IBM WebSphere® Enterprise Service Bus V6.0.2 – Lab exercise

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 5

Part 1: Prepare environment for the lab 6

Part 2: Create mediation module and flow 9

Part 3: Deploy the modules and configure Web services explorer 26

Part 4: Event generation test A and B using the Web services explorer 34

Part 5: Event generation test C using the test client 44

Part 6: Save the work and clean up server 50

What you did in this exercise 51

Solution instructions 52

Task: Adding remote server to WebSphere Integration Developer test environment 53

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 V6.0.2 with the WebSphere ESB test server option installed

What you should be able to do

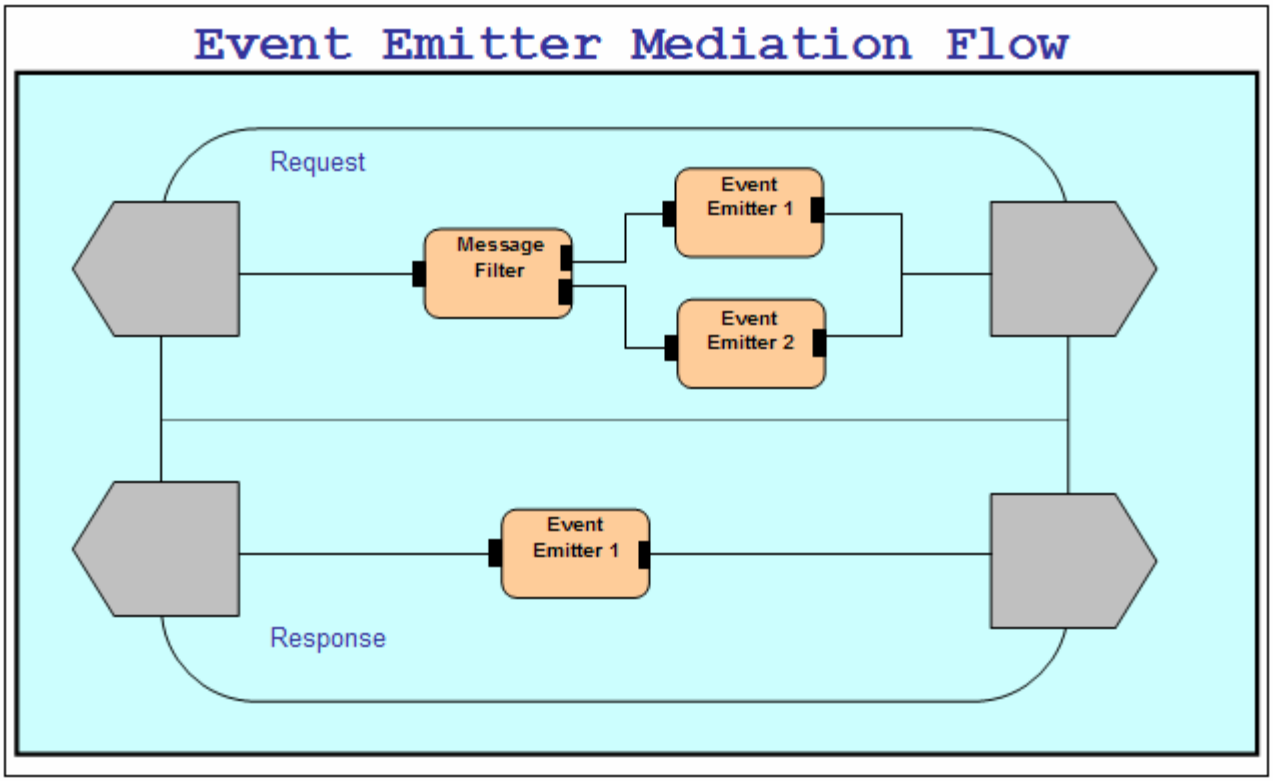
At the end of this lab you should be able to:

- Import the project interchange file into the WebSphere Integration Developer V6.0.2 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 6.0 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 will be run in the request flow; otherwise event emitter 2 will be 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

Note: In this lab, the mediation flow contains only event emitter primitives and one message filter primitive but in true customer scenarios it is expected that the event emitter will be used in combination with other primitive types. However, as each mediation primitive is stateless, the addition of other primitives to this scenario would not add any additional test execution paths for the event emitter line item.

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
<WID_HOME>	C:\WID602	
<LAB_FILES>	C:\Labfiles602	/tmp/Labfiles602

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

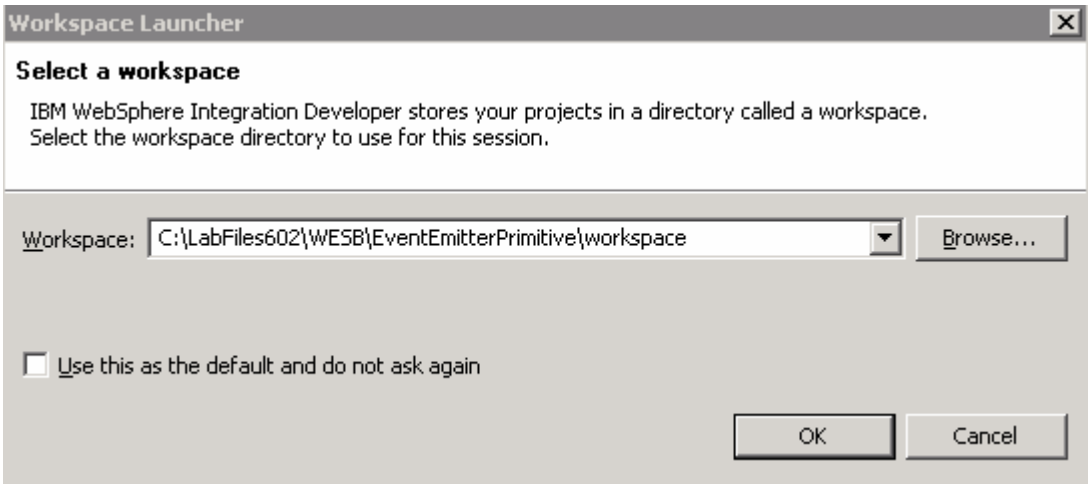
Reference Variable	Example: Remote Windows test server location	Example: Remote z/OS test server location	Input your values for the remote location of the test server
<SERVER_NAME>	server1	cl1sr01	
<WAS_HOME>	C:\Program Files\IBM\WebSphere\App Server	/etc/cl1cell/AppServerNode1	
<HOSTNAME>	Localhost	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 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 V6.0.2 with a workspace location of **C:\LabFiles602\WESB\EventEmitterPrimitive\workspace**

