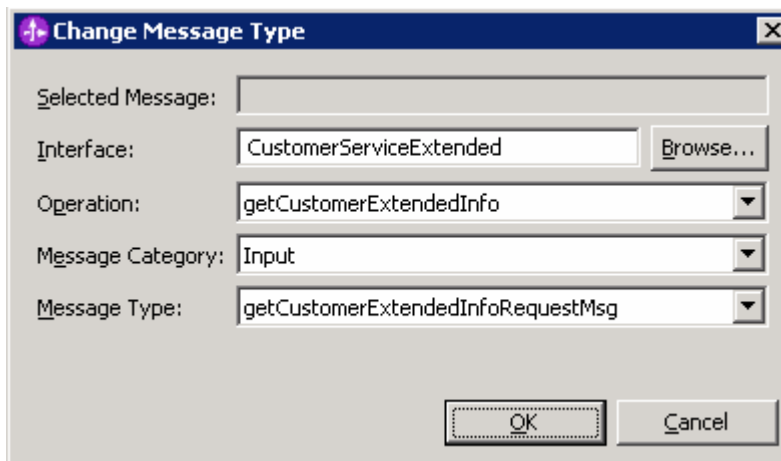
Create a new mapping for this XSL Transformation.

___ c. Select XSLTransformation1 and open the **Details tab** of the **Properties View**.



___ d. Click the **New...** button on the Mapping File row.

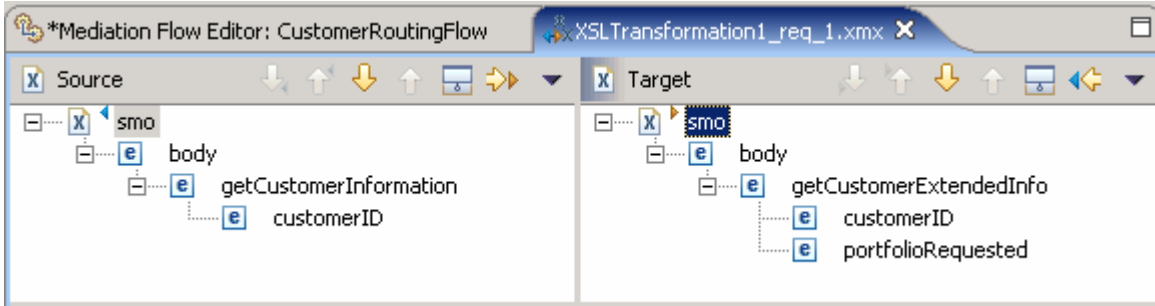
- 1) Ensure the **/body** is selected for Message Root filed.
- 2) Since you wired the MessageFilter1 to XSLTransformation1, the input message body is already assigned **getCustomerInformationRequestMsg**.
- 3) Click the **Browse...** button next to Output Message Body. The Change Message Type panel opens.
- 4) On the Change Message Type panel, click the **Browse...** button to open a list of interfaces. Select **CustomerServiceExtended** and click the **OK** button.
- 5) For operation, click the drop-down list to populate with the **getCustomerExtendedInfo** operation.
- 6) For Message Category, click the drop-down list and select **Input** since you are still working on the request side.
- 7) The Message Type then gets populated when choosing input or output (request or response). The Change Message Type panel must look like this:



