

WebSphere Enterprise Service Bus lab

Event emitter primitive

What this exercise is about	2
Lab requirements	2
What you should be able to do	2
Introduction	3
Exercise instructions	4
Part 1: Prepare environment for the lab.....	6
Part 2: Create mediation module and flow.....	8
Part 3: Deploy the modules to the test server.....	23
Part 4: Event generation test A and B using the Web services explorer	26
Part 5: Event generation test C using the test client.....	35
Part 6: Save the work and clean up server	41
What you did in this exercise	42
Solution instructions	43
Task: Adding remote server to WebSphere Integration Developer test environment	44

What this exercise is about

The objective of this lab is to demonstrate the ability to produce different events based on the event emitter primitive configuration. It focuses particularly on the event label and root properties which define the different event types at runtime.

Lab requirements

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

- WebSphere Integration Developer V6.1 with the WebSphere ESB test server option installed

What you should be able to do

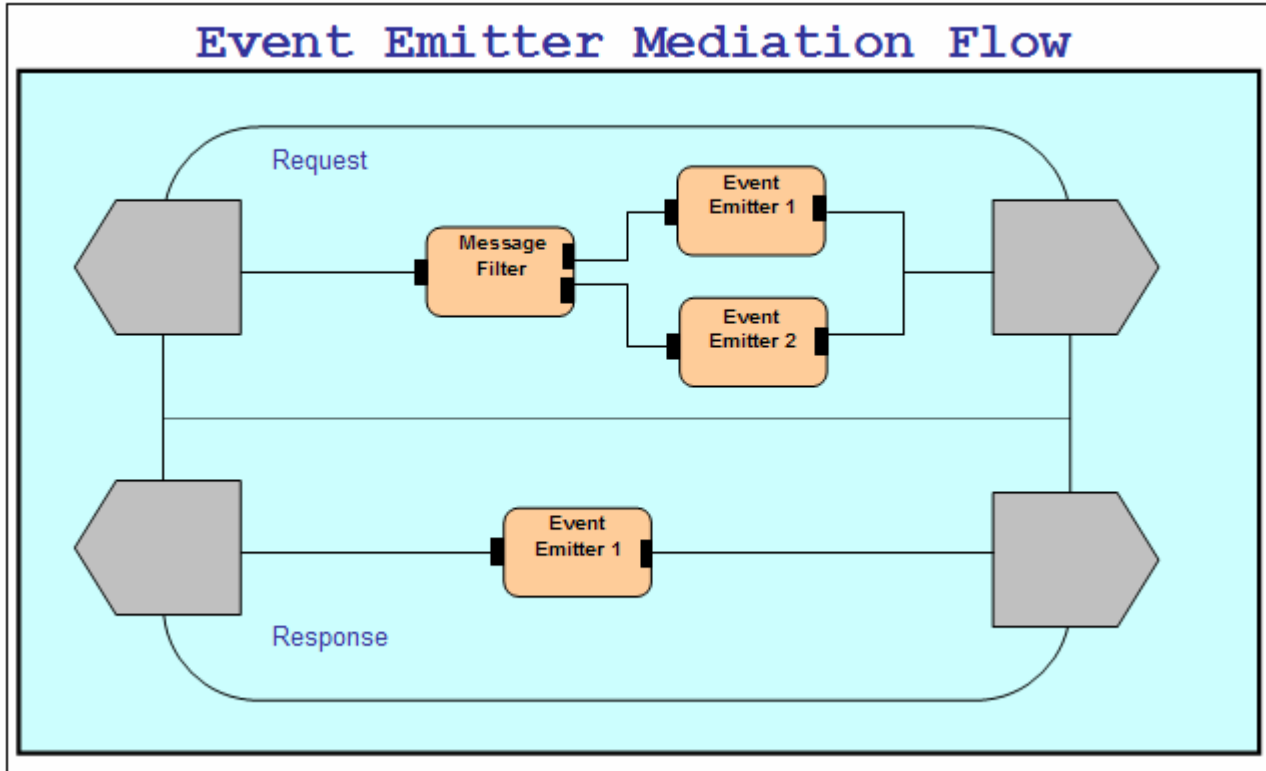
At the end of this lab you should be able to:

- Import the project interchange file into the WebSphere Integration Developer V6.1 development environment
- Create a Mediation Module and a Mediation Flow
- Visually compose the Mediation Flow that was created
- Run the Mediation Module on the WebSphere Process Server 6.1 test server
- Test the Event Emitter primitive to produce events using the Web Service Explorer
- Also do the same test using a Test Client

Introduction

The scenario in this lab demonstrates the ability to produce different events based on the event emitter primitive configuration. It focuses particularly on the event label and the root properties which define the different event types at runtime.

The Mediation flow represented in the diagram below; has the request flow with two Event Emitter primitives and the response flow with one Event Emitter primitive.



The following are the permutations which ensure that the tools generate appropriate default event labels for request and response flows. At runtime the mediation flow should emit three different types of events testing the runtimes ability to specify the event's extension name and content. The message filter primitive filters on the age element in the CustomerDetails business object. If the value of 'age' is equal to or greater than 16, event emitter 1 is run in the request flow; otherwise event emitter 2 is run.

Event Emitter 1 (Request Flow)	
Event Label	<default generated by the tools>
Root	<unspecified>
Transaction Mode	Default
Event Emitter 2 (Request Flow)	
Event Label	"JuniorAccount"
Root	/body
Transaction Mode	Default
Event Emitter 1 (Response Flow)	
Event Label	<default generated by the tools>
Root	"/body/getCustomerID/input1/customerID"
Transaction Mode	Default

Exercise instructions

Some instructions in this lab might be Windows operating system specific. If you plan on running WebSphere Integration Developer on a Linux operating system you will need to issue 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
<ESB_PROFILE_HOME>	<WID_HOME>\pf\esb	<WID_HOME>/pf/esb
<LAB_FILES>	C:\Labfiles61\WESB\EventEmitterPrimitive	/tmp/Labfiles61/WESB/EventEmitterPrimitive
<WORKSPACE>	C:\Labfiles61\WESB\EventEmitterPrimitive\workspace	/tmp/Labfiles61/WESB/EventEmitterPrimitive/workspace

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. This table is related to where you are running remote test environment:

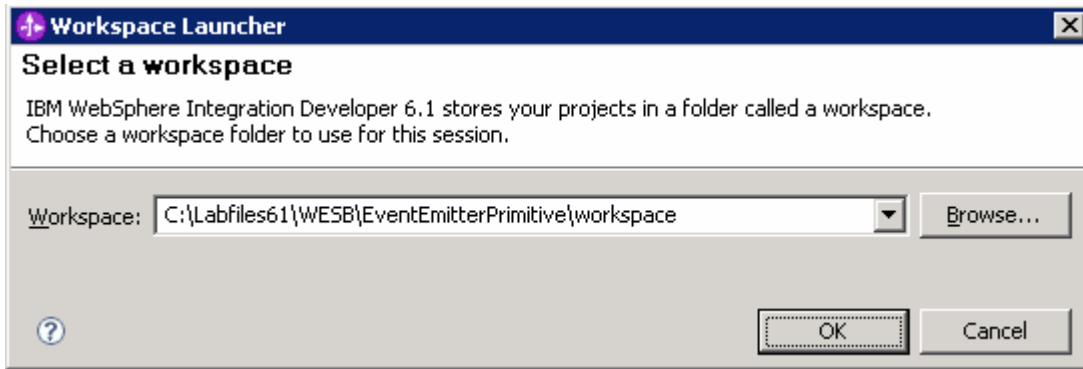
Reference variable	Example: Remote Windows test server location	Example: Remote z/OS® test server location	Input your values for the remote location of the test server
<SERVER_NAME>	server1	sssr011	
<WAS_HOME>	C:\Program Files\IBM\WebSphere\AppServer	/etc/sscell/AppServer	
<HOSTNAME>	localhost	mvsxxx.rtp.raleigh.ibm.com	
<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](#).

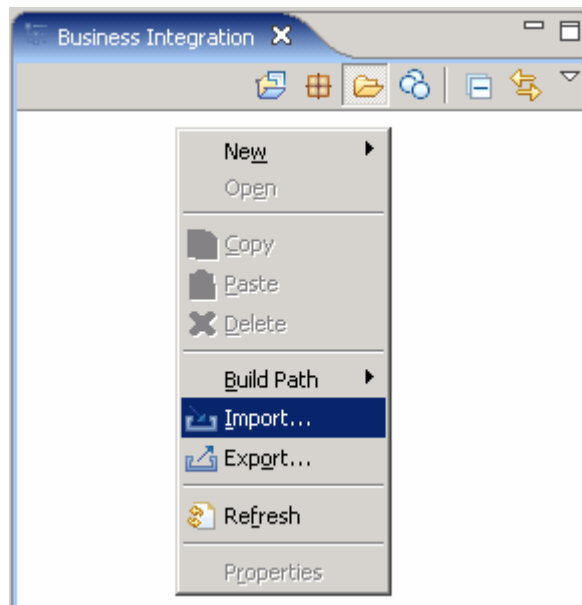
Part 1: Prepare environment for the lab

In this section of the lab, all the projects that are part of **WESB_EventEmitterPrimitive_Pi.zip** project interchange file are imported into a new workspace.

- ___ 1. Start WebSphere Integration Developer V6.1 with a workspace location of **C:\LabFiles61\WESB\EventEmitterPrimitive\workspace**



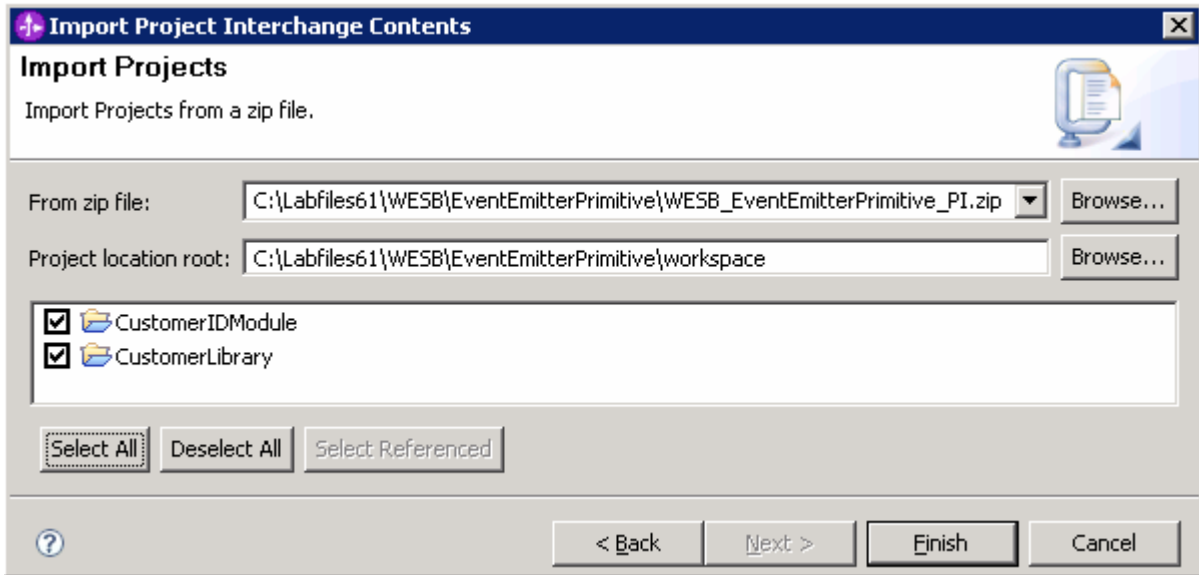
- ___ 2. On the welcome screen, click the curved arrow at the top right to '**Go to the business integration perspective**' (), to close the Welcome window
- ___ 3. Import the Project Interchange file, **WESB_EventEmitterPrimitive_Pi.zip**, into the development environment
 - ___ a. Right-click inside **Business Integration View** (top left view in the Business Integration Perspective)
 - ___ b. Select **Import** from the pop-up menu



- ___ c. From the **Import** dialog, expand '**Other**' and select **Project Interchange** from the list.

___ d. Click **Next**

___ e. Click the **Browse** button for '**From zip file**' to navigate for the Project interchange file, **WESB_EventEmitterPrimitive_Pi.zip**



___ f. Click the **Select All** button to ensure all projects listed are selected

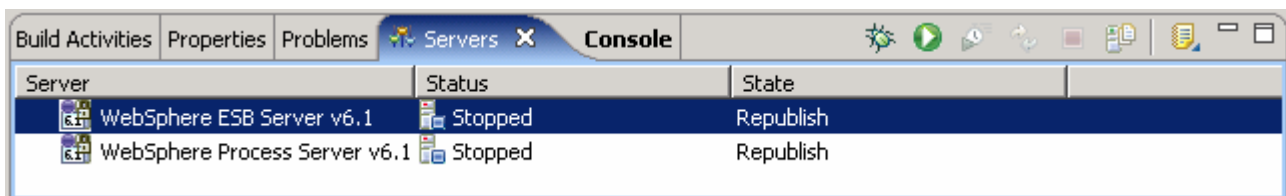
- Projects listed are, **CustomerIDModule** and **CustomerLibrary**

___ g. Click the **Finish** button (this imports the projects and auto-build is run)

___ h. Verify that the **CustomerIDModule** and **CustomerLibrary** modules are listed in the Business Integration view



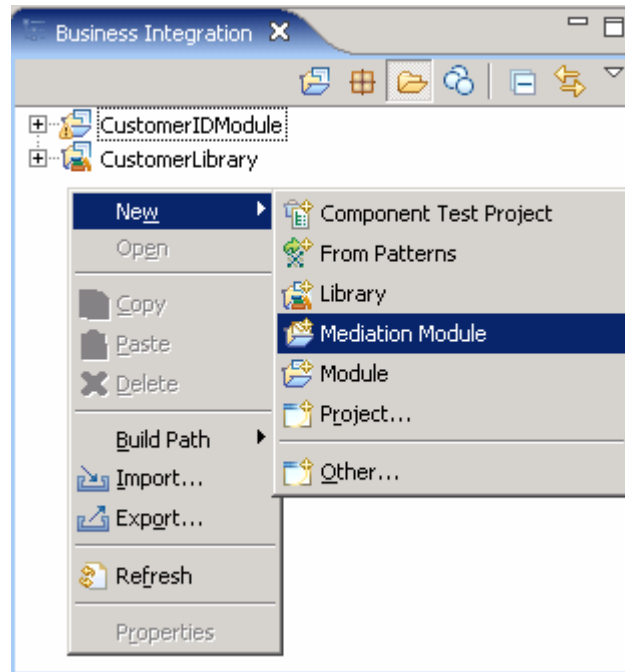
___ 4. Verify that the '**WebSphere ESB Server v6.1**' is listed in the **Servers** view



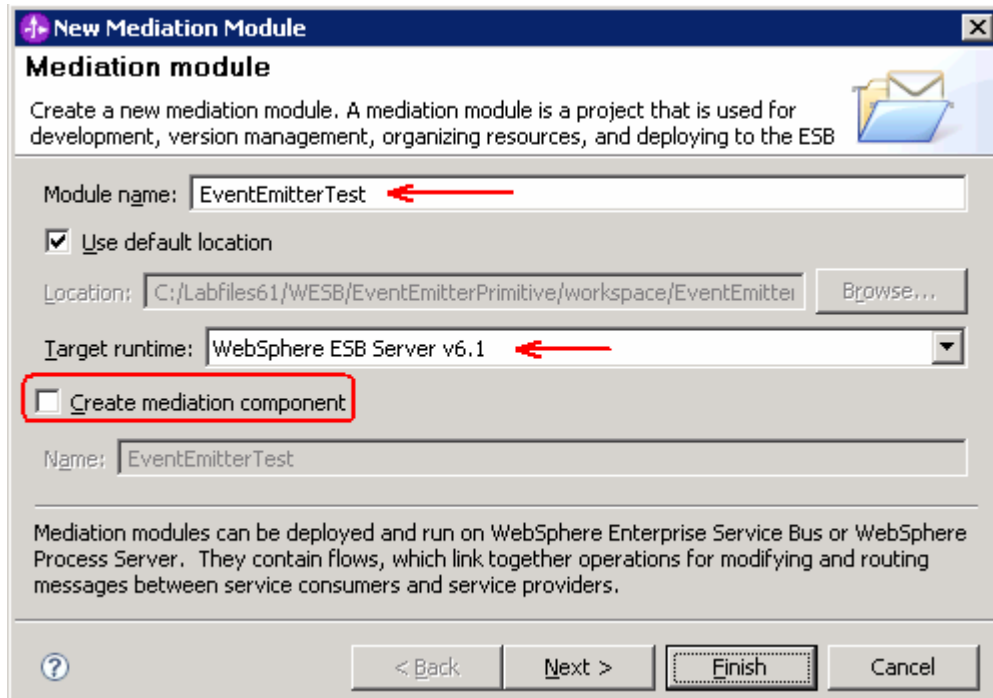
Part 2: Create mediation module and flow

In this section of the lab, a new mediation module and a Mediation Flow are created. The request and response flows are created for the Mediation flow. Further the Mediation Module is visually composed using the assembly editor.