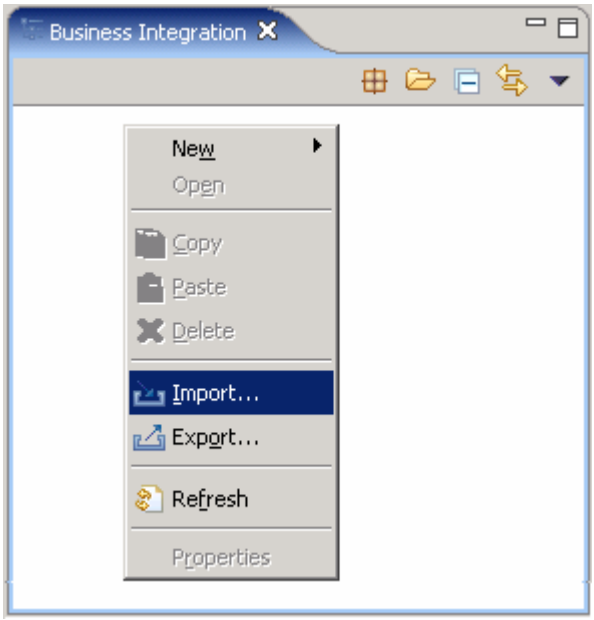


____ 2. On the welcome screen, click the curved arrow at the top right to "**Go to the business integration perspective** (), to close the Welcome window

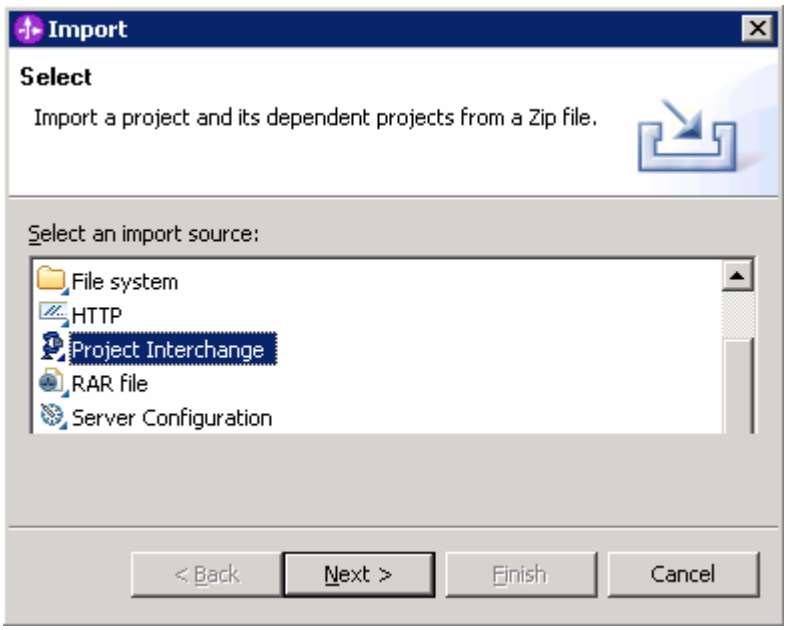
____ 3. Import the Project Interchange file, **WESB_EventEmitterPrimitive_PI.zip**, into the development environment

__ a. Right-click inside **Business Integration View** (top left view in the Business Integration Perspective)

__ b. Select **Import** from the context menu

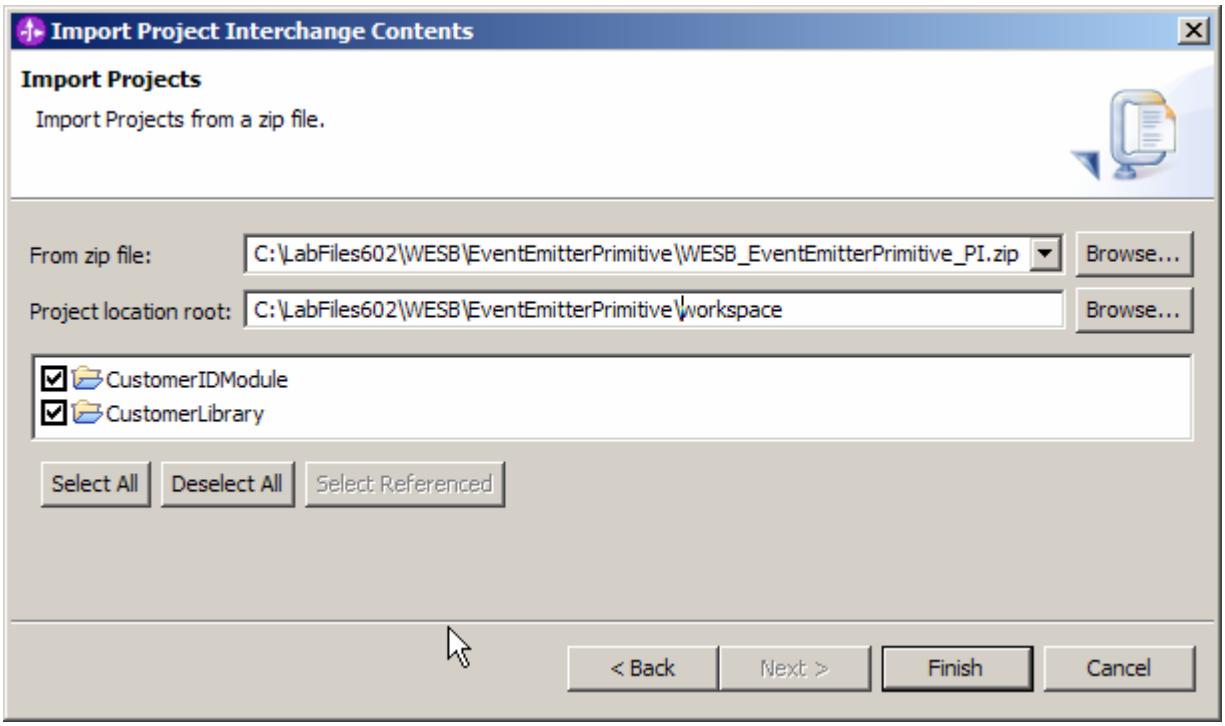


__ c. From the **Import** dialog, select **Project Interchange** from the list.



__ d. Click **Next**

__ e. Click the **Browse** button for "**From zip file"** to navigate for the Project interchange file, **WESB_EventEmitterPrimitive_PI.zip**