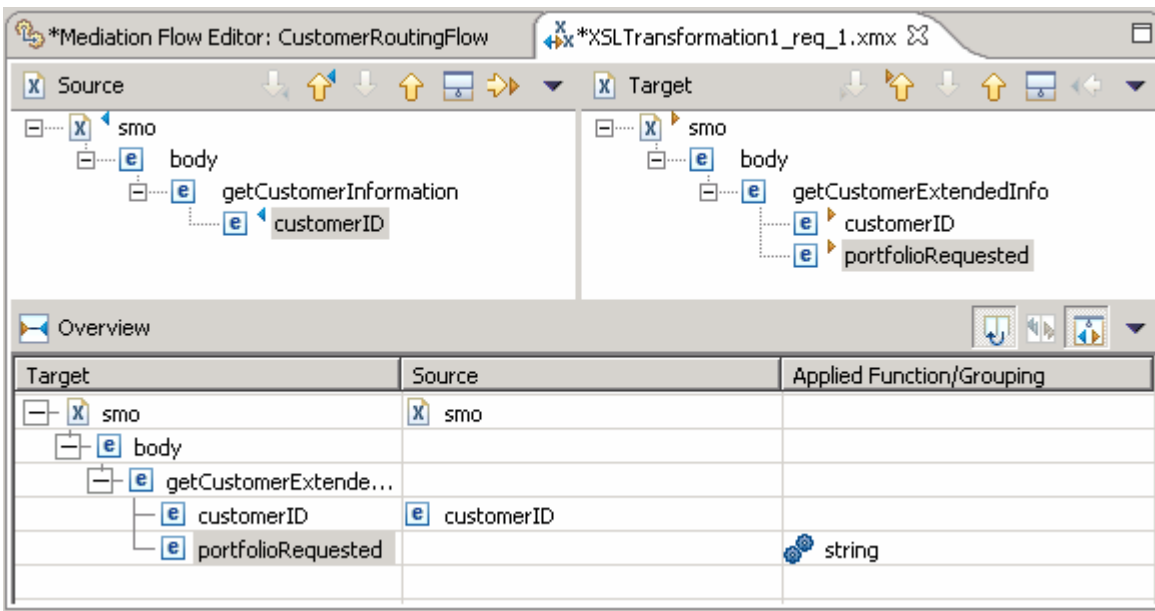
- 8) Click **OK**.

___ e. Now you should have **getCustomerInformationRequestMsg** for the Input Message Body and **getCustomerExtendedInfoRequestMsg** for the Output Message Body. Click **Finish**.

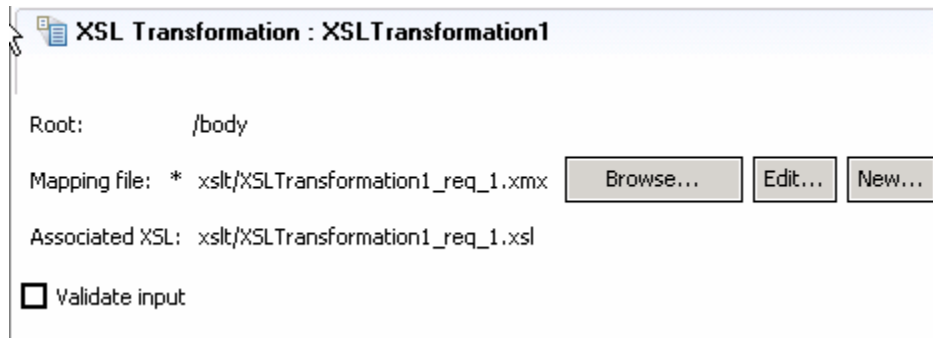
___ f. The mapping file XSLTransformation1_req_1.xmx will be created and opened. Completely expand both the **Source** and **Target objects** as shown below:



- ___ g. Click on **customerID** from the **Source** side and drag to the **customerID** on to the **Target** side. Notice triangle icons appear next to the mapped elements and that the Overview view shows the mapping.
- ___ h. 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 **Define XSLT Function**.
- ___ i. Leave **string** as default function. Click **Next**.
- ___ j. Change **Function Name** to **String** in the drop-down list.
- ___ k. Click the **Add button** for input parameters and type ‘default’. Click **OK**. Click **Finish**.