- ___ 1. To create the mediation module, complete these steps:
 - ___ a. In the Business Integration view, right-click to see the pop-up menu and select **New → Mediation Module**. The new Mediation Module window opens



- ___ b. In the 'New Mediation Module' panel, type the **Module Name** as **EventEmitterTest**
- ___ c. Verify that the target runtime is the **WebSphere ESB Server v6.1** and **clear** the '**Create mediation flow component**' check box



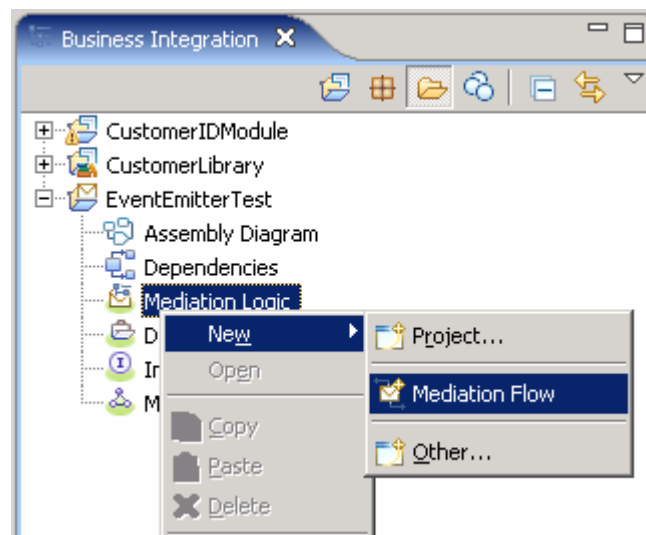
___ d. Click **Next**

___ e. In the Select Required Libraries panel, select **CustomerLibrary**

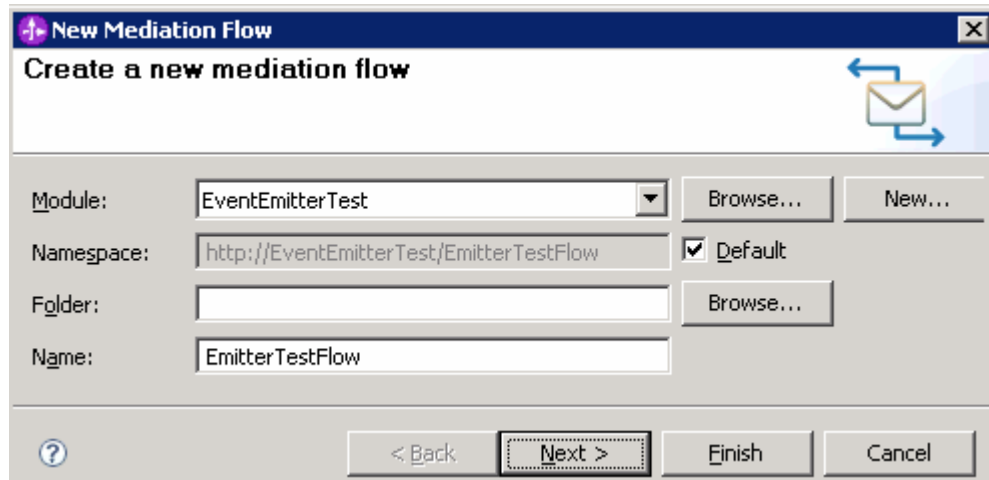
___ f. Click **Finish**. A mediation module named **EventEmitterTest** is created

___ 2. Create a Mediation Flow by completing these steps:

___ a. In the Business Integration view, expand '**EventEmitterTest**' Mediation Module, right-click on '**Mediation Logic**' and select **New** → **Mediation Flow** from the pop-up menu



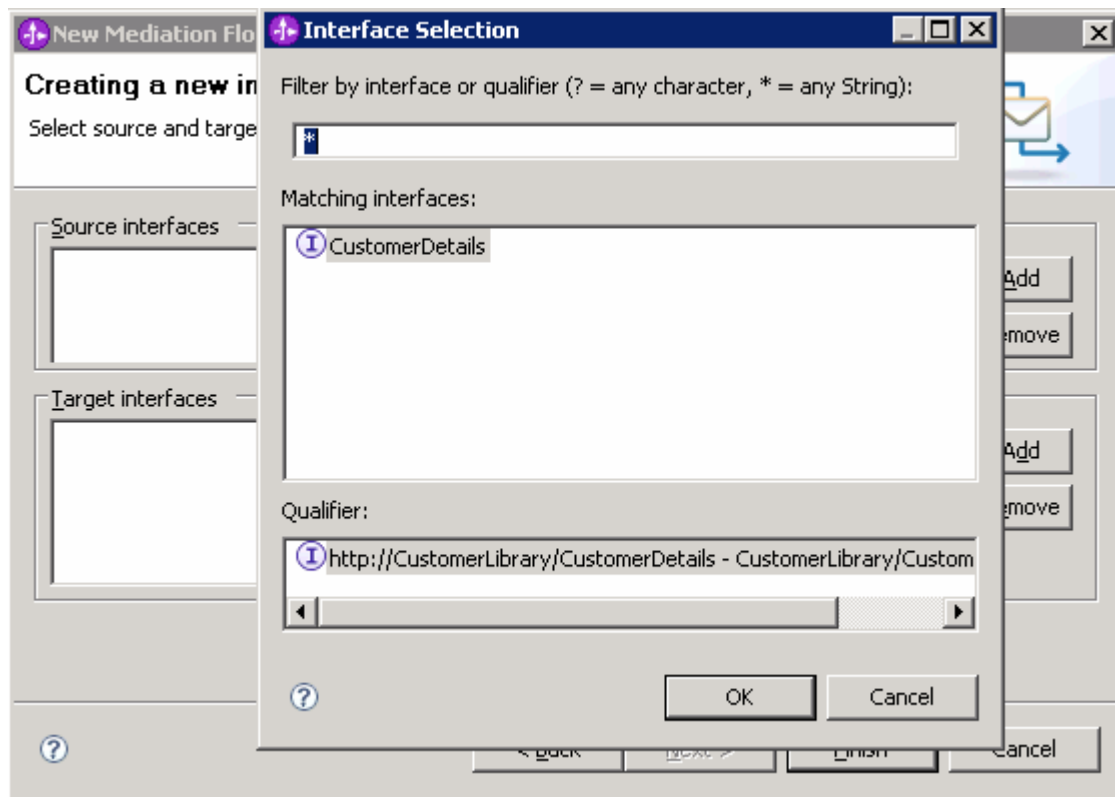
___ b. In the New Mediation Flow panel, enter the **Name** as **EmitterTestFlow**



__ c. Click **Next**

__ d. In the 'Creating a new interface' panel, add the source and target interfaces

- 1) Click the **Add** button next to the **Source interfaces** text area and select **CustomerDetails** from the **Interface Selection** pop-up window

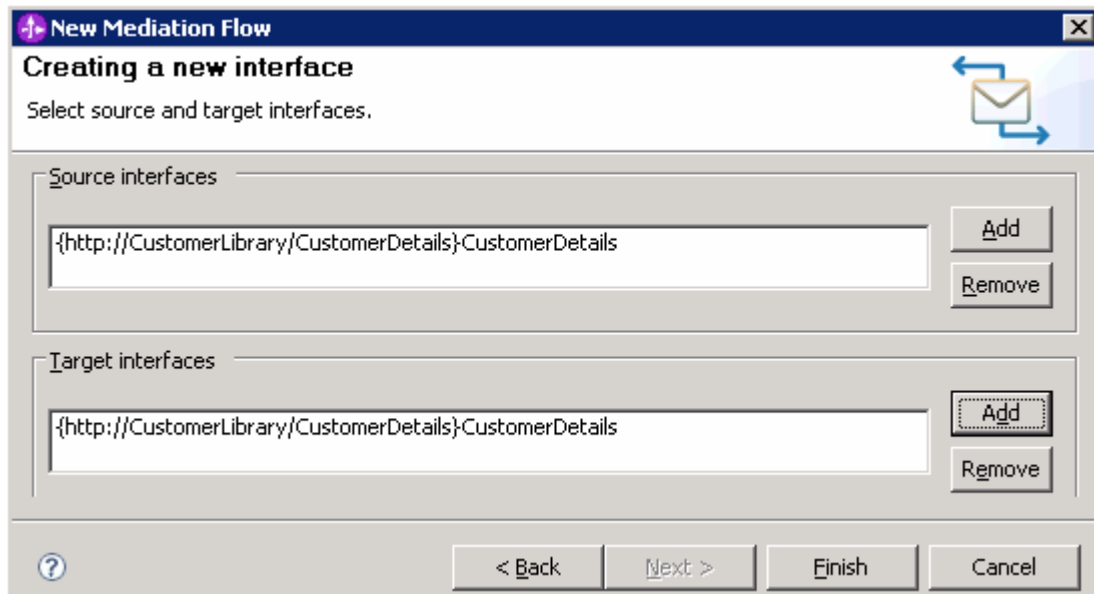


2) Click **OK** over the **Interface Selection** pop-up window

3) Similarly, click the **Add** button next to the **Target interfaces** text area and select **CustomerDetails** from the **Interface Selection** pop-up panel

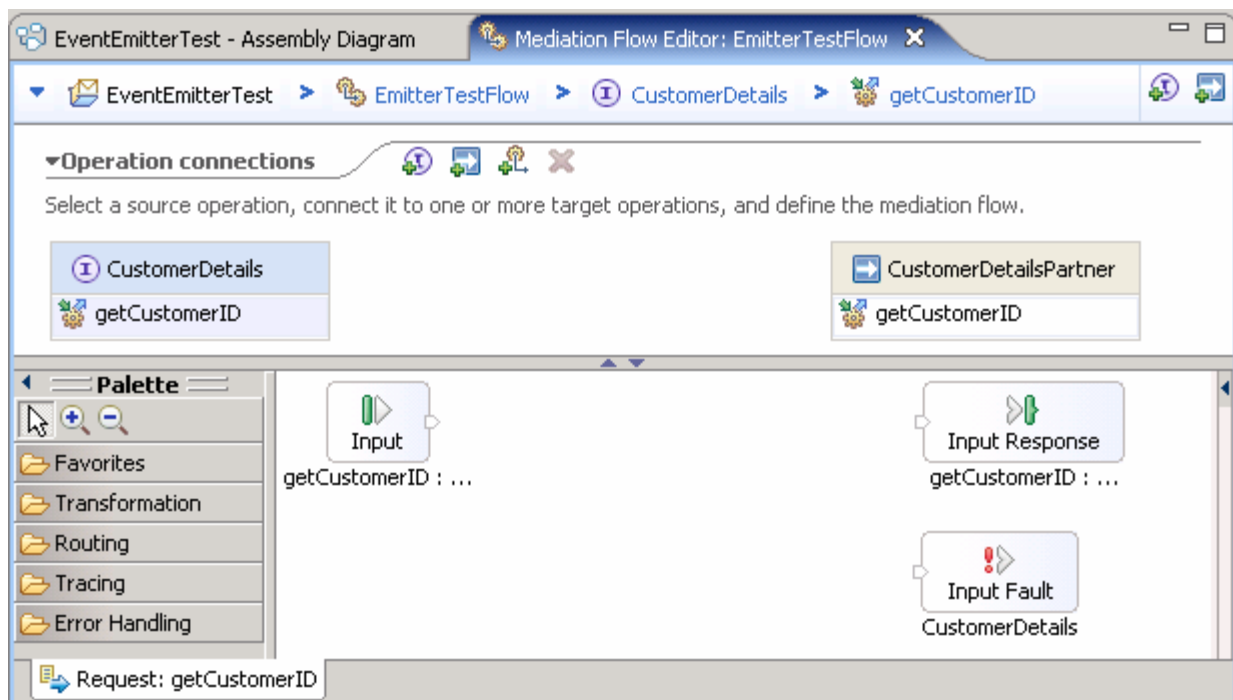
4) Click **OK** over the **Interface Selection** pop-up panel

5) The added source and target interfaces should look like in the picture shown below:



6) Click **Finish**

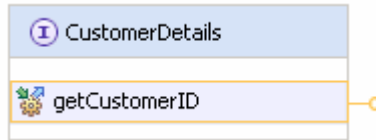
___ e. Upon creation of the **EmitterTestFlow** Mediation Flow, it is opened in the Mediation Flow Editor as shown in the picture below:



Create a request flow:

To create the request flow, complete these steps:

- ___ 3. In the EmitterTestFlow's mediation flow editor, connect the **source** to **target operations** in the Operation Connections view
 - ___ a. Click anywhere on the source operation, **CustomerDetails/getCustomerID** on the left side of the Operation Connections view



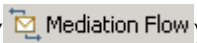
- ___ b. Drag to the target operation, **CustomerDetailsPartner/getCustomerID** on the right side of the Operation Connections view and release the mouse click

Alternative: Right-click on the source component, **CustomerDetails/getCustomerID** and select **Create an operation connection** and then click on the target operation, **CustomerDetailsPartner/getCustomerID**



- ___ c. Click on the black arrow (wire) to view the Mediation Flow View and ensure the **Request** tab is selected to build the Request flow

- ___ 4. Add an **Message Filter** primitive to the Request Mediation Flow diagram canvas

- ___ a. In the **Mediation Flow Editor**(middle window), click the **Message Filter** icon ( Mediation Flow) to select the Message Filter primitive from the routing palette tray on the left side and drop it into the canvas between the Input Request node and the Callout Request Node

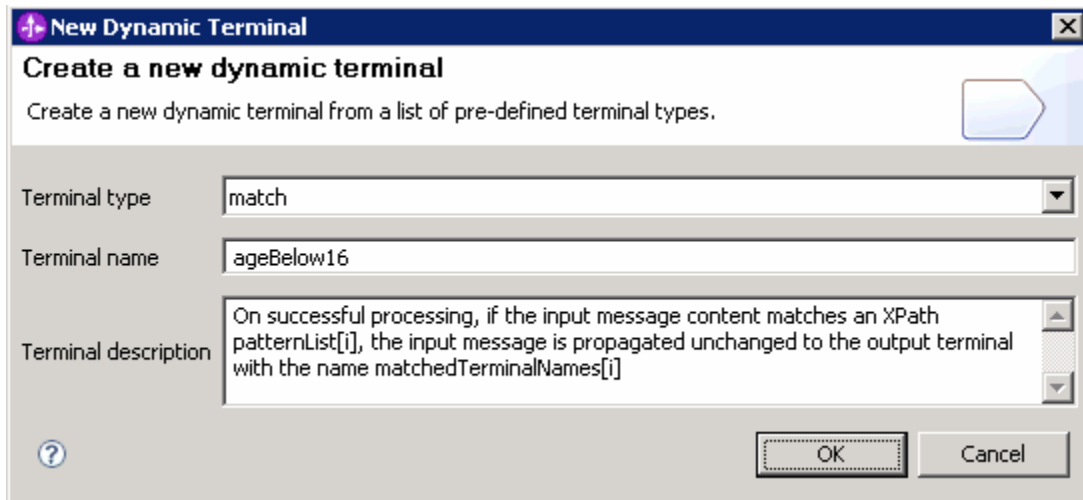


- ___ b. Hover the mouse over the **Input** node's output terminal and drag the handle that appears, to the input terminal of the Message Filter primitive, **MessageFilter1**