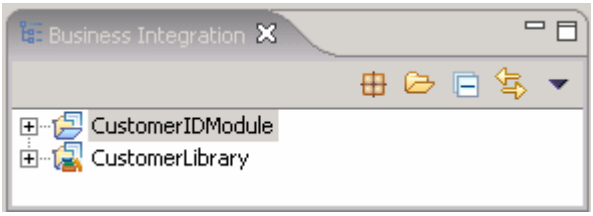


__ f. Click the **Select All** button to ensure all projects listed are selected

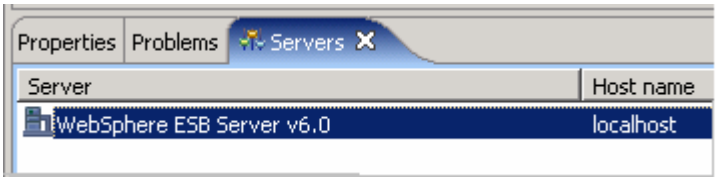
- Projects listed are : **CustomerIDModule** and **CustomerLibrary**

__ g. Click the **Finish** button (projects will be imported and auto-build will run)

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



____ 4. Verify that the WebSphere ESB Server v6.0 is listed in the **Servers** view



Part 2: Create mediation module and flow

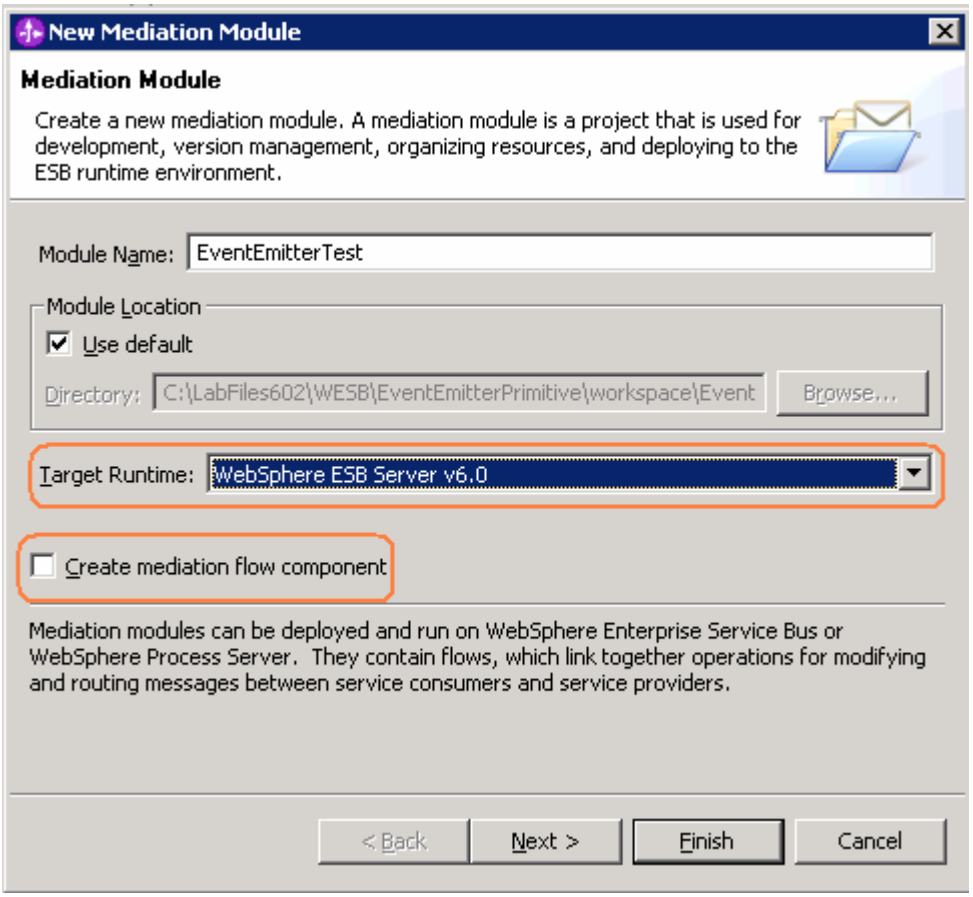
In this section of the lab, a new mediation module and a Mediation Flow are created. The request and response flows are created for the Mediation flow. Further the Mediation Module is visually composed using the assembly editor.

____ 1. To create the mediation module, complete the following steps:

__ a. In the Business Integration view, right-click to see the context menu and select **New > Mediation Module**. The new Mediation Module window opens

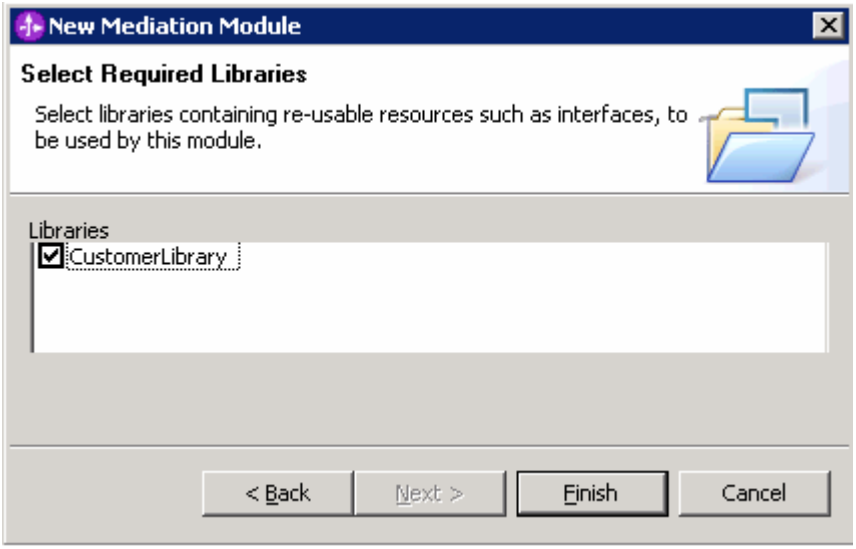
__ b. In the New Mediation Module window, type the **Module Name** as **EventEmitterTest**