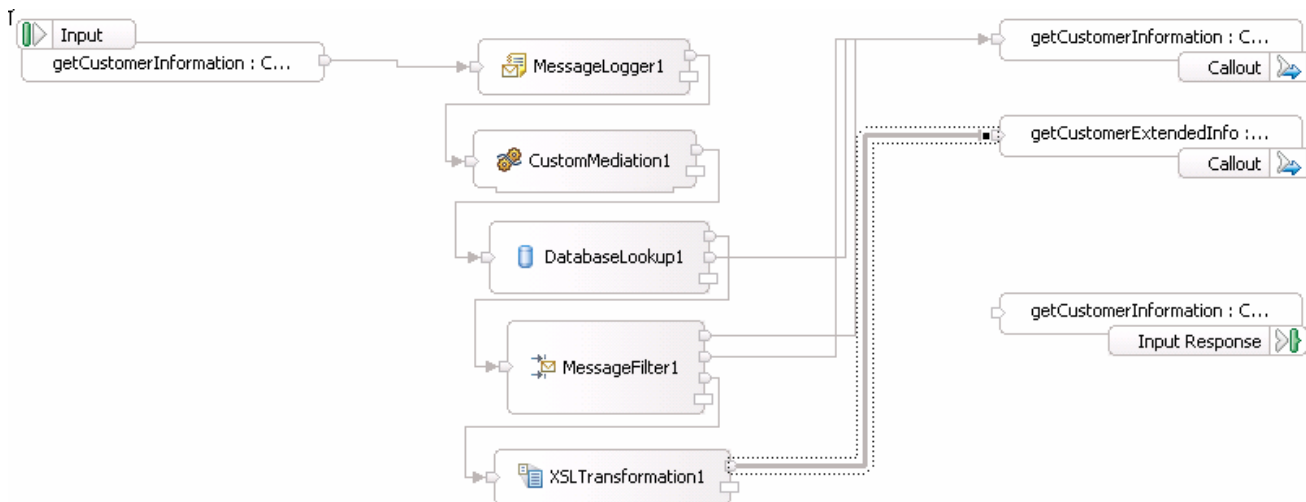
- ___ l. Save the mapping by navigating to **File → Save** or hit **Ctrl + S** on your keyboard. **Close** mapping file.
- ___ m. The details for the XSL Transformation, XSLTransformation1 must look like the picture shown below:



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

Wire the output terminal of XSLTransformation1 to the **getCustomerExtendedInfo: CustomerServiceExtendedPartner** Callout node

__ o. Click on output terminal of XSLTransformation1 and drag to the **getCustomerExtendedInfo: CustomerServiceExtendedPartner** Callout node.



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

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.

__ q. Click on the **Response tab** of mediation flow.

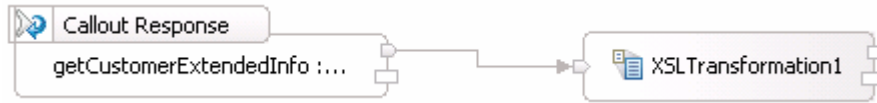


__ r. Drag and drop an XSL Transformation mediation primitive on the Mediation Flow to create a new XSL Transformation.

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

__ s. Wire the callout response for CustomerServiceExtendedPartner to the input terminal of XSLTransformation1.

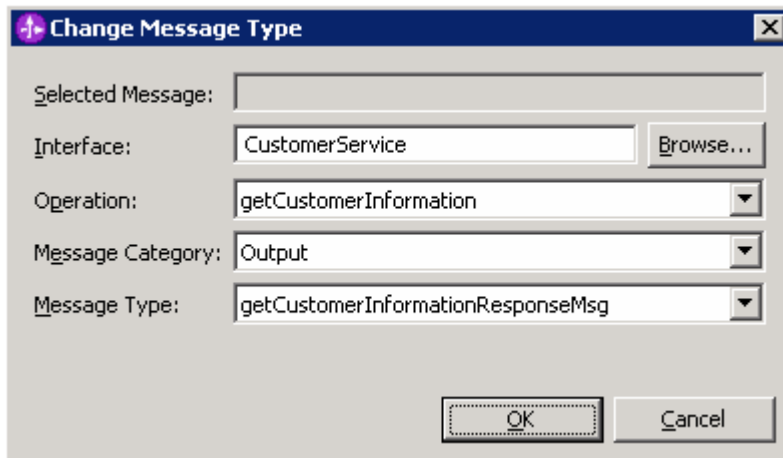
- 1) Click on **output terminal** of CustomerServiceExtendedPartner and drag a wire to the **input terminal** of XSLTransformation1.