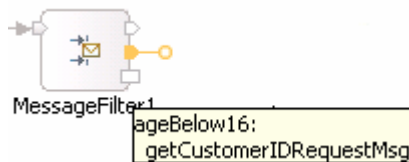




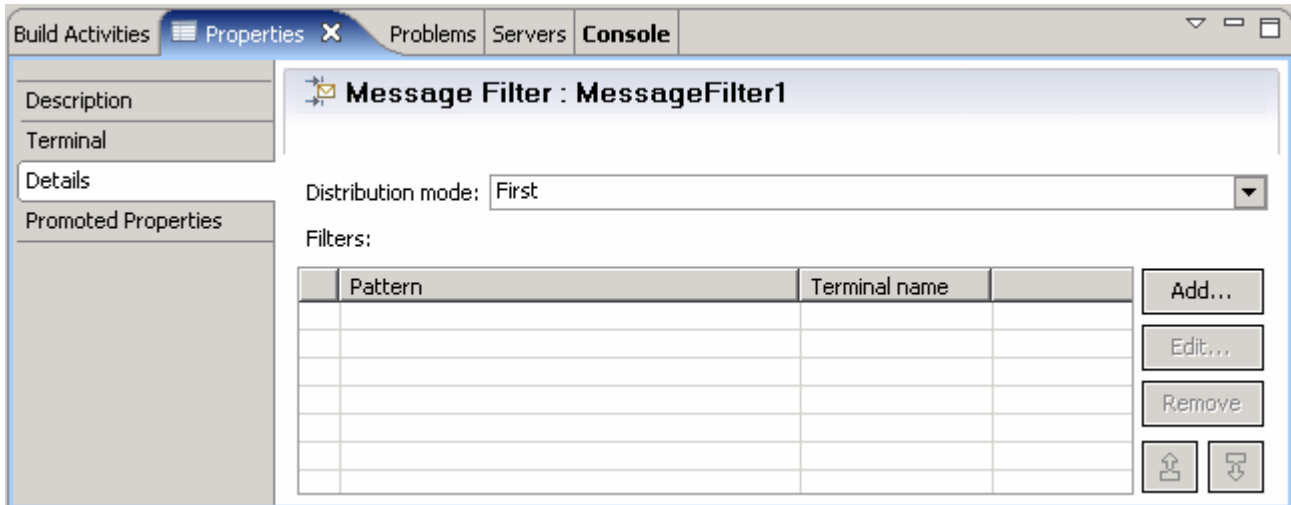
- ___ c. Right Click on the Message Filter primitive, **MessageFilter1** and select **Add Output Terminal** from the pop-up menu
- ___ d. Enter **ageBelow16** for the **Terminal name** field



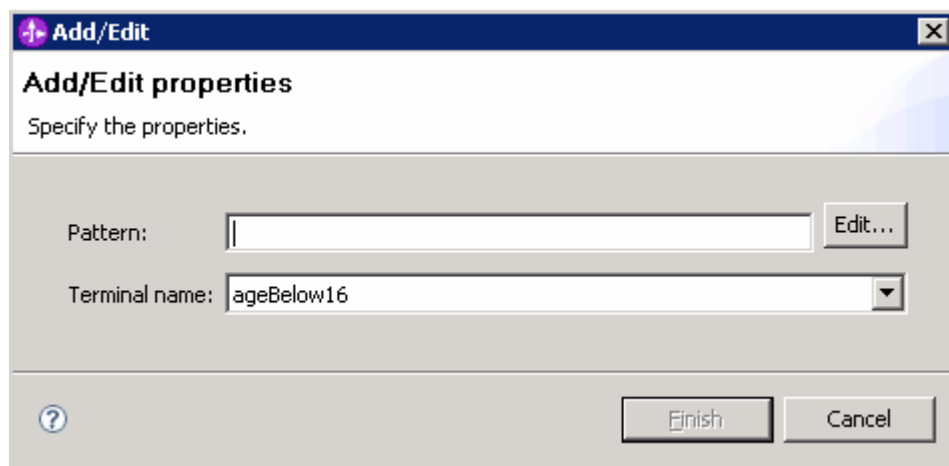
- ___ e. Click **OK**
- ___ f. The Message Filter primitive should look like in the picture shown below with a new Output terminal added:



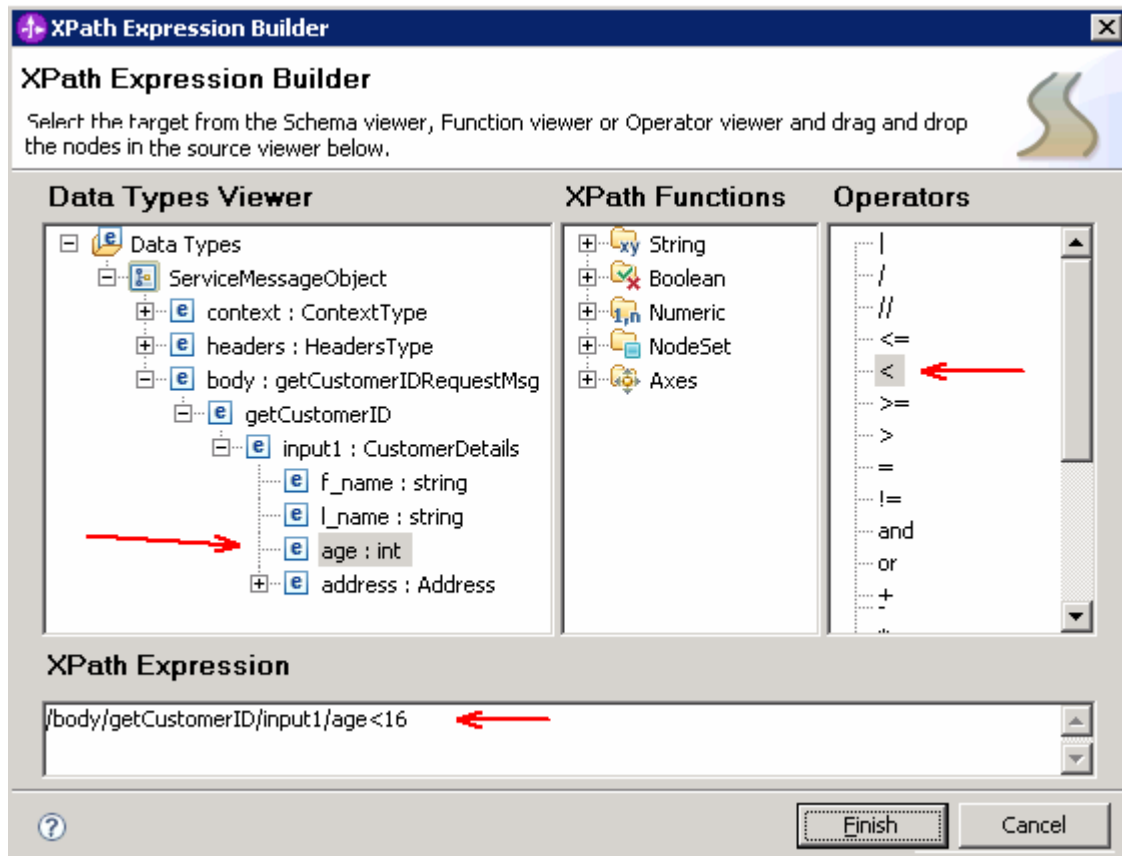
- ___ g. Select the Message Filter primitive, **MessageFilter1** and select the **Details** tab under the properties view



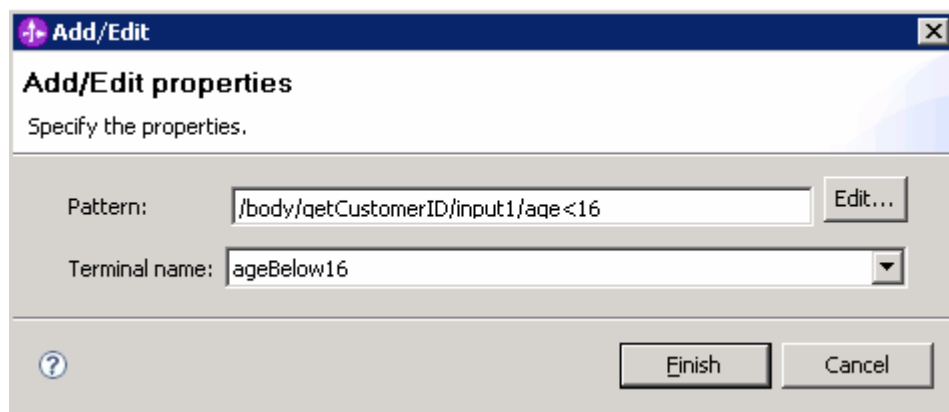
- ___ h. Click the **Add** button for **Filters**
- ___ i. The '**Add/Edit properties**' panel opens



- ___ j. From the **Add/Edit properties** panel, click the **Edit** button to set the Pattern. The **XPath Expression Builder** wizard opens
- ___ k. In the **XPath Expression Builder** wizard, navigate to the **age** element (`ServiceMessageObject>body>getCustomerID>input1>age`) in the '**Data Types Viewer**' section, and alter the XPath expression to `/body/getCustomerID/input1/age<16` as follows:
 - 1) Select the **age** element and double click to populate the expression to the '**XPath Expression**' text area
 - 2) Now append '`< 16`' to `/body/getCustomerID/input1/age` (`/body/getCustomerID/input1/age<16`).



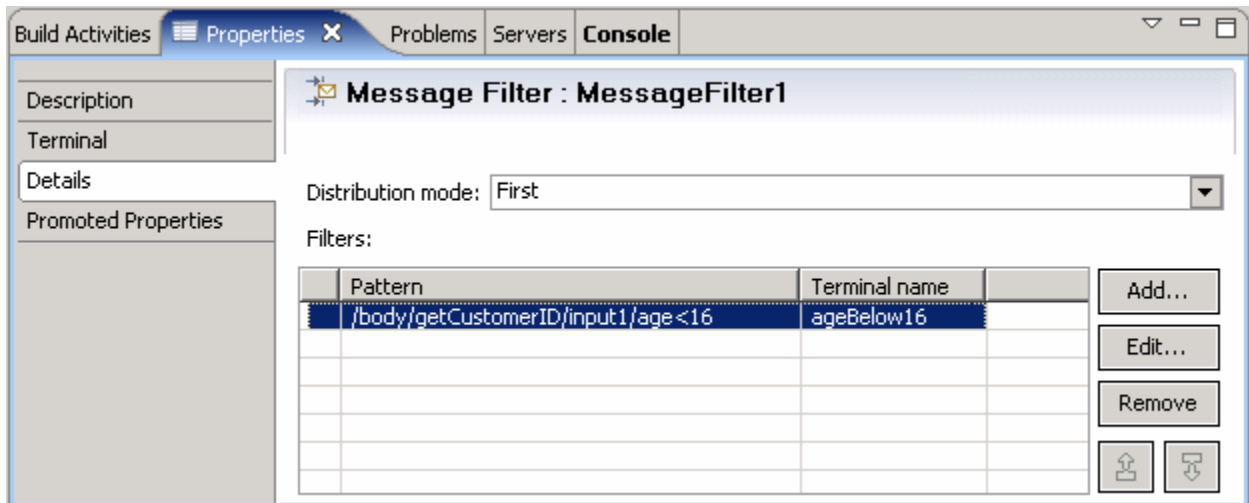
__ l. Click **Finish**

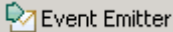


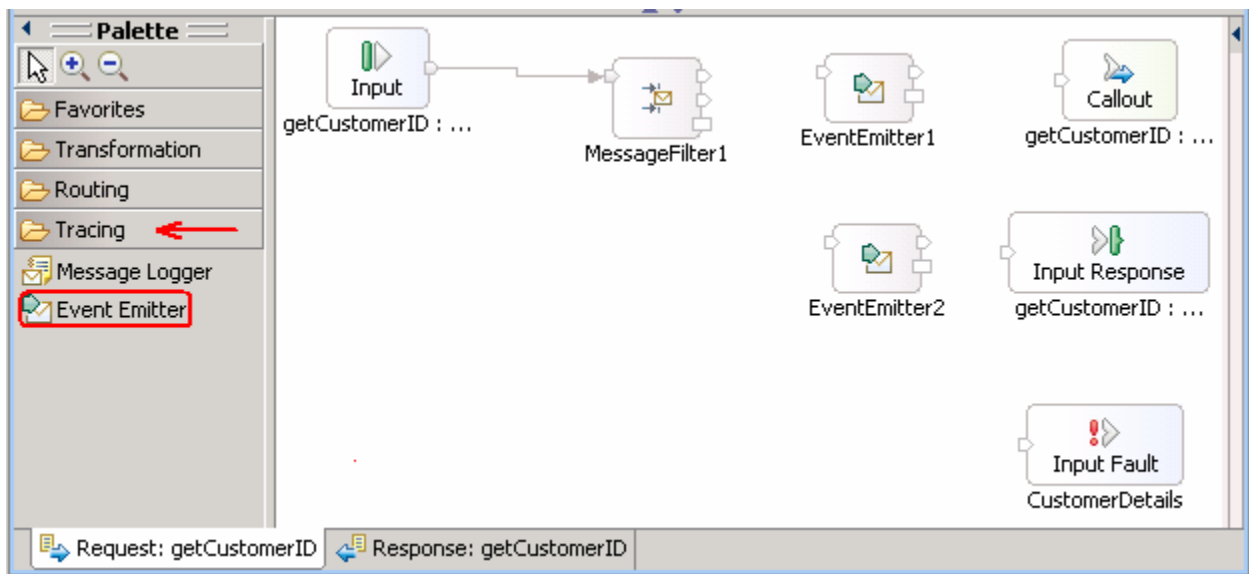
__ m. Select **'ageBelow16'** for 'Terminal name'

__ n. Click **Finish** over the **'Add/Edit properties'** panel

__ o. The Message Filter Primitive Filter added should look as shown below:

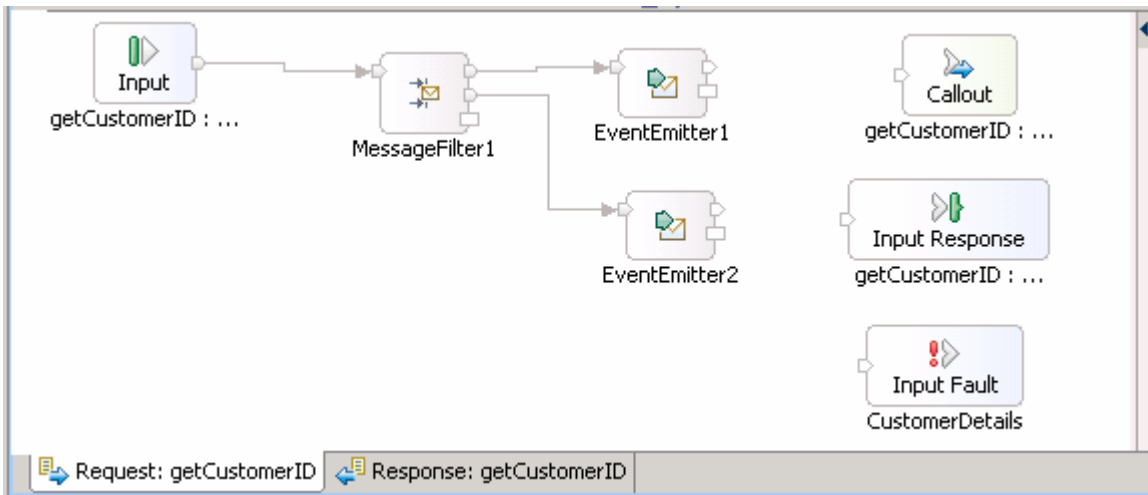


___ p. In the **Mediation Flow Editor**(middle window), click on the **Event Emitter** icon () to select the Event Emitter primitive from the tracing palette on left side of view and drop **two** of them into the canvas between the, **MessageFilter1** and the Callout Request Node as shown below:

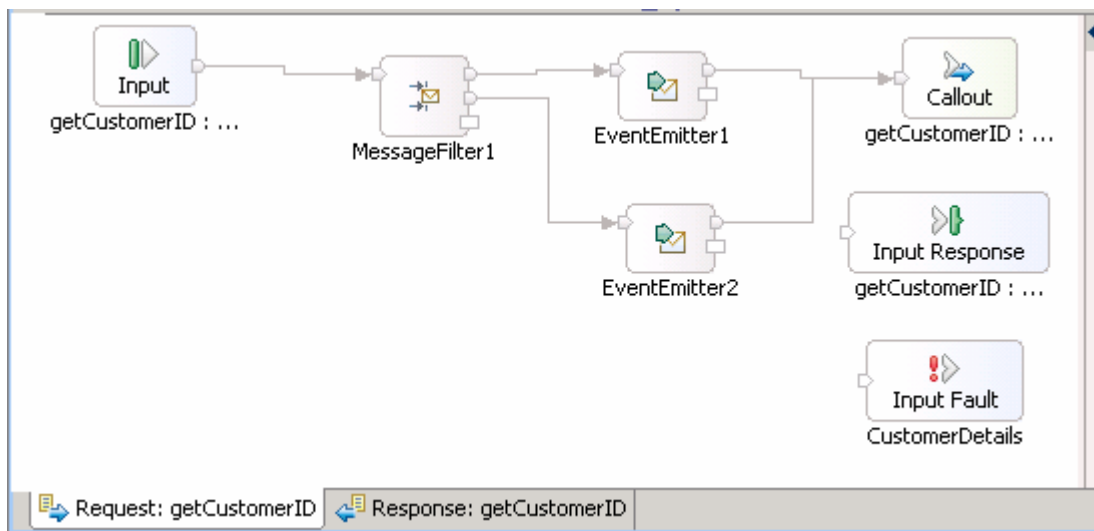


___ q. Wire the Message Filter primitive, **MessageFilter1** and the Event Emitter primitives, **EventEmitter1** and **EventEmitter2** as shown below:

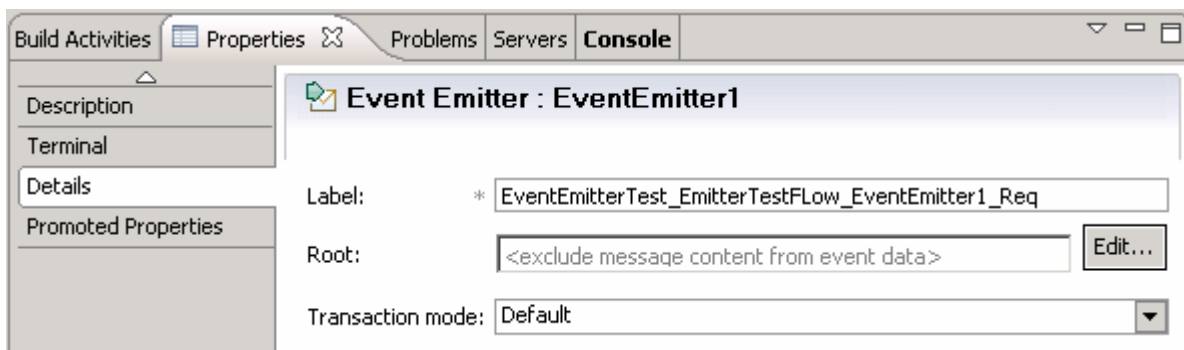
Note: The **ageBelow16** output terminal of the Message Filter is wired to the input terminal of **EventEmitter2** primitive. The **default** output terminal is wired to the input terminal of **EventEmitter1** primitive.



__ r. Also wire the output terminals of the Event Emitter primitives, **EventEmitter1** and **EventEmitter2** to the input terminal of the Request **Callout** node as shown below:

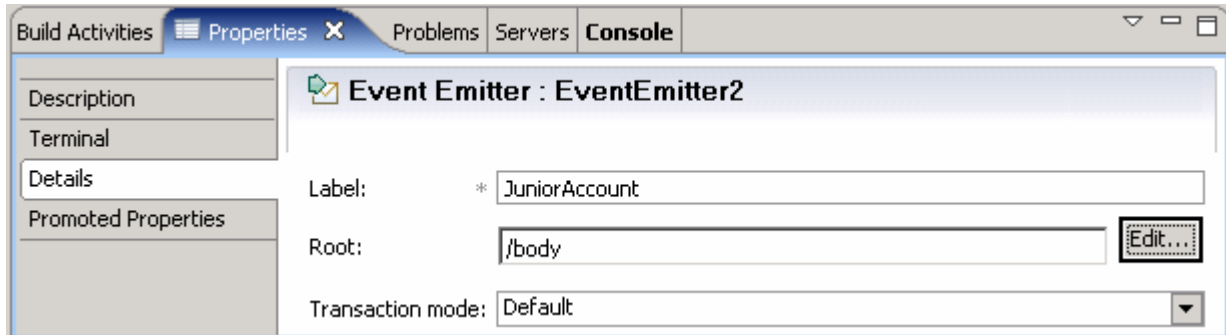


__ s. Select the Event Emitter primitive, **EventEmitter1** and select the **Details** tab under the properties view. Accept the default values for this primitive and should match the diagram below:



__ t. Select the Event Emitter primitive, **EventEmitter2** and select the **Details** tab under the properties view

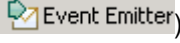
- 1) Enter **JuniorAccount** for the **Label**
- 2) Click the **Edit** button for **Root** and then select **'/body'** as XPath Expression
- 3) Leave the **Transaction mode** as default with the value, **Default**

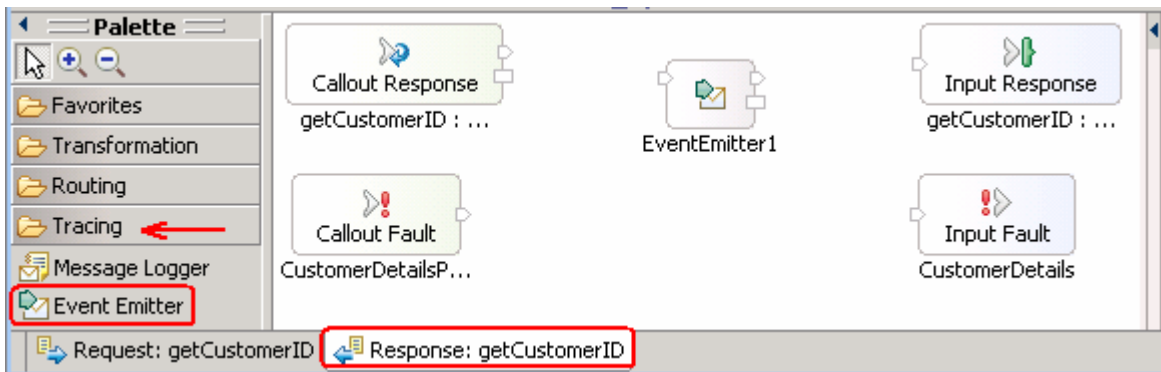


__ u. Save all work (**File → Save All** or **Ctrl + Shift + S**)

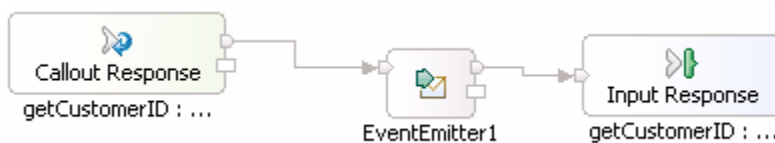
Create a Response Flow:

___ 5. Create the Request Flow by completing these steps:

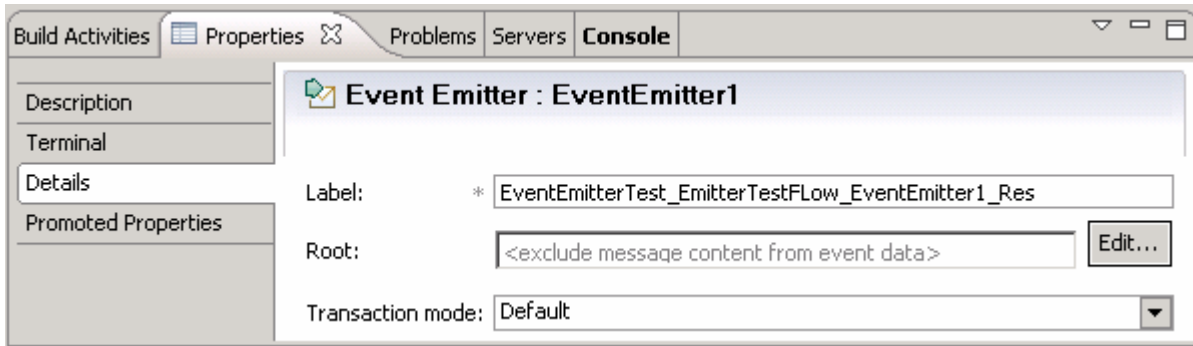
- __ a. While you are in the Mediation Flow Editor(middle section), select the **Response** tab
- __ b. In the **Mediation Flow Editor**(middle), click on the **Event Emitter** icon ( Event Emitter) to select the Event Emitter primitive from the tracing pallet on the left side and drop into the canvas as shown below:



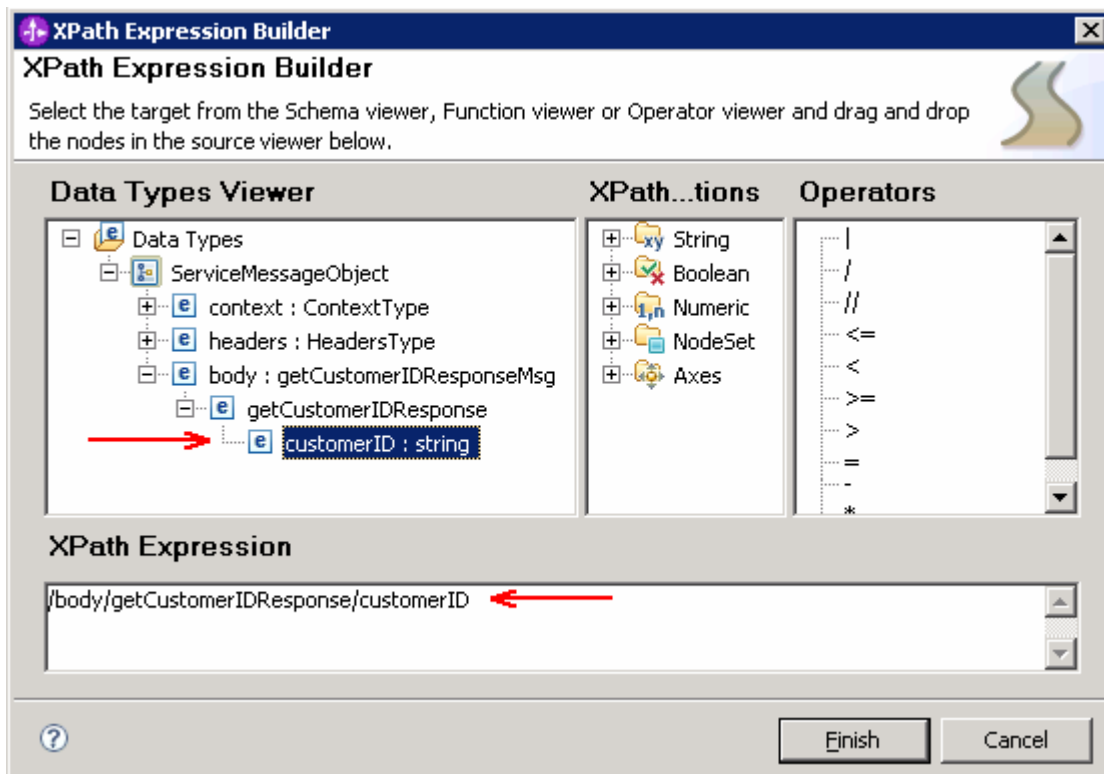
__ c. Wire the Event Emitter primitive, **EventEmitter1** as shown below:



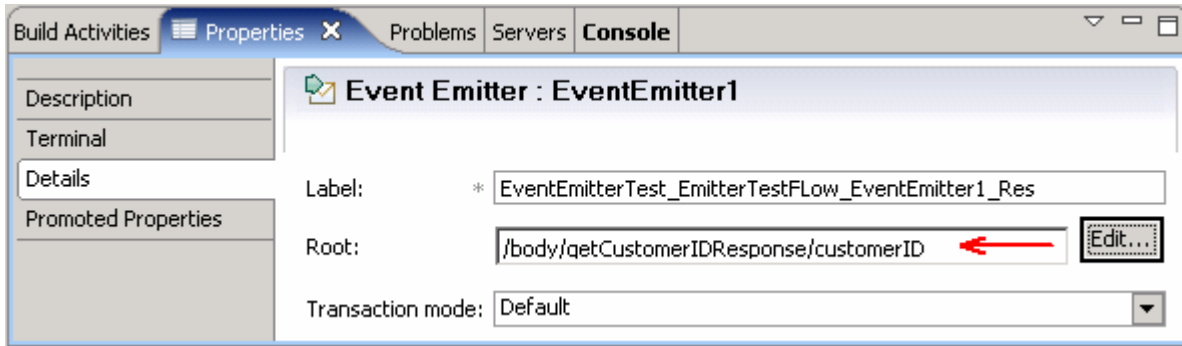
__ d. Select the Event Emitter primitive, **EventEmitter1** in the Response Flow and select the **Details** tab under properties view



- ___ e. Click the **Edit** button for the **Root** property. The **XPath Expression Builder** wizard opens
- ___ f. In the **XPath Expression Builder** wizard, navigate to the **customerID** element; select and double click to populate the XPath Expression as shown below:



- ___ g. Click **Finish**
- ___ h. Accept the remaining properties as default. The properties view for the **EventEmitter1** primitive of the Response Flow should look like in the picture below:

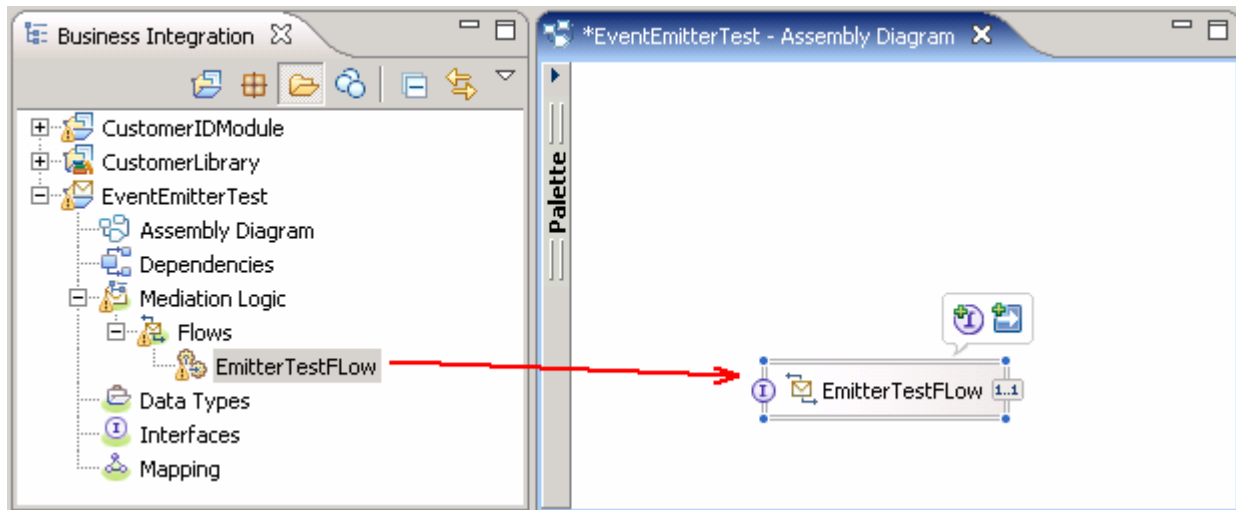


___ i. Save all work (**File → Save All** or **Ctrl + Shift + S**)

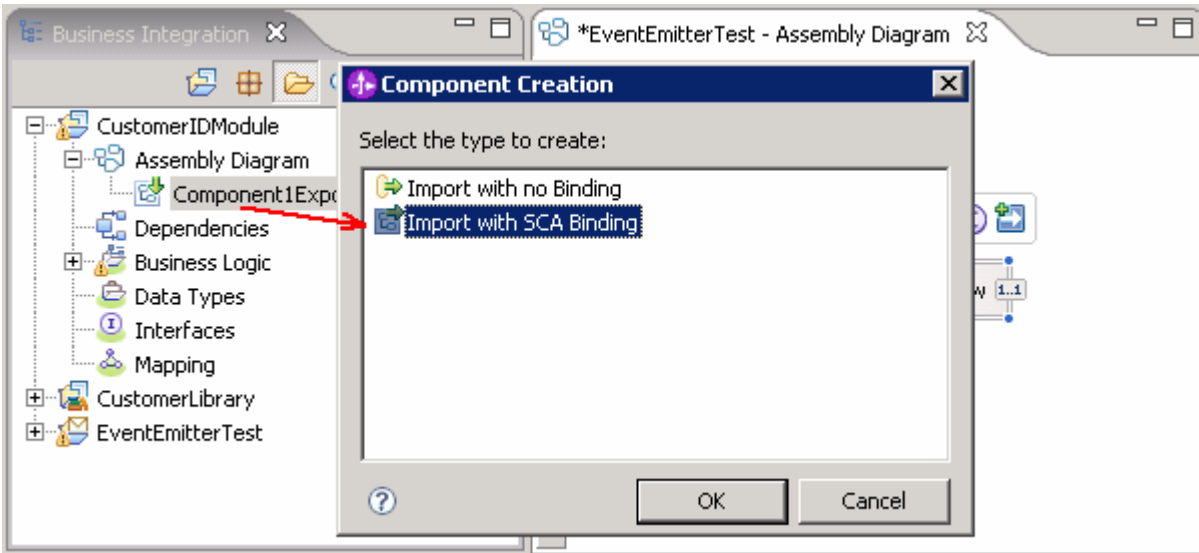
Visually compose the Mediation Module:

___ 6. To Visually compose the Mediation Module, complete these steps:

- ___ a. In the Business Integration view, expand the **EventEmitterTest** mediation module and double-click the mediation module assembly (Assembly Diagram) to open it with the assembly editor
- ___ b. An empty assembly editor for the **EventEmitterTest** mediation module is opened
- ___ c. In the Business Integration view, expand the **EventEmitterTest → Mediation Logic → Flows**; select **EmitterTestFlow** mediation flow and then drag it over the assembly editor's canvas as shown below:



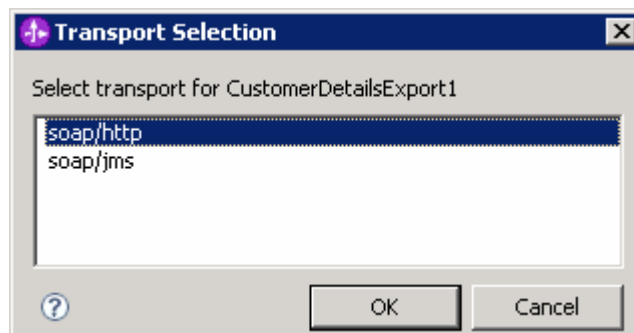
___ d. In the Business Integration view, expand the **CustomerIDModule → Assembly Diagram**; select **Component1Export** an then drag it over the assembly editor's canvas



- ___ e. Select **Import with SCA Binding** from the **Component Creation** panel that pops-up upon dragging the **Component1Export**. Click **OK**
- ___ f. The assembly diagram should look like in the picture shown below:



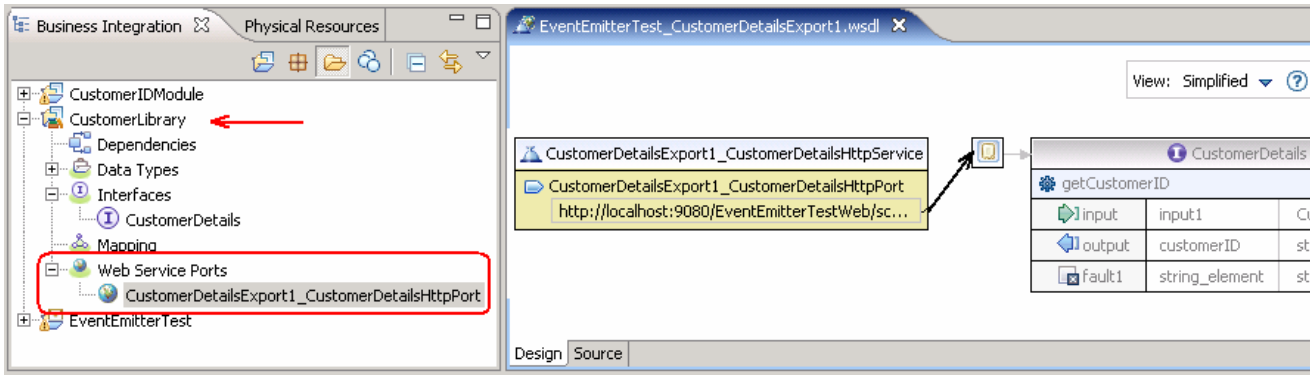
- ___ g. Right click the **EmitterTestFlow** component and select **Generate Export → Web Service Binding** from the pop-up menu
- ___ h. Select **soap/http** from the Transport Selection pop-up window



- ___ i. Click **OK**
- ___ j. The assembly diagram must look like the diagram shown below:



- ___ k. You should see a '**Web Service Port**', that is a **CustomerDetailsExport1_CustomerDetailsExport1.wsdl** file generated under '**CustomerLibrary**' library as shown below:



___ l. Wire the mediation Flow, **EmitterTestFlow** and **Import1**



___ m. Save all work (**File** → **Save All** or **Ctrl + Shift + S**)

___ 7. The Mediation Module and Mediation Flow creation is complete

Part 3: Deploy the modules to the test server


In this part of the lab, the Mediation Module is published to the WebSphere ESB Server 6.1 test server.

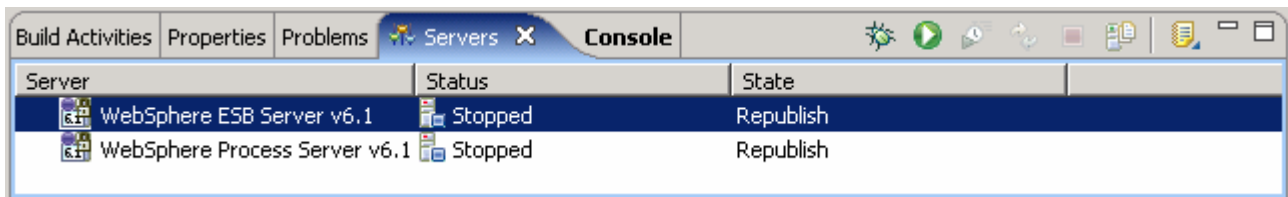
- ___ 1. Start the **WebSphere ESB Server** if not started and **add modules** to the server

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 ESB Integrated test environment, complete the steps below:

- ___ a. Open Servers View

- ___ b. Select the '**WebSphere ESB Server v6.1**' and right-click to select "() **Start**" from the context menu



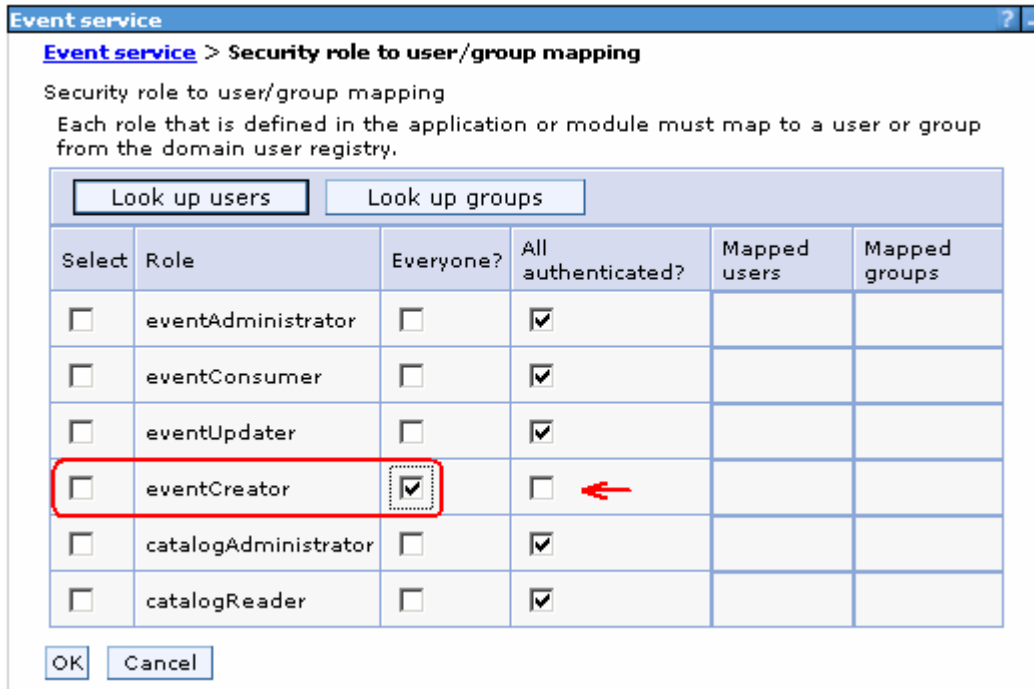
- ___ c. This action takes some time. Wait for the server to start

- ___ 2. (Optional) If the WebSphere Enterprise Service Bus server default is security is enabled, complete these instruction to disable the event creator security:

Note: In this lab, the application is an event source and needs to submit events to an emitter using synchronous EJB calls to the CEI server. Since you are using a 'Web Service Explorer' to submit data, the '**eventCreator**' role is not authenticated. In order to overcome this situation, you need to allow every one as an event creator role.

- ___ a. Right-click on the '**WebSphere ESB Server v6.1**' in the Servers view and select **Run administration console** from the context menu
- ___ b. Log-in to the administration console.
- ___ c. In the left navigation menu, expand **Service integration** → **Common Event Infrastructure** and then click the '**Event service**' link
- ___ d. In the '**Event service**' page, click the '**Map security roles to users or groups**' link under the '**Additional Properties**' section to right side
- ___ e. Clear the check box for '**eventCreator**' under '**All authenticated?**' column and then select the check box under the '**Everyone?**' column as shown below:

Note: When you map everyone to a role, anyone can access the resources that are protected by this role and, essentially, there is no security.



__ f. Click **OK**

__ g. Save to the master configuration

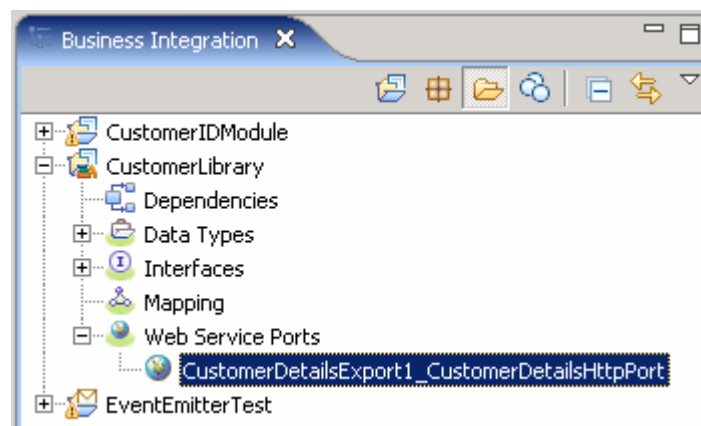
___ 3. Add the **projects** to WebSphere ESB Server. In the servers view, right-click on WebSphere ESB Server v6.1 and select '**Add and Remove Projects...**' from the context menu

__ a. Click the **Add-All>>** button to move all projects to server

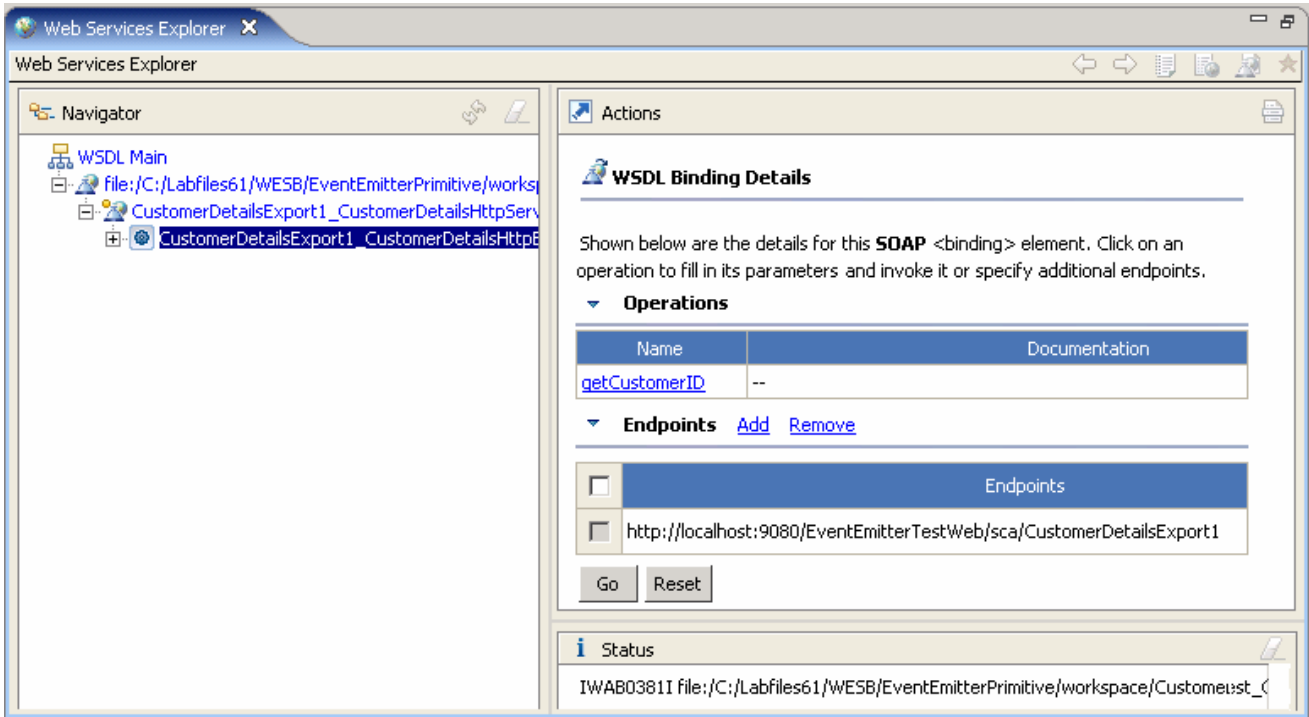
__ b. Click **Finish**

__ c. Wait for the deployment to finish.

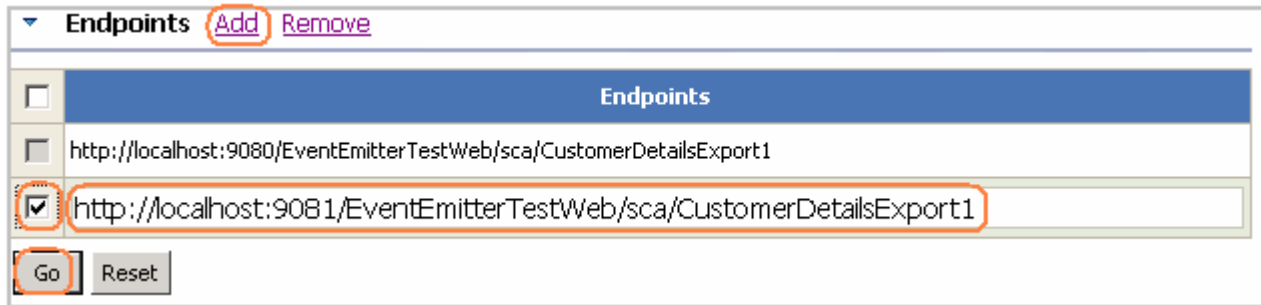
___ 4. While you are in the business integration perspective, expand the **CustomerLibrary** project; right-click the **CustomerDetailsExport1_CustomerDetailsHttpPort**, web services port and then select '**Web Services → Test with Web Services Explorer**' from the context menu



5. The **EventEmitterTest_CustomerDetailsExport1.wsdl** file opens in the Web Services Explorer as shown below:



Note: If you are running against a server other than the localhost and or on non-default ports you will need to configure different endpoints for the Web services explorer to use. To configure the a new Endpoint, click the **Add** link next to **Endpoints**, and then the modify the endpoints URL to match your environment, select the check box next to the text filed and click on the **Go** button. Ensure that the Endpoints are successfully updated.