__ c. Verify that the target runtime is the **WebSphere ESB Server v6.0** and **clear** the "**Create mediation flow component**" check box



__ d. Click **Next**

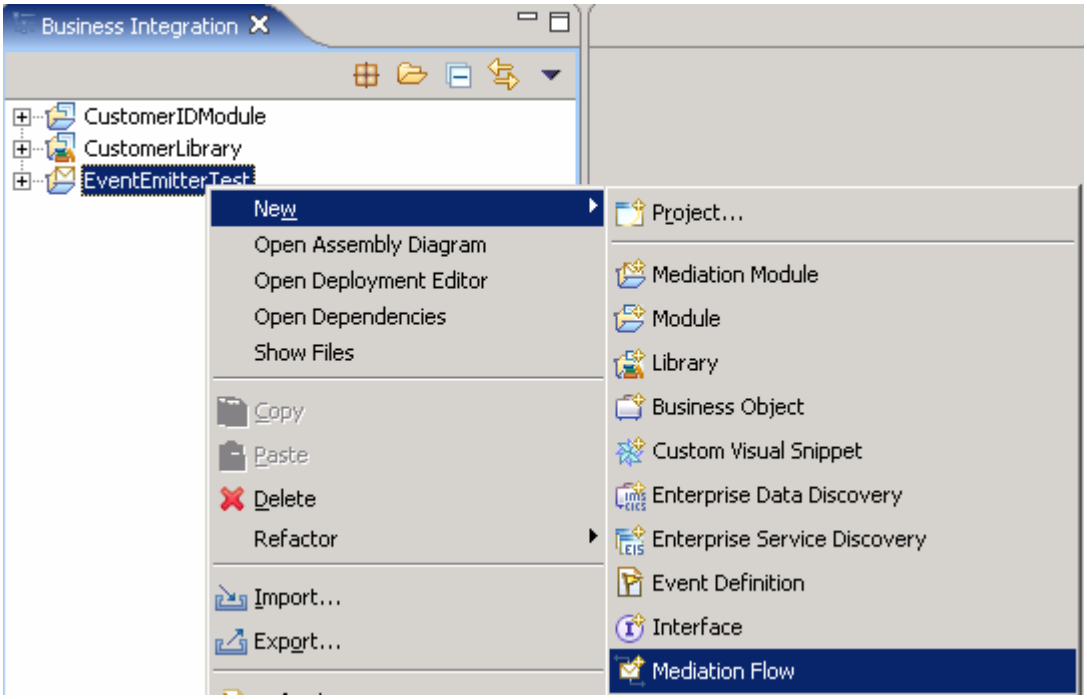
__ e. In the Select Required Libraries wizard, select **CustomerLibrary**



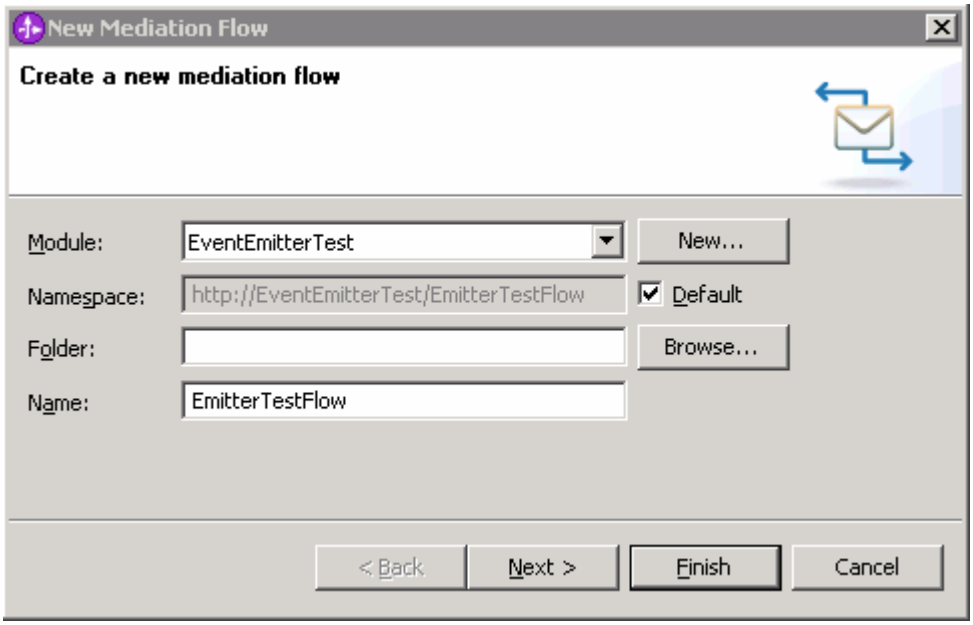
__ f. Click **Finish.** A mediation module called **EventEmitterTest** is created

____ 2. Create a Mediation Flow by completing the following steps:

__ a. In the Business Integration View, right-click on **EventEmitterTest** Mediation Module to see the context menu and select **New → Mediation Flow**