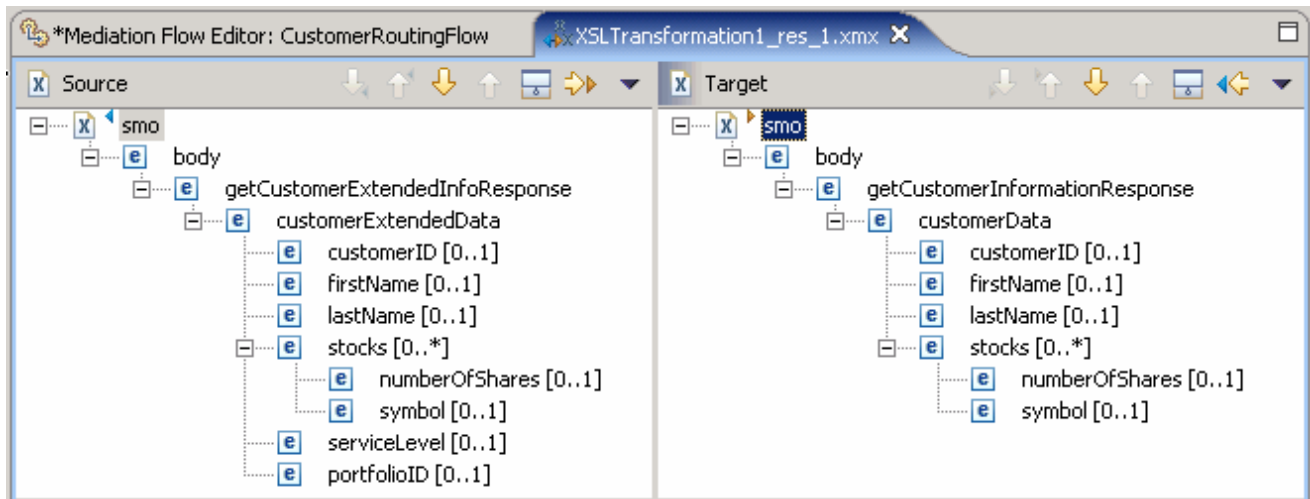
Create a new mapping for this XSL Transformation.

___ t. Select XSLTransformation1 and open the **Details tab** of the **Properties View**.

- 1) Click on the **New...** button on the Mapping File row.
- 2) Ensure that **/body** is selected for Message Root filed
- 3) Since you wired the Callout Response to XSLTransformation1, the input message body is already assigned as **getCustomerExtendedInfoResponseMsg**
- 4) Click the **Browse...** button next to Output Message Body. The Change Message Type panel opens.
- 5) On the Change Message Type panel, click the **Browse...** button to open a list of interfaces. Select **CustomerService** and click the **OK** button
- 6) For operation, click the drop-down list to populate with the **getCustomerExtendedInfo** operation.
- 7) For Message Category, click the drop-down and select **Output** since you are now working on the response side.
- 8) The Message Type then gets populated when choosing input or output (request or response). The Change Message Type panel must look like this:



- 9) Click **OK**.
- 10) Now you should have **getCustomerExtendedInfoResponseMsg** for the Input Message Body and **getCustomerInformationResponseMsg** for the Output Message Body. Click **Finish**.
- 11) The mapping file XSLTransformation1_res_1.xmx will be created and opened. Completely expand both the **Source** and **Target objects** as shown below:



- 12) Click on **customerID** from the **Source** side and drag to the **customerID** on the **Target** side. Notice triangle icons appear next to the mapped elements and that the Overview view shows the mapping.
- 13) Do this same dragging action (from source to target) for **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.

The screenshot shows the Mediation Flow Editor with two XML trees side-by-side. The left tree is the source, representing a 'getCustomerExtendedInfoResponse' with elements like 'customerExtendedData', 'customerID', 'firstName', 'lastName', 'stocks', 'numberOfShares', 'symbol', 'serviceLevel', and 'portfolioID'. The right tree is the target, representing a 'getCustomerInformationResponse' with elements like 'customerData', 'customerID', 'firstName', 'lastName', 'stocks', 'numberOfShares', and 'symbol'. Below the trees is an 'Overview' table showing the mapping between source and target elements.