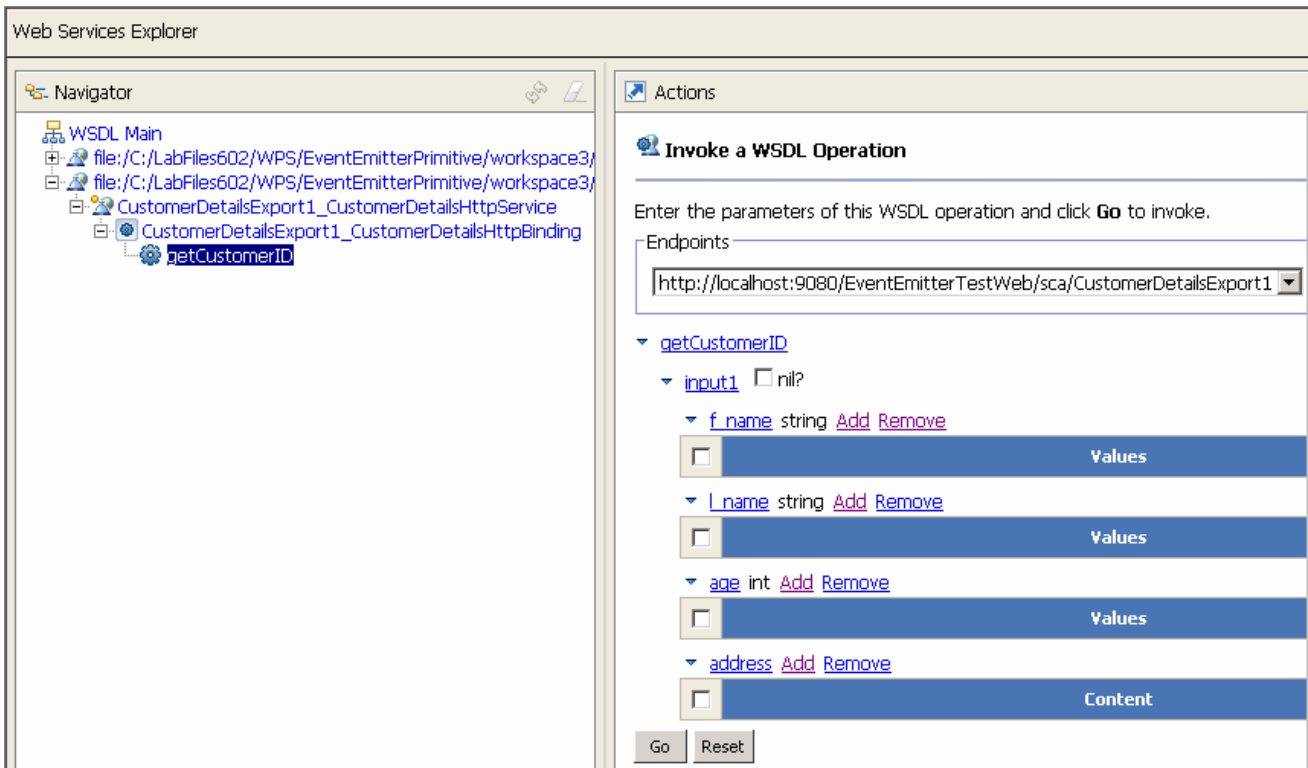


Part 4: Event generation test A and B using the Web services explorer

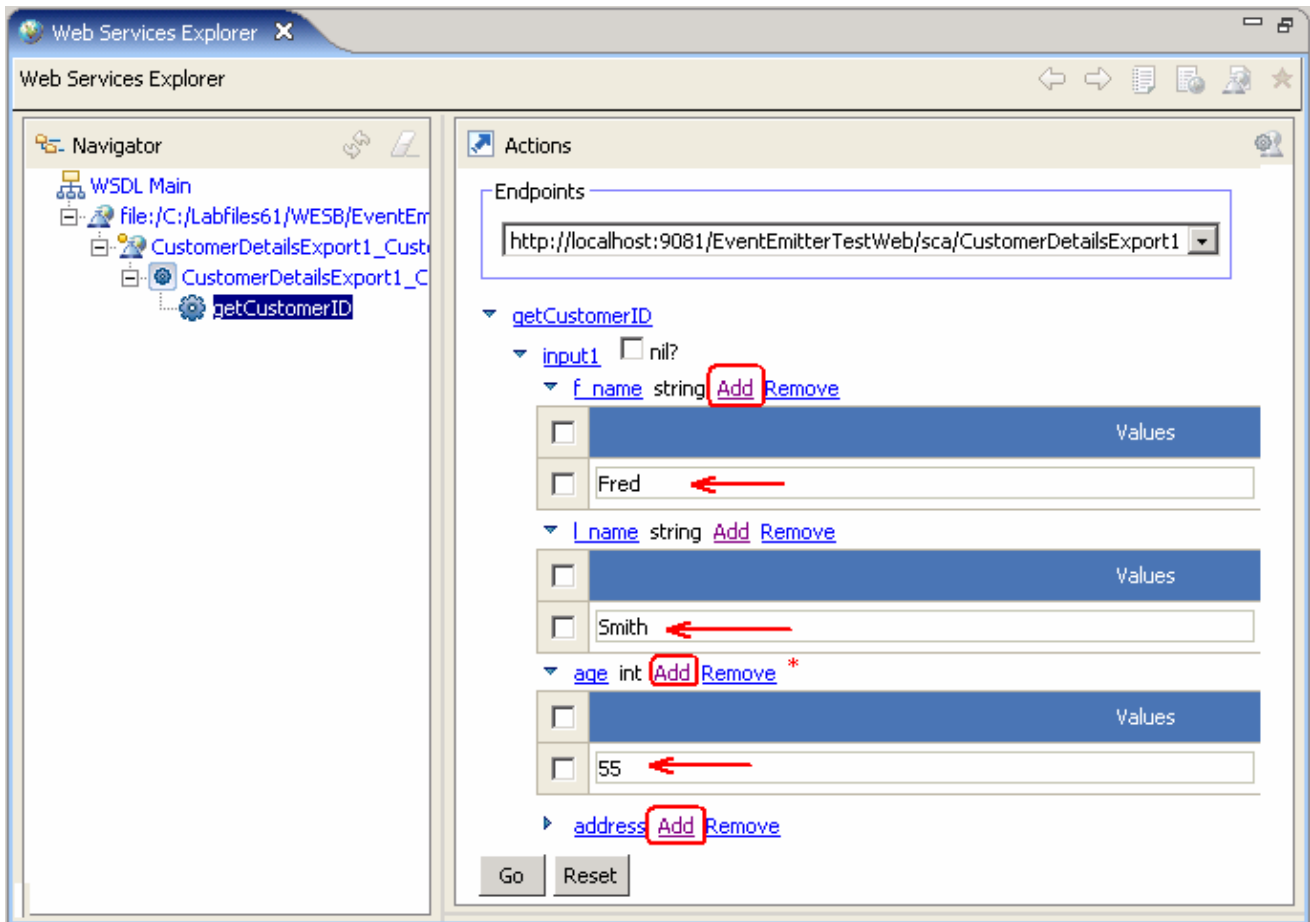
In this section of the lab two tests, test A and test B, are performed to produce events at the CEI Server. The Test A is to generate the request and response events with the value of the age element of the in the CustomerDetails business object; **greater than 16** (>16). The Test B is to do the same with the value of the age **less than 16** (<16).

Event Generation Test A

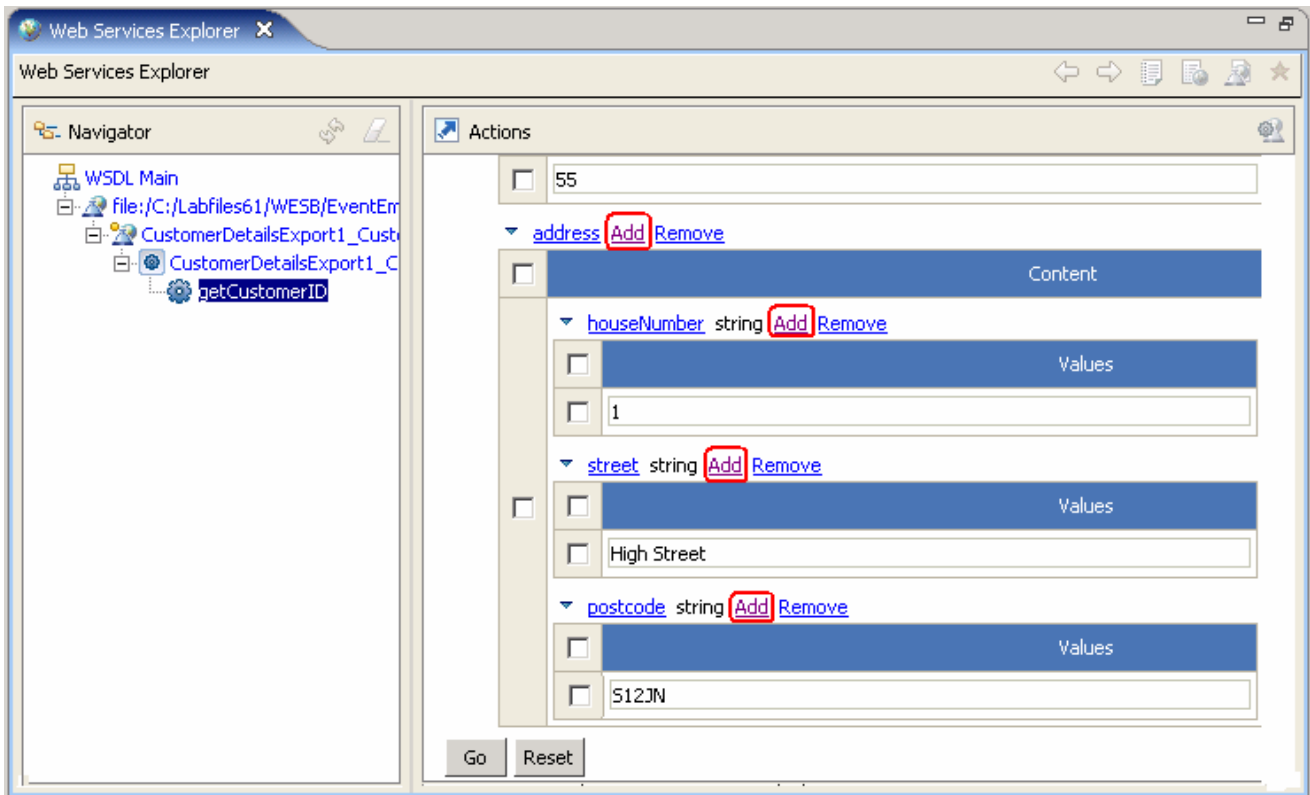
1. In the Web Services Explorer's Navigator pane, expand **CustomerDetailsExport1_CustomerDetailsHttpBinding** and select the operation **getCustomerID** as shown below:



2. Enter these values in the fields on the Actions pane of the explorer by clicking on the **Add** link to add a blank field for each of the values below:
 - a. f_name: **Fred**
 - b. l_name: **Smith**
 - c. age : **55**



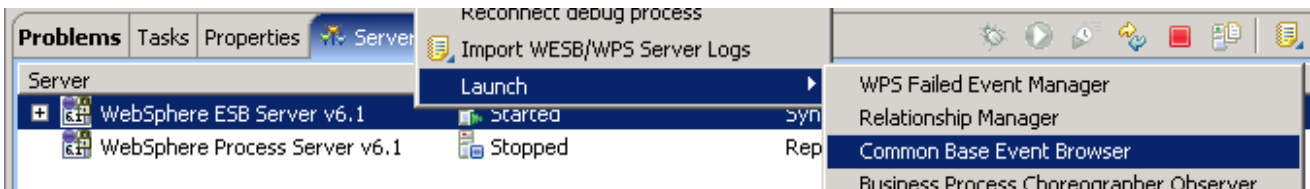
- ___ 3. Scroll down and click the **Add** link next **address** to add the address fields. Enter the following values for the address fields:
- ___ a. houseNumber: **1**
 - ___ b. street : **High Street**
 - ___ c. postcode : **S12JN**



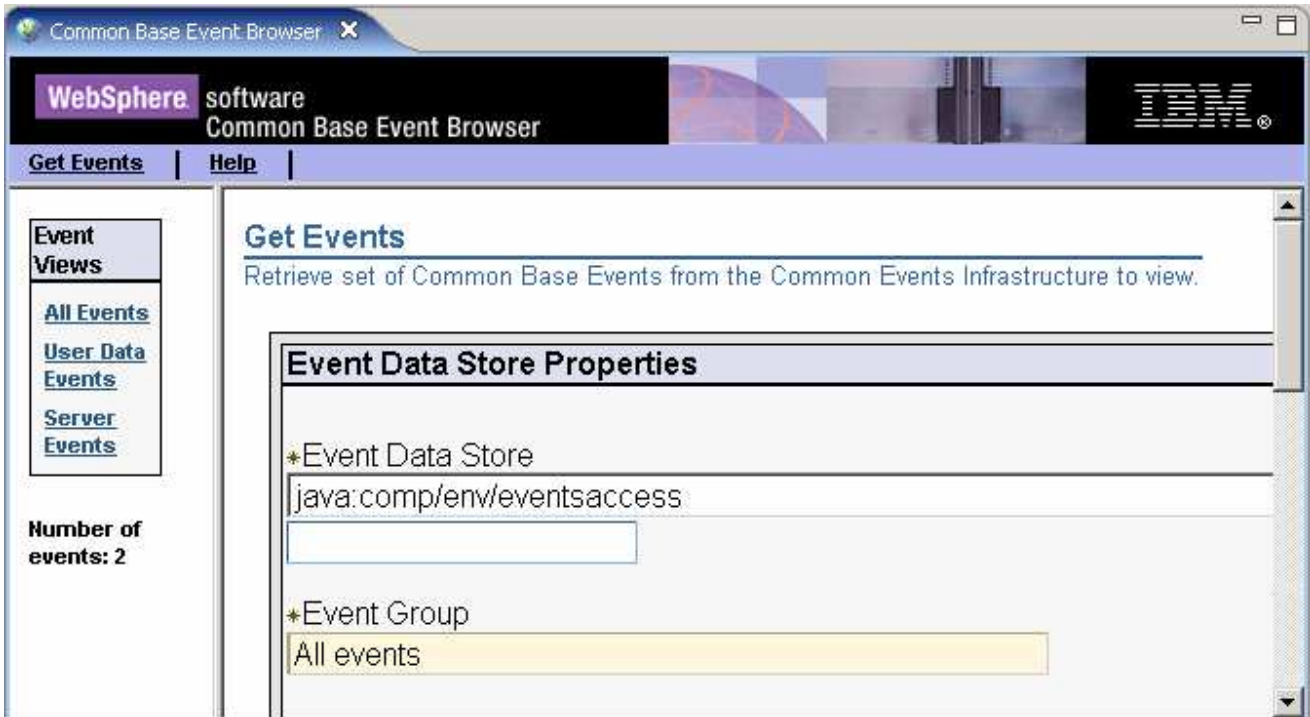
- ___ 4. Click the **Go** button to generate the Events
- ___ 5. Ensure that there are no errors displayed in the **Status** pane at the bottom