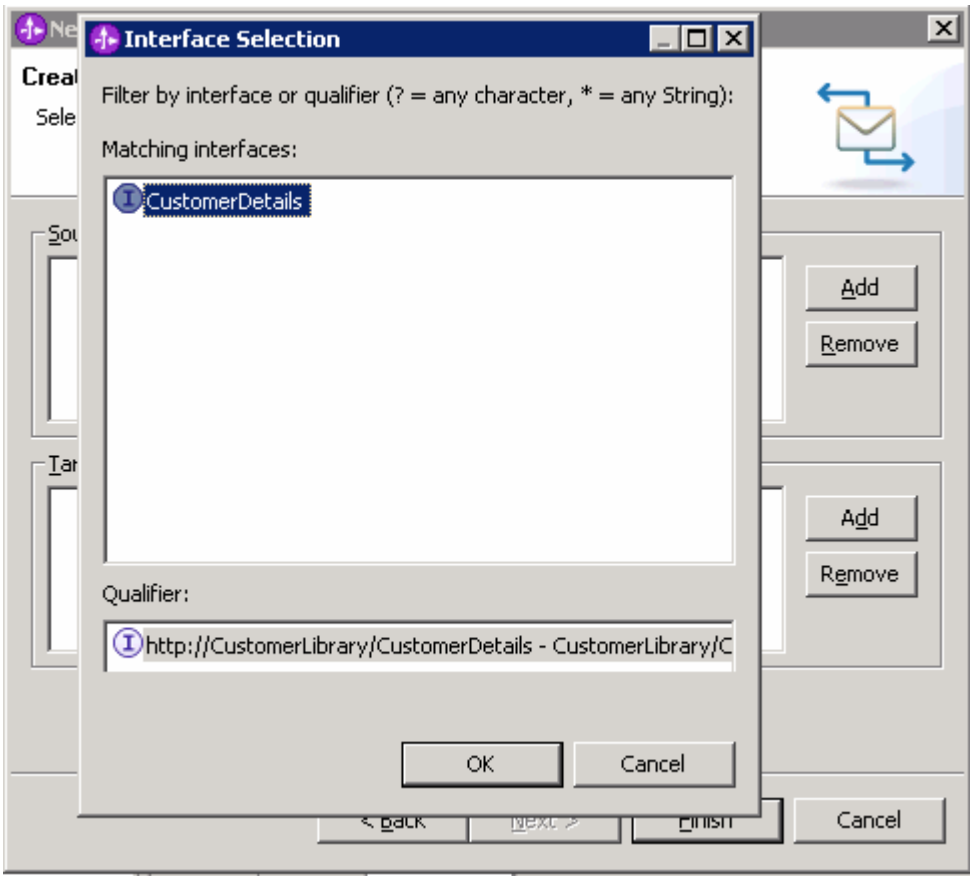
__ b. In the New Mediation Flow window, enter the **Name** as **EmitterTestFlow**



__ c. Click **Next**

__ d. In the following window (creating a new interface), 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

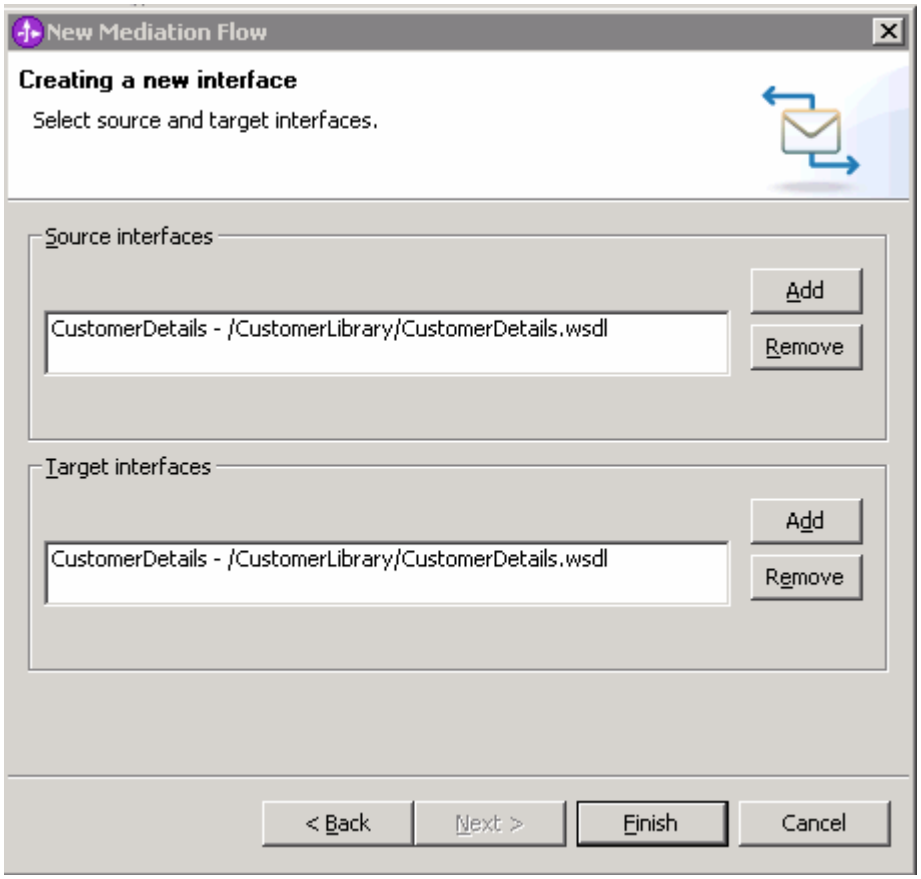


2) Click **OK** over the **Interface Selection** pop-up window

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

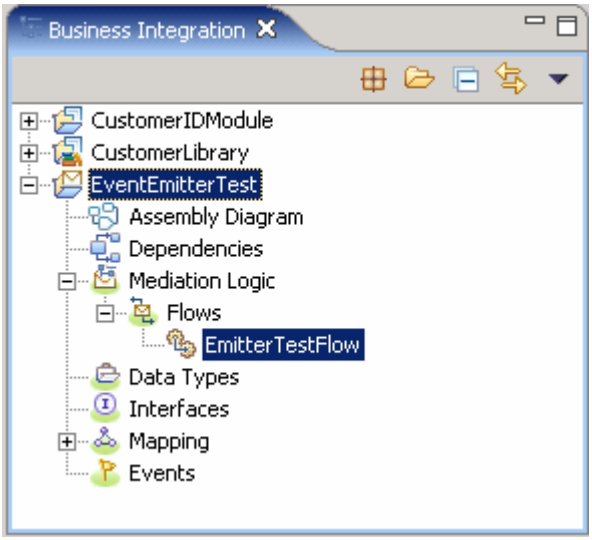
4) Click **OK** over the **Interface Selection** pop-up window