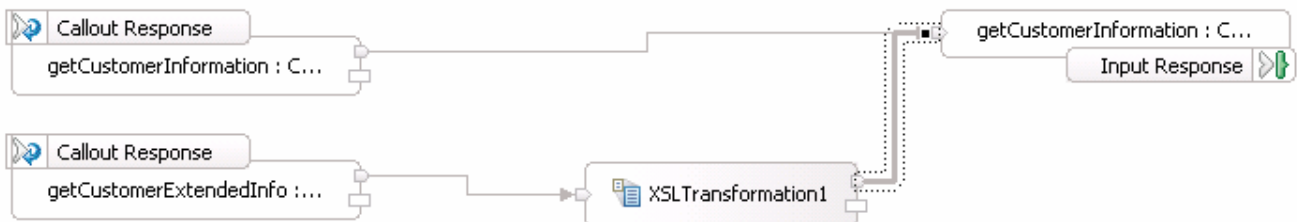
Target	Source	Applied Function/Grouping
smo	smo	
body		
getCustomerInformationR...		
customerData		
customerID [0..1]	customerID [0..1]	
firstName [0..1]	firstName [0..1]	
lastName [0..1]	lastName [0..1]	
stocks [0..*]	stocks [0..*]	
numberOfShare...	numberOfShares [0..1]	
symbol [0..1]	symbol [0..1]	

14) Save the mapping by navigating to **File → Save** or hit **Ctrl + S** on your keyboard. **Close** the mapping file.

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

__ u. Wire the output terminal of XSLTransformation1 to the CustomerService_getCustomerInformation_InputResponse.

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



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


Part 7: Test the CustomerServiceExtended backend

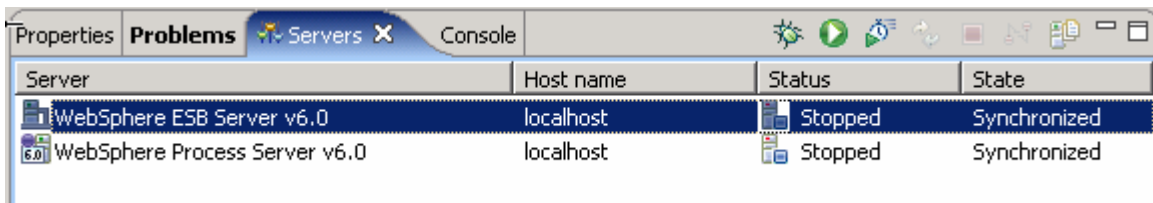
In this section you will run a JSP that enters a customerID to start off the process. However, for lab 2 you are going to test the new CustomerServiceExtended backend instead of the old CustomerService backend.

- ___ 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 v6.0 and click **Start button** ().

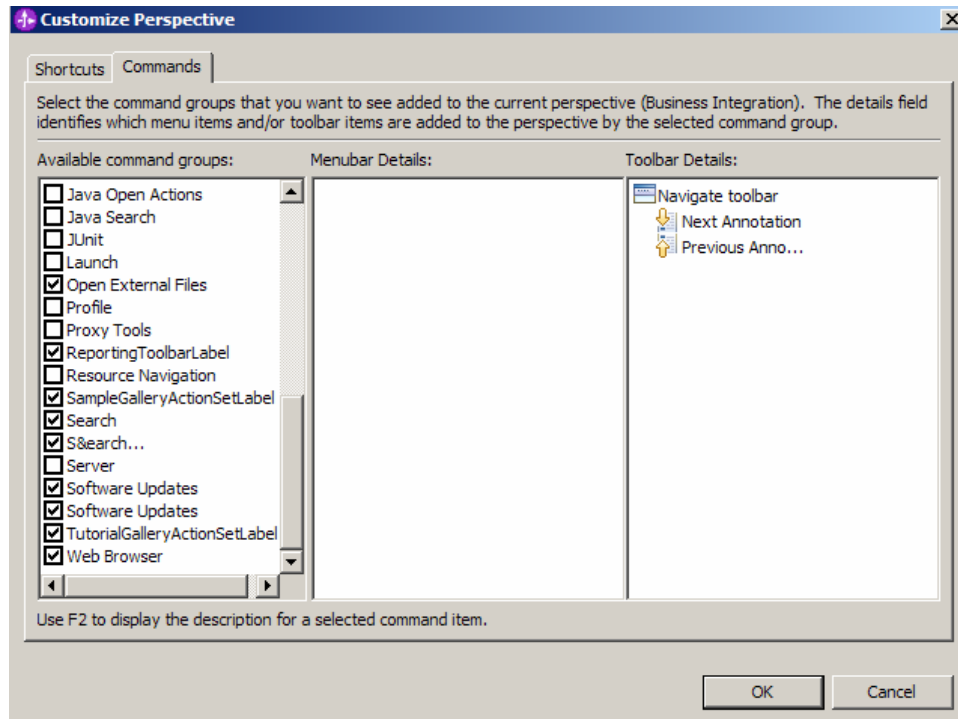


Add **projects** to WebSphere Enterprise Service Bus Server (once the ESB Server is started, not before).