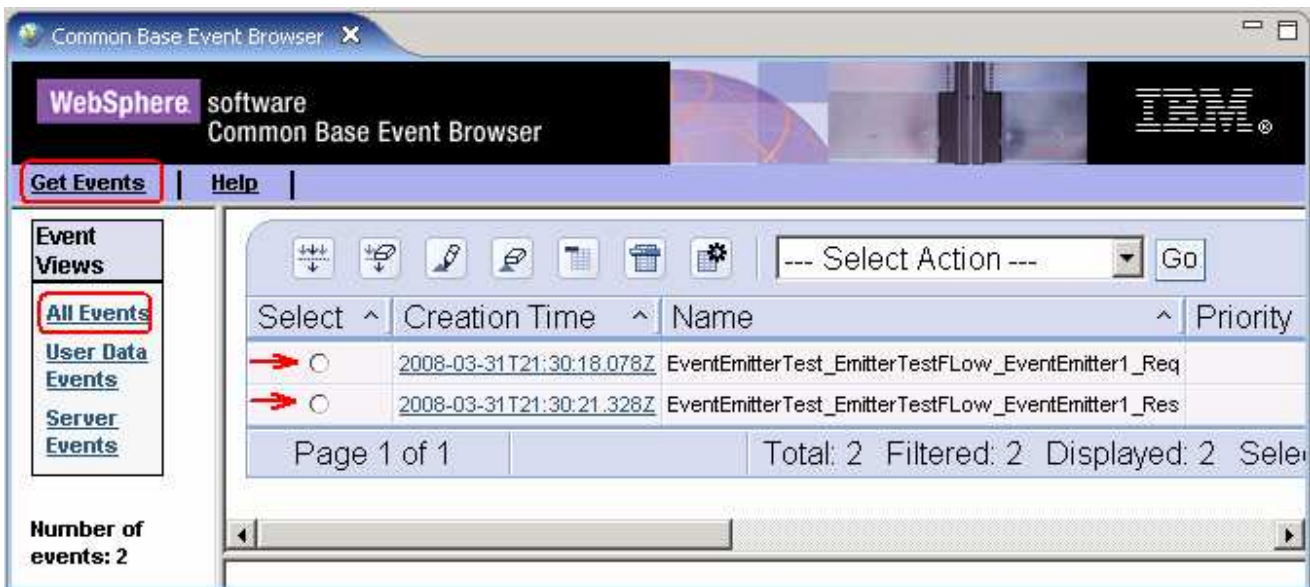
- ___ 6. To confirm that the Events are generated, open the **Common Base Event browser**:
 - ___ a. Right click on '**WebSphere ESB Server v6.1**' in the servers view and select the '**Launch → Common Base Event Browser**' from the context menu



- ___ b. Enter the login credentials and click **Login** if security is enabled



7. From the Event Browser, on the top left of the navigation menu, click on the **Get Events** link and then the **All Events** link under Events View. Ensure that two events have been generated; one for **EventEmitter1** on the request flow and the other for **EventEmitter1** on the response flow as shown below:



8. To view the contents of the Request Flow **EventEmitter1**, click on the Creation Time link or select the radio button next to **EventEmitterTest_EmitterTestFlow_EventEmitter1_Req**. The common base event browser displays the events in the Event Data pane
9. Scroll down to the bottom of the 'Event Data' pane and click the **'wbi:event'** link

wbi:event ← <wbi:event xmlns:wbi="http://www.ibm.com/xmlns/prod/websphere/monitoring/6.1" xmlns:xsi="http://w...

wbi:event

Show the details for the Common Base Event path wbi:event

expand

collapse

```
<wbi:event>
  <wbi:eventHeaderData>
    <wbi:WBISESSION_ID> 9.3.75.99;EventEmitterTest;;getCustomerID;1206999014328;140642402</wbi:WBISESSION_ID>
    <wbi:ECSCurrentID> 9.3.75.99;EventEmitterTest;null;;getCustomerID;1206999014328;140642402</wbi:ECSCurrentID>
    <wbi:ECSParentID> 9.3.75.99;EventEmitterTest;;getCustomerID;1206999014328;140642402</wbi:ECSParentID>
    <wbi:WBIEventVersion> 6.1</wbi:WBIEventVersion>
  </wbi:eventHeaderData>
  <wbi:eventPointData xsi:type="esb:WESB_ESBEventMediation.CUSTOM" >
    <wbi:eventNature> CUSTOM</wbi:eventNature>
    <wbi:payloadType> full</wbi:payloadType>
    <esb:ModuleName> EventEmitterTest</esb:ModuleName>
    <esb:MediationName> EventEmitter1</esb:MediationName>
    <esb:EventEmitterLabel> EventEmitterTest_EmitterTestFlow_EventEmitter1_Req</esb:EventEmitterLabel>
  </wbi:eventPointData>
</wbi:event>
```

Close

___ 10. Click the 'Close' button

___ 11. Look for the following wbi elements in the event to ensure the correct data has been generated:-

___ a. **EventEmitterTest_EmitterTestFlow_EventEmitter1_Req**

___ b. ModuleName → **EventEmitterTest**

___ c. MediationName → **EventEmitter1**

___ 12. To view the contents of the Response Flow **EventEmitter1**, click on the Creation Time link or select the radio button next to **EventEmitterTest_EmitterTestFlow_EventEmitter1_Res**. The common base event browser will display the events in the Event Data pane as shown below:

wbi:event

Show the details for the Common Base Event path wbi:event

expand

collapse

```

<wbi:event>
  <wbi:eventHeaderData>
    <wbi:WBISESSION_ID> 9.3.75.99;EventEmitterTest;;getCustomerID;1206999014328;140642402</wbi:WBISESSION_ID>
    <wbi:ECSCurrentID> 9.3.75.99;EventEmitterTest;null;;getCustomerID;1206999014328;140642402</wbi:ECSCurrentID>
    <wbi:ECSParentID> 9.3.75.99;EventEmitterTest;;getCustomerID;1206999014328;140642402</wbi:ECSParentID>
    <wbi:WBIEventVersion> 6.1</wbi:WBIEventVersion>
  </wbi:eventHeaderData>
  <wbi:eventPointData xsi:type="esb:WESB.ESBEventMediation.CUSTOM" >
    <wbi:eventNature> CUSTOM</wbi:eventNature>
    <wbi:payloadType> full</wbi:payloadType>
    <esb:ModuleName> EventEmitterTest</esb:ModuleName>
    <esb:MediationName> EventEmitter1</esb:MediationName>
    <esb:EventEmitterLabel> EventEmitterTest_EmitterTestFlow_EventEmitter1_Res</esb:EventEmitterLabel>
    <esb:Root> /body/getCustomerIDResponse/customerID</esb:Root>
  </wbi:eventPointData>
  <wbi:applicationData>
    <wbi:content wbi:name="Message" >
      <wbi:value xsi:type="xsd:string" > ABCDE12345</wbi:value>
    </wbi:content>
  </wbi:applicationData>
</wbi:event>

```

Close

___ 13. Look for the following wbi elements in the event to ensure the correct data has been generated:-

- ___ a. **EventEmitterTest_EmitterTestFlow_EventEmitter1_Res**
- ___ b. ModuleName → **EventEmitterTest**
- ___ c. MediationName → **EventEmitter1**
- ___ d. Root → **/body/getCustomerIDResponse/customerID**
- ___ e. Message → **ABCD12345**

___ 14. The Event Generation Test A is complete

Event Generation Test B

___ 15. In the Web Services Explorer's Navigator pane, expand **CustomerDetailsExport1_CustomerDetailsHttpBinding** and select the operation **getCustomerID** and enter the following values in their respective fields:

(Note that you might need to delete the old values from their fields and replace them with these new ones)

- ___ a. f_name : **Harry**
- ___ b. l_name : **Brown**
- ___ c. age : **13**

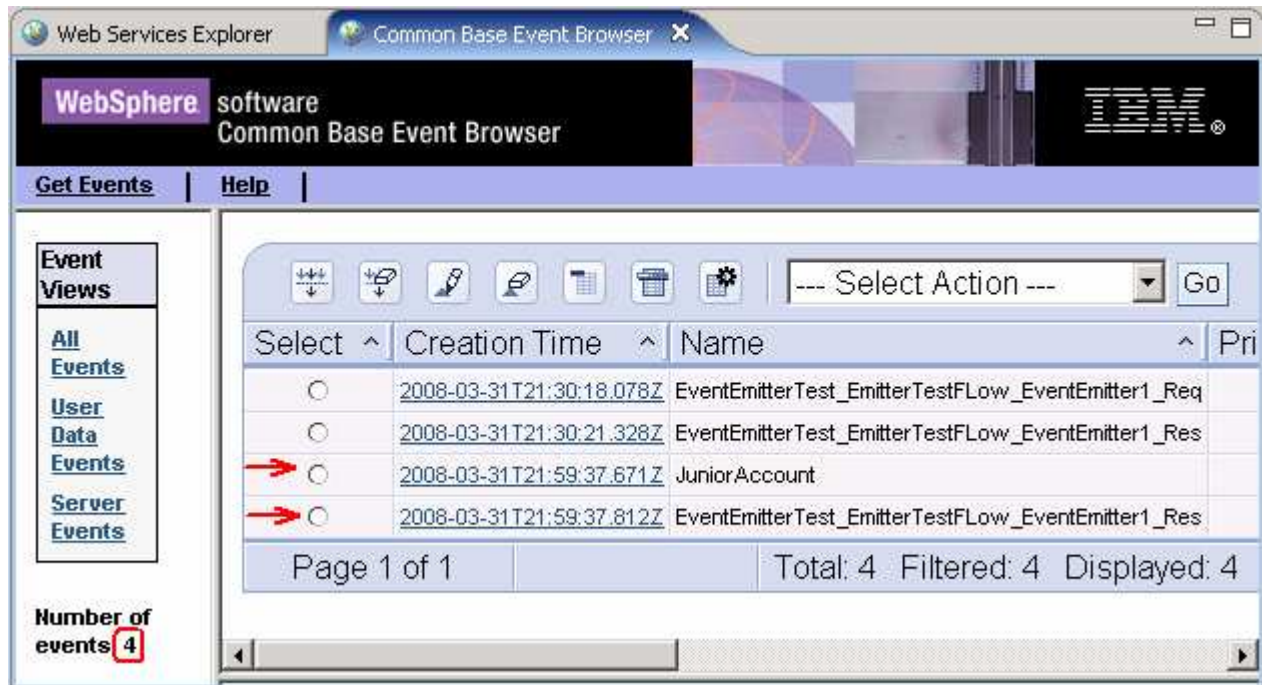
Address

- ___ d. houseNumber : **55**

___ e. street : **Long Lane**

___ f. postcode : **ABC23**

- ___ 16. Click on the **Go** button to generate the Events
- ___ 17. Follow the steps 11 and 12 of this section to view the events generated. There must be two more events added to the number of events listed as result of Test A in the common base event browser; one for **EventEmitter2** on the request flow and the other for **EventEmitter1** on the response flow as shown here:



- ___ 18. To view the contents of the Request Flow **EventEmitter2**, click on the Creation Time link or select the radio button next to **JuniorAccount**. The common base event browser will display the events in the Event Data pane as shown below:


```

<wbi:event>
  <wbi:eventHeaderData>
    <wbi:WBISESSION_ID> 9.3.75.99;EventEmitterTest;;getCustomerID;1207000777671;792604478</wbi:WBISESSION_ID>
    <wbi:ECSCurrentID> 9.3.75.99;EventEmitterTest;null;;getCustomerID;1207000777671;792604478</wbi:ECSCurrentID>
    <wbi:ECSParentID> 9.3.75.99;EventEmitterTest;;getCustomerID;1207000777671;792604478</wbi:ECSParentID>
    <wbi:WBIEventVersion> 6.1</wbi:WBIEventVersion>
  </wbi:eventHeaderData>
  <wbi:eventPointData xsi:type="esb:WESB_ECSEventMediation_CUSTOM" >
    <wbi:eventNature> CUSTOM</wbi:eventNature>
    <wbi:payloadType> full</wbi:payloadType>
    <esb:ModuleName> EventEmitterTest</esb:ModuleName>
    <esb:MediationName> EventEmitter2</esb:MediationName>
    <esb:EventEmitterLabel> JuniorAccount</esb:EventEmitterLabel>
    <esb:Root> /body</esb:Root>
  </wbi:eventPointData>
  <wbi:applicationData>
    <wbi:content wbi:name="Message" wbi:businessObjectName="getCustomerIDRequestMsg"
    wbi:targetNamespace="http://CustomerLibrary/CustomerDetails" >
      <wbi:value xsi:type="de:getCustomerIDRequestMsg" >
        <p:getCustomerID>
          <input1>
            <f_name> Harry</f_name>
            <l_name> Brown</l_name>
            <age> 13</age>
            <address>
              <houseNumber> 55</houseNumber>
              <street> Long Lane</street>
              <postcode> ABC23</postcode>
            </address>
          </input1>
        </p:getCustomerID>
      </wbi:value>
    </wbi:content>
  </wbi:applicationData>
</wbi:event>

```

Close

___ 19. Look for the following wbi elements in the event to ensure the correct data has been generated:-

- ___ a. **JuniorAccount**
- ___ b. ModuleName → **EventEmitterTest**
- ___ c. MediationName → **EventEmitter2**
- ___ d. Root → **/body**
- ___ e. f_name : **Harry**
- ___ f. l_name : **Brown**
- ___ g. age : **13**
- ___ h. houseNumber : **55**
- ___ i. street : **Long Lane**
- ___ j. postcode : **ABC23**

- ___ 20. To view the contents of the Response Flow **EventEmitter1**, click on the **latest** Creation Time link or select the radio button next to **EventEmitterTest_EmitterTestFlow_EventEmitter1_Res**. The common base event browser will display the events in the Event Data pane as shown below:

wbi:event

Show the details for the Common Base Event path wbi:event

expand

collapse

```
<wbi:event>
  <wbi:eventHeaderData>
    <wbi:WBISESSION_ID> 9.3.75.99;EventEmitterTest;;getCustomerID;1207000777671;792604478</wbi:WBISESSION_ID>
    <wbi:ECSCurrentID> 9.3.75.99;EventEmitterTest;null;;getCustomerID;1207000777671;792604478</wbi:ECSCurrentID>
    <wbi:ECSParentID> 9.3.75.99;EventEmitterTest;;getCustomerID;1207000777671;792604478</wbi:ECSParentID>
    <wbi:WBIEventVersion> 6.1</wbi:WBIEventVersion>
  </wbi:eventHeaderData>
  <wbi:eventPointData xsi:type="esb:WESB_ESBEventMediation_CUSTOM" >
    <wbi:eventNature> CUSTOM</wbi:eventNature>
    <wbi:payloadType> full</wbi:payloadType>
    <esb:ModuleName> EventEmitterTest</esb:ModuleName>
    <esb:MediationName> EventEmitter1</esb:MediationName>
    <esb:EventEmitterLabel> EventEmitterTest_EmitterTestFlow_EventEmitter1_Res</esb:EventEmitterLabel>
    <esb:Root> /body/getCustomerIDResponse/customerID</esb:Root>
  </wbi:eventPointData>
  <wbi:applicationData>
    <wbi:content wbi:name="Message" >
      <wbi:value xsi:type="xsd:string" > ABCDE12345</wbi:value>
    </wbi:content>
  </wbi:applicationData>
</wbi:event>
```

Close

- ___ 21. Look for the following elements in the event to ensure the correct data has been generated:-
- ___ a. **EventEmitterTest_EmitterTestFlow_EventEmitter1_Res**
 - ___ b. ModuleName → **EventEmitterTest**
 - ___ c. MediationName → **EventEmitter1**
 - ___ d. Root → **/body/getCustomerIDResponse/customerID**
 - ___ e. Message → **ABCDE12345**
- ___ 22. The Event Generation Test B is complete
- ___ 23. Close the Web Services Explorer

Part 5: Event generation test C using the test client


In this section of the lab, a Test C is performed to produce events at the CEI Server using a Test Client. Note that the Test A and Test B which were done using a WebServices Explorer are combined to perform a Test C using a Test Client rather than a WebServices Explorer.