5) The added source and target interfaces panel must like the one below:

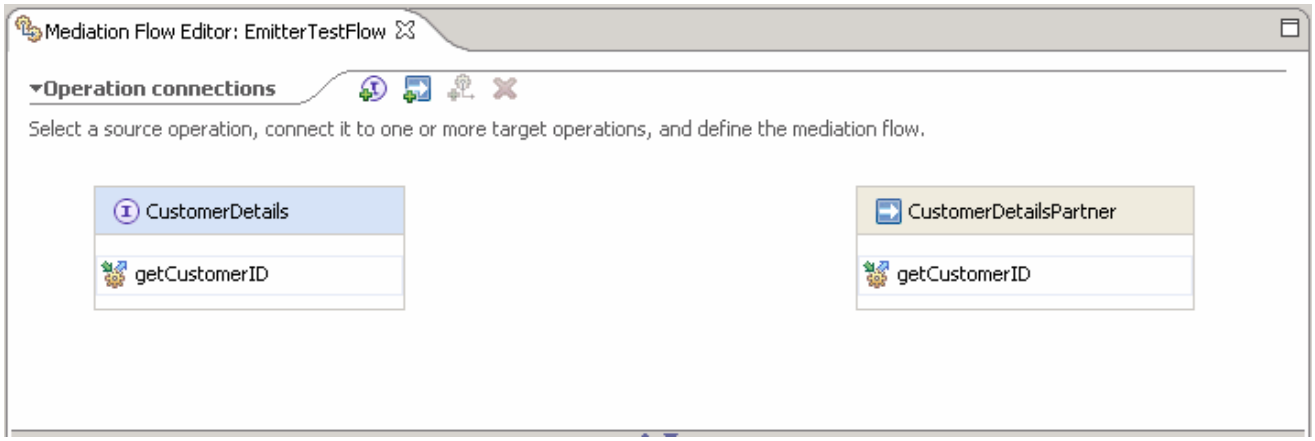


6) Click **Finish**

__ e. A new Mediation Flow, that is, **EmitterTestFlow** is created for the **EventEmitterTest** Mediation Module as shown below:



__ f. Upon creation of the **EmitterTestFlow** Mediation Flow, it is opened in the Mediation Flow Editor as shown in the diagram below:

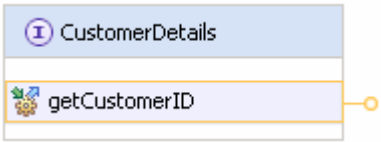


Create a request flow:

To create the request flow, complete the following 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-hand side of the Operation Connections view



__ b. Drag to the target operation, **CutomerDetailsPartner/getCustomerID** on the right-hand 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, **CutomerDetailsPartner/getCustomerID**

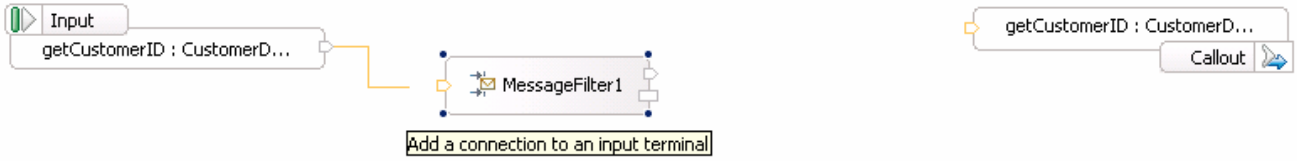
__ c. Click on the black line (wire) to view the Mediation Flow View and ensure the **Request** tab is selected to build the Request flow

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

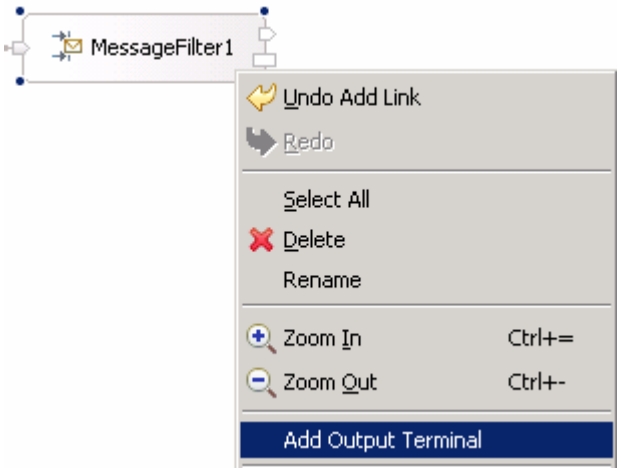
__ a. In the **Mediation Flow Editor**(middle window), click on the **Message Filter** icon ( ) to select the Message Filter primitive from the pallet tray on left-hand side of view and drop it into the canvas between the Input Request node and the Callout Request Node

Note: Click Down arrow of the pallet tray to scroll and locate the **Message Filter** icon if not visible.

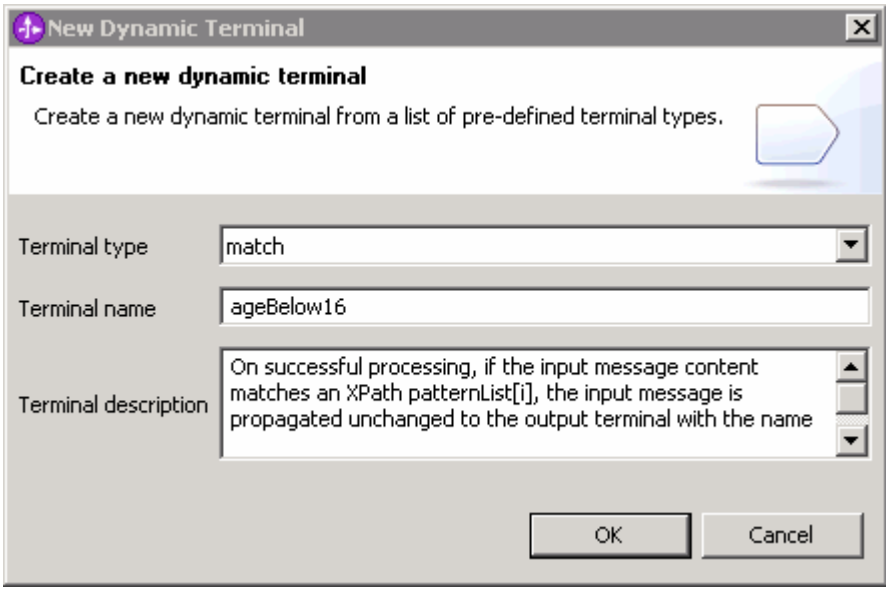
__ b. Hover the mouse over the **Input** node's output terminal and drag the handle that appears, to the input terminal of the Message Filter primitive, **MessageFilter1**