- ___ a. In Servers view, right-click on WebSphere ESB Server v6.0 and select "**Add and Remove Projects...**"
- ___ b. Click **Add-All** button to move all projects to server and click **Finish** button. Wait for the deployment to finish.

Enable Web browser icon in WebSphere Integration Developer. You only need to do this once.

- ___ c. Go to **Window → Customize Perspective**
- ___ d. Click on the commands tab and scroll to the bottom
- ___ e. Click the check box next to Web browser



__ f. Click OK. This should put an icon for Web browser in the WebSphere Integration Developer tools panel.



__ g. Click on the Web browser icon to launch a browser in WebSphere Integration Developer

__ h. Enter <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. For z/OS, to find the port look at **Application servers > <SERVER_NAME> > HTTP transport** in the admin console.

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: `C:\WID602\pf\esb\config\cells\esbCell\nodes\`

__ c. 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

- ___ d. Once you have the value of the customerID in the JSP, **Close** the browser.
- ___ e. Stop WebSphere ESB Server v6.0 by highlighting the WebSphere ESB Server v6.0 in the Servers View and click **Stop** button ().
(FYI... You cannot view data in a Cloudscape database when server is running.)

You are done with this exercise. Go ahead and clean up the workspace (steps below) for future work.

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:/LabFiles602/WESB/Lab2/
- ___ e. Name the project interchange **WPIv602_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 v6.0** from the Servers view of the Business Integration perspective.
2. Right-click on WebSphere ESB Server v6.0 (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 v6.0.

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 V6.0.2. 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 (BO) 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 **C:/LabFiles602/WESB/Lab2/WPIv602_ESB_Lab2_solution_Pi.zip**

___ c. Click **Finish** button.

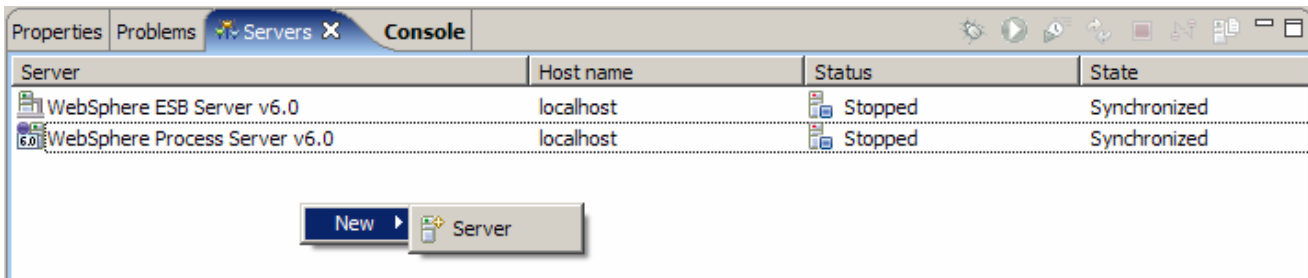
Add the data source from the first section of **Part 4**, the Database Lookup.

Continue past to **Part 7** to test the application.