Note: To avoid confusion, delete all the events generated as result of the actions during the course of Part 4. To achieve this:

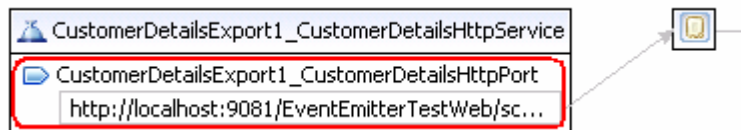
→ Open a command line window and change directory to <ESB_PROFILE_HOME>\bin>

→Run the following command to delete all the events:

```
eventpurge.bat -group "All events" -seconds 0
```

- ____ 1. In WebSphere Integration Developer work bench, switch to the Business Integration Perspective. To switch to this perspective: **Window** → **Open Perspective** → **Other...** and select **Business Integration** from the **Select Perspective** window and click **OK**
- ____ 2. In the Business Integration view, expand the **EventEmitterTest** mediation module and double-click the mediation module assembly ( Assembly Diagram) to open it with the assembly editor
- ____ 3. Right-click anywhere on the white space on the assembly diagram canvas and select **Test Module** from the context menu
- ____ 4. The Test client for **EventEmitterTest** mediation module is opened

Note: Ensure that the 'Web Service Port' is correct with respect to the environment you are working. In the business integration view, expand 'CustomerLibrary → Web Service Ports' and double click 'CustomerDetailsExport1_CustomerDetailsHttpPort' to open it in an interface editor. Update the Web Service URL to meet your requirements and save the changes. Re-publish the mediation module.



Events



General Properties

Detailed Properties

Configuration: Default Module Test

Module: EventEmitterTest

Component: CustomerDetailsExport1

Interface: CustomerDetails

Operation: getCustomerID

Invoke export using binding

Initial request parameters

Name	Type	Value
anyAttribute	anySimpleType[]	
Body	Body	
any	anyType[]	
getCustomerID	getCustomerID	
input1	CustomerDetails	
f_name	string	
l_name	string	
age	int	0
address	Address	
houseNumber	string	
street	string	
postcode	string	

5. Enter the same values in the fields, used for **Event Generation Test A** of Part4, as shown below:

Events



General Properties

Detailed Properties

Configuration: Default Module Test

Module: EventEmitterTest

Component: CustomerDetailsExport1

Interface: CustomerDetails

Operation: getCustomerID

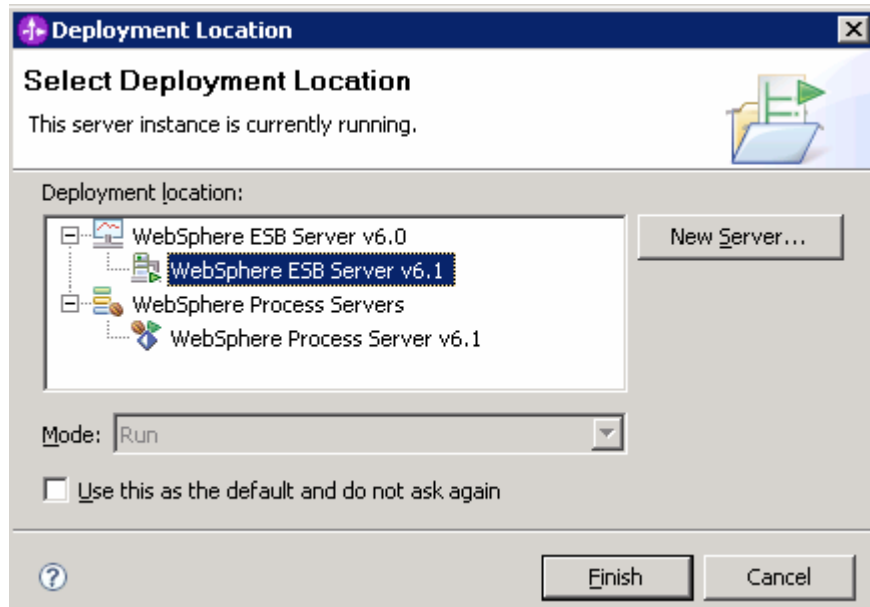
Invoke export using binding

Initial request parameters

Name	Type	Value
anyAttribute	anySimpleType[]	✓
Body	Body	✓
any	anyType[]	68
getCustomerID	getCustomerID	✓
input1	CustomerDetails	✓
f_name	string	✓ Fred
l_name	string	✓ Smith
age	int	✓ 55
address	Address	✓
houseNumber	string	✓ 1
street	string	✓ High Street
postcode	string	✓ 512JN

___ a. Click **Continue** ()

___ b. Select WebSphere ESB Server 6.1 as the deployment location from the pop-up window when prompted



___ c. Click **Finish**

___ 6. Ensure the following is resulted on a successful execution:

Events

General Properties

Detailed Properties

Module: [EventEmitterTest](#)
 Component: [CustomerDetailsExport1](#)
 Interface: [CustomerDetails](#)
 Operation: [getCustomerID](#)

Return parameters:

Name	Type	Value
customerID	string	✓ ABCDE12345

___ 7. Now follow the **Steps 11 to 17 of Part4** to view the events generated and the test is successful

___ 8. Close the Test Client and say no to save changes when prompted


___ 9. Run the test again and enter the same values in the fields, used for **Event Generation Test B** of Part4

___ a. In the Business Integration view's tree, expand the **EventEmitterTest** mediation module and double-click the mediation module assembly (Assembly Diagram) to open it with the assembly editor

___ b. Right-click any where on the canvas and select **Test Module** from the context menu

___ c. Enter the same values in the fields as used for Event Generation Test B of Part4, as shown below:

Events



General Properties

Detailed Properties

Configuration: Default Module Test

Module: EventEmitterTest

Component: CustomerDetailsExport1


Interface: CustomerDetails

Operation: getCustomerID

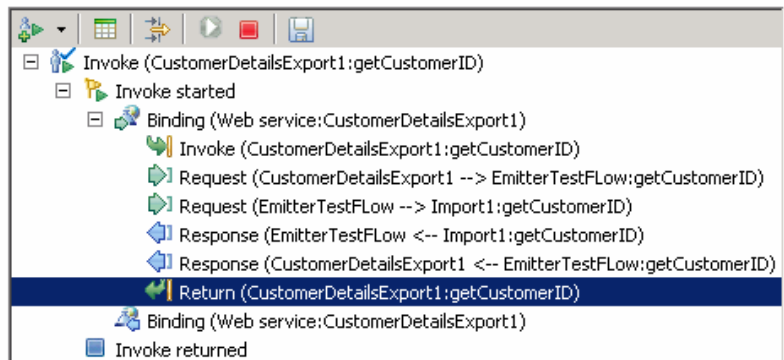
Invoke export using binding

Initial request parameters

Name	Type	Value
[] anyAttribute	anySimpleType[]	✓
Body	Body	✓
[] any	anyType[]	68
[] getCustomerID	getCustomerID	✓
[] input1	CustomerDetails	✓
[] f_name	string	✓ Harry
[] l_name	string	✓ Brown
[] age	int	✓ 13
[] address	Address	✓
[] houseNumber	string	✓ 55
[] street	string	✓ Long Lane
[] postcode	string	✓ ABC23
[] anyAttribute	anySimpleType[]	✓

- ___ d. Click **Continue** ()
- ___ e. Select WebSphere ESB Server 6.1 as the deployment location from the pop-up window when prompted
- ___ f. Click **Finish**
- ___ g. Ensure the following is resulted on a successful execution:

Events



General Properties

Detailed Properties

Module: [EventEmitterTest](#)

Component: [CustomerDetailsExport1](#)

Interface: [CustomerDetails](#)

Operation: [getCustomerID](#)

Return parameters:

Name	Type	Value
[] customerID	string	✓ ABCDE12345

___ 10. Now follow the **Steps 18 to 25 of Part4** to view the events generated and the test is successful

- ____ 11. Close the Test Client and say no to save changes when prompted
- ____ 12. The Event Generation Test C using the Test Client is complete

Part 6: Save the work and clean up server

- ___ 1. Export project as Project Interchange file
 - ___ a. In WebSphere Integration Developer, Navigate to **File → Export**.
 - ___ b. Select **Project Interchange**.
 - ___ c. Out of all the projects listed, select only the following projects:
 - CustomerIDModule
 - CustomerLibrary
 - EventEmitterTest
 - ___ d. Save in C:/LabFiles61/WESB/EventEmitterPrimitive/
 - ___ e. Name the project interchange **WESB_EventEmitterPrimitive_Solution_PI.zip**
- ___ 2. Remove all the projects and **clean** up the ESB Server.
 - ___ a. Right-click on WebSphere ESB Server v6.1 (once started) and select **Add and Remove Projects...**
 - ___ b. Select **Remove-All** and click **Finish**
 - ___ c. After the projects are removed, **stop** the WebSphere ESB Server v6.1.

What you did in this exercise

In this lab, you created a Mediation Module and a Mediation Flow. You created the Request and Response Flows for the Mediation Flow. Further you visually composed the Mediation Module in the Assembly Editor. Finally you tested the Mediation Module using a Web Services Explorer and a Test Client.

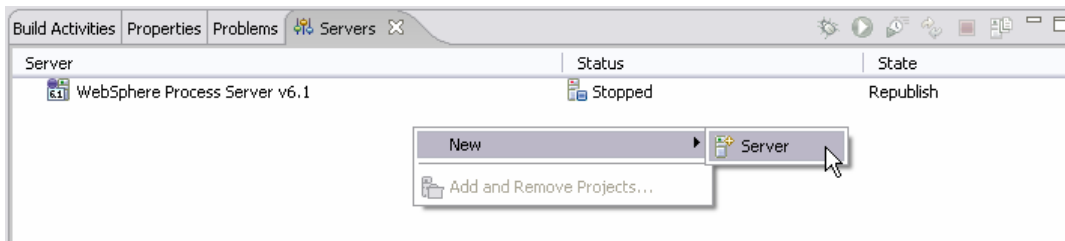
Solution instructions

- ___ 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:/LabFiles61/WESB/EventEmitterPrimitive/WESB_EventEmitterPrimitive_Solution_Pi.zip**
 - ___ c. Click **Finish** button
- ___ 2. Continue with **Part 3, Part 4 & Part 5** of this LAB

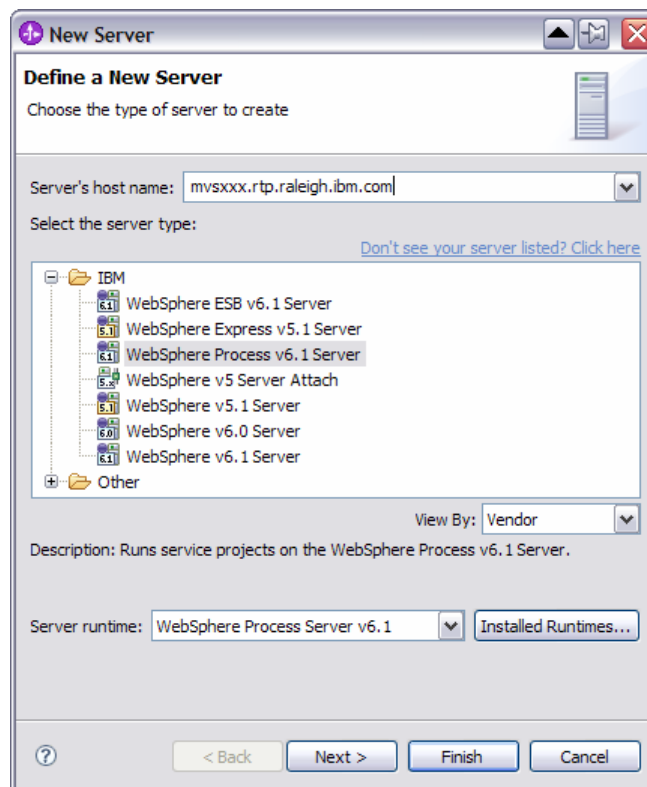
Task: Adding remote server to WebSphere Integration Developer test environment

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**.

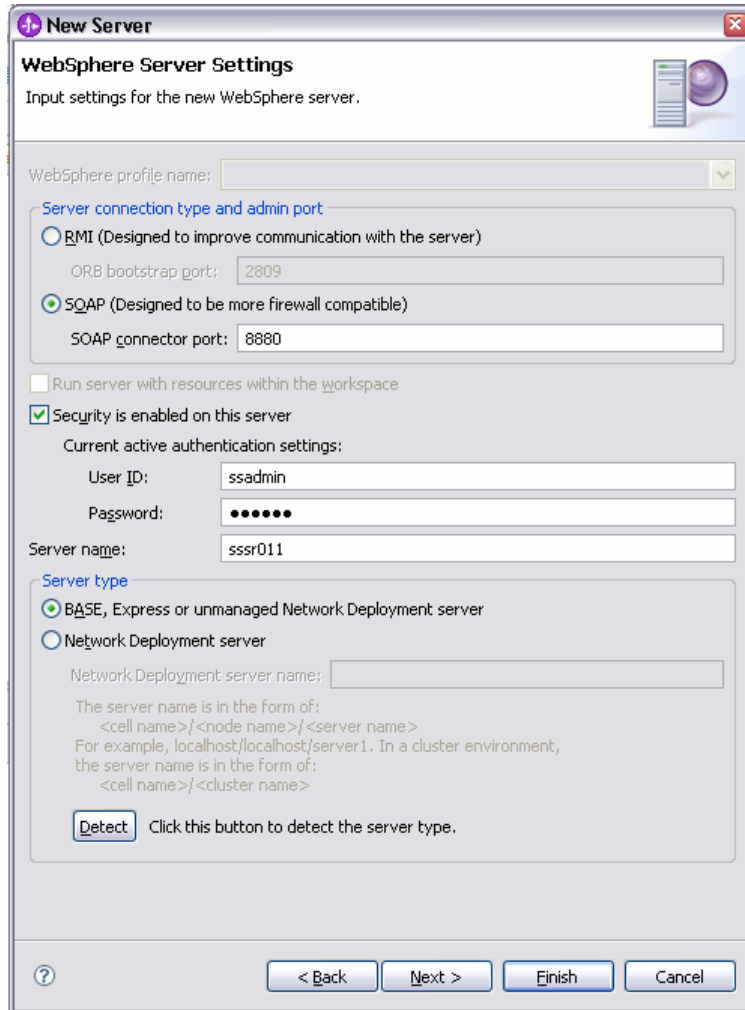


- ___ c. In the New Server dialog, specify the remote server's host name, **<HOSTNAME>**.
- ___ d. Ensure that the appropriate server type, **'WebSphere Process v6.1 Server'** or **'WebSphere ESB v6.1 Server'**, is highlighted in the server type list

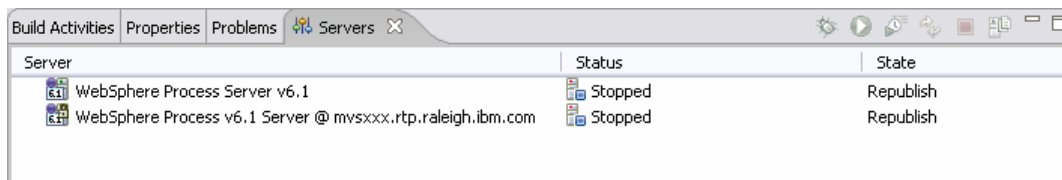


- ___ e. Click **Next**.

- ___ 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 userid 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>'

userid: <USERID>

pwd: <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
```