__ c. Right Click on the Message Filter Primitive, **MessageFilter1** and select **Add Output Terminal** from the context menu

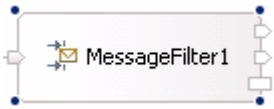


__ d. Enter **ageBelow16** for the **Terminal name** field

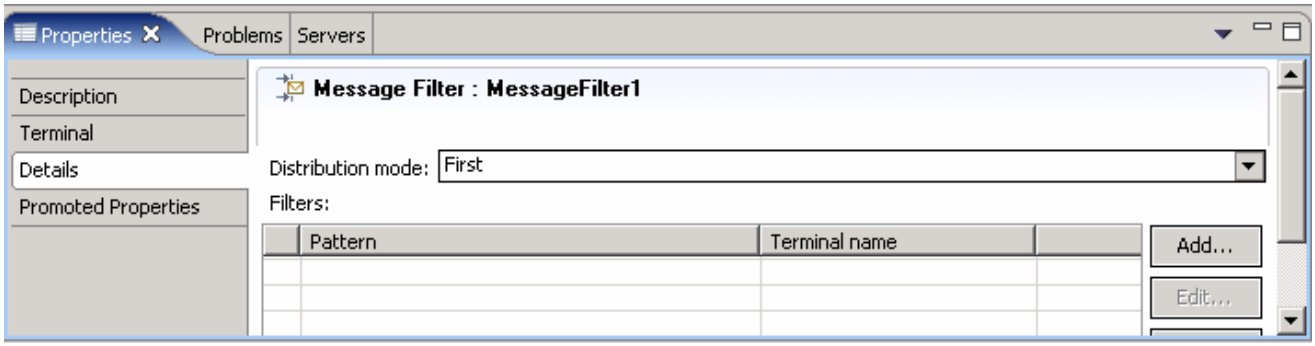


__ e. Click **OK**

__ f. The Message Filter primitive must look like the one below with a new Output terminal added:

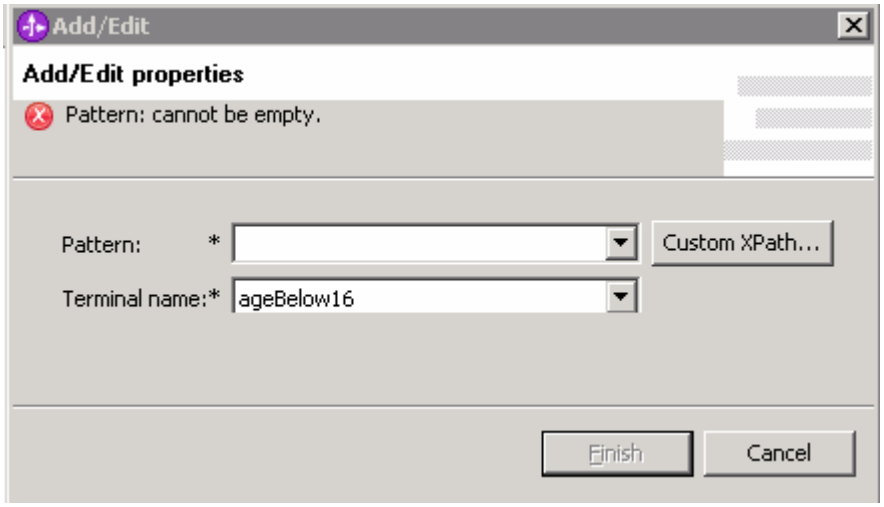


__ g. Select the Message Filter primitive, **MessageFilter1** and choose **Details** under its properties view



__ h. Click the **Add** button for **Filters**

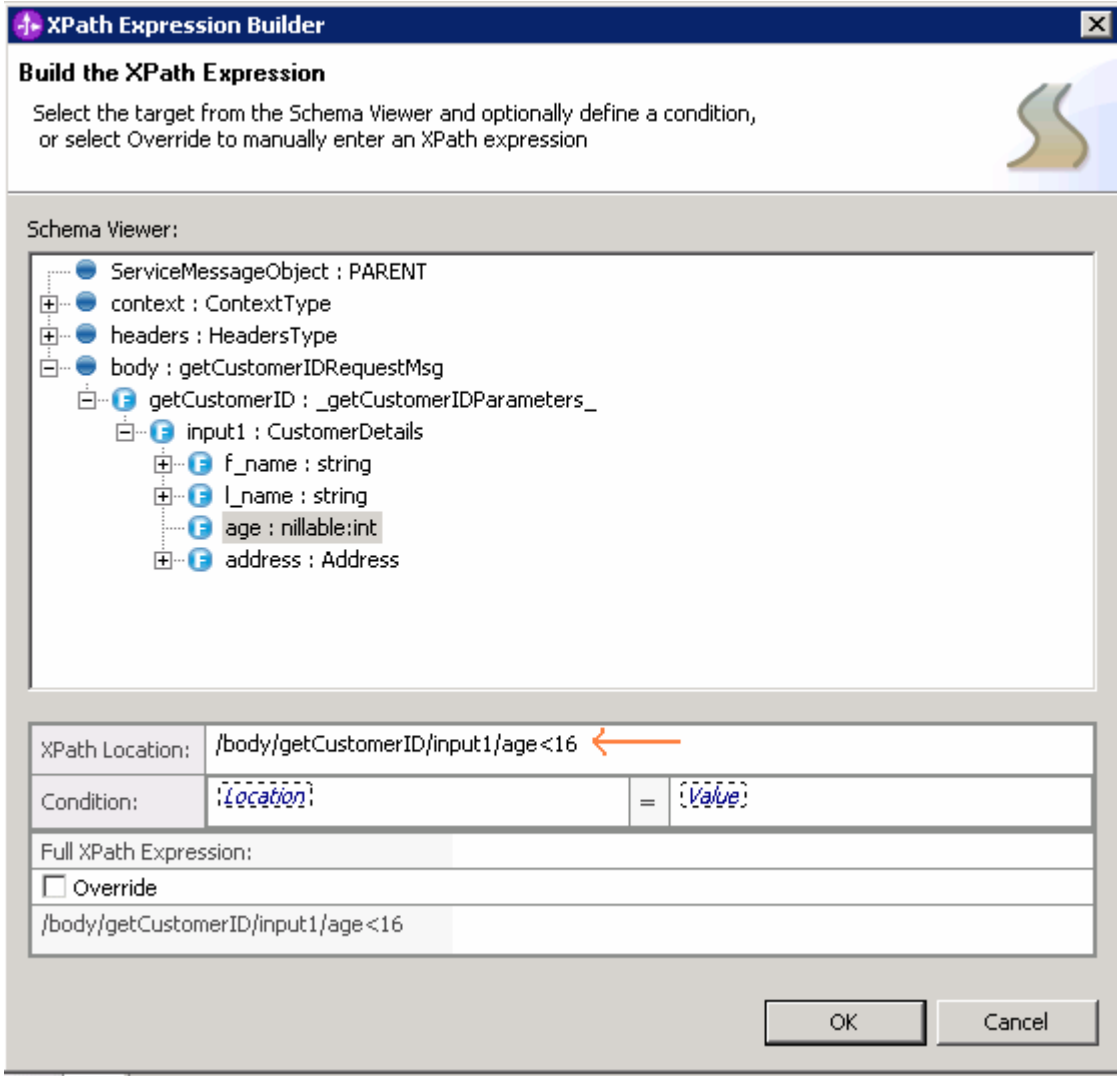
__ i. The **Add/Edit properties** window opens