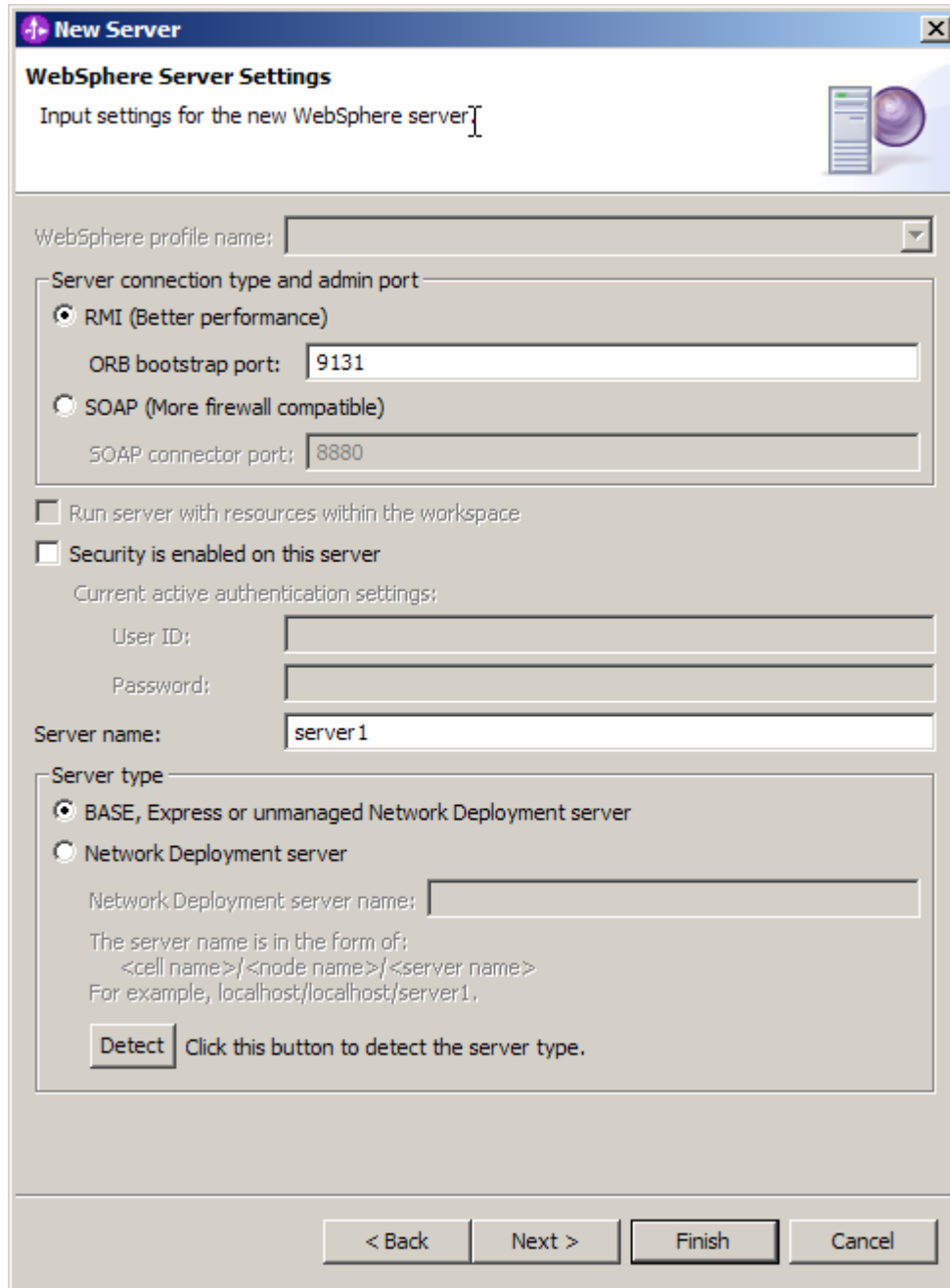
Task: Adding remote server to WebSphere Integration Developer test environment

This task describes how to add a remote server to the WebSphere Integration Developer Test environment. The sample will use a z/OS machine.

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



- ___ c. Specify hostname to the remote server, **<HOSTNAME>**.
- ___ d. Ensure that **'WebSphere Process v6.0 Server'** is highlighted in the server type list.
- ___ e. Click **Next**.
- ___ f. On the WebSphere Server Settings page, select the radio button for **RMI** and change the ORB bootstrap port to the correct setting (**<BOOTSTRAP_PORT>**).



___ g. Click **Finish**.

___ h. The new server should be seen in the Server view.

Start the remote server if it is not already started. WebSphere Integration Developer does not support starting remote servers from the Server View.

___ i. From a command prompt, telnet to the remote system if needed:

'telnet <HOSTNAME> <TELNET_PORT>'

userid: **<USERID>**

pw: **<PASSWORD>**

If the remote server runs on a System i[®] server, sign on to the system and start a Qshell session.

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

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

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

__ l. 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
```

This page is left intentionally blank.