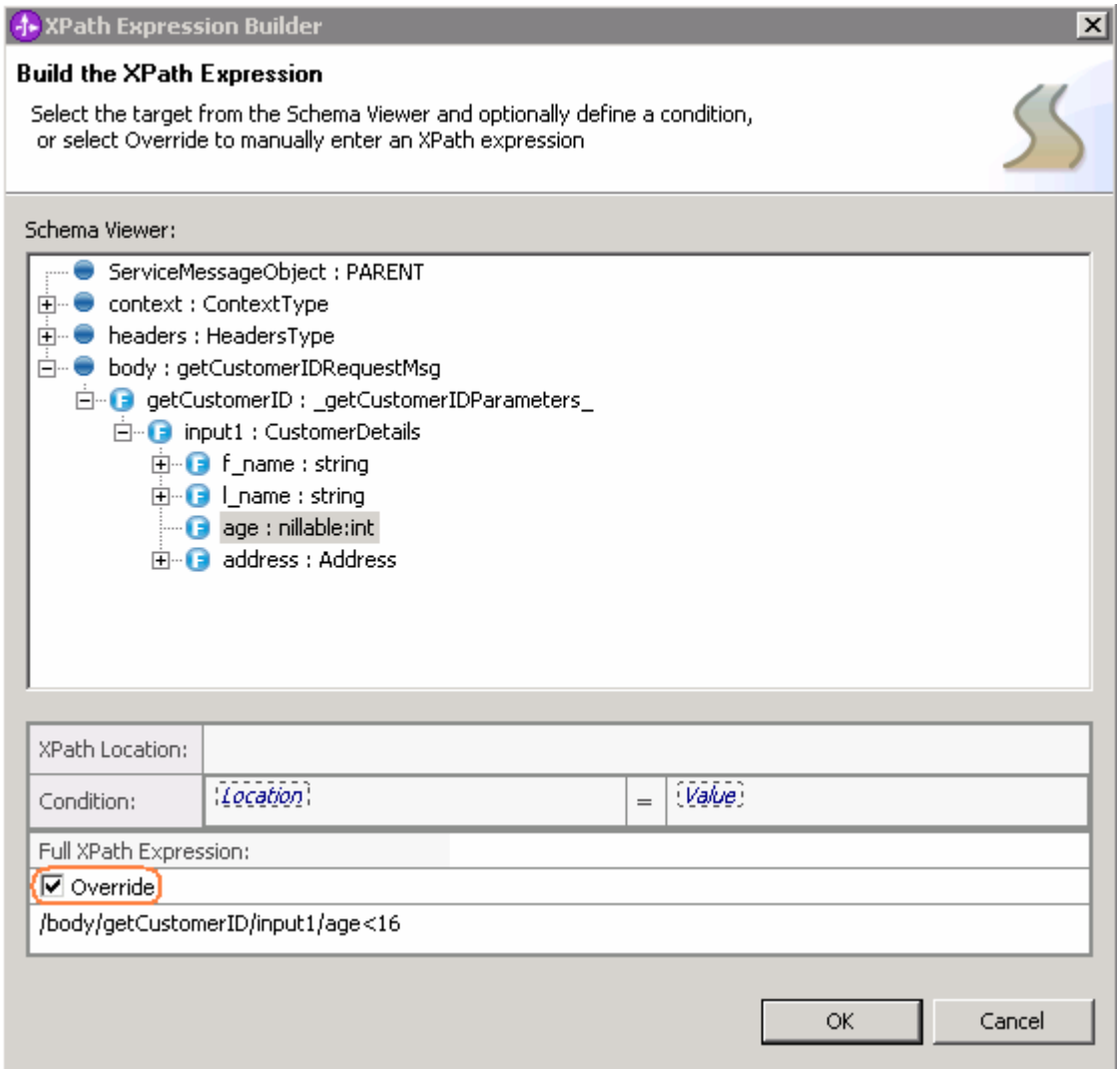
__ j. From the **Add/Edit properties** window, click the **Custom XPath** button to set the Pattern. The **XPath Expression Builder** window opens

__ k. In the **XPath Expression Builder** window, navigate to the **age** element (ServiceMessageObject> body>getCustomerID>input1>age) in the SMO body, and alter the XPath to **/body/getCustomerID/input1/age<16** as shown below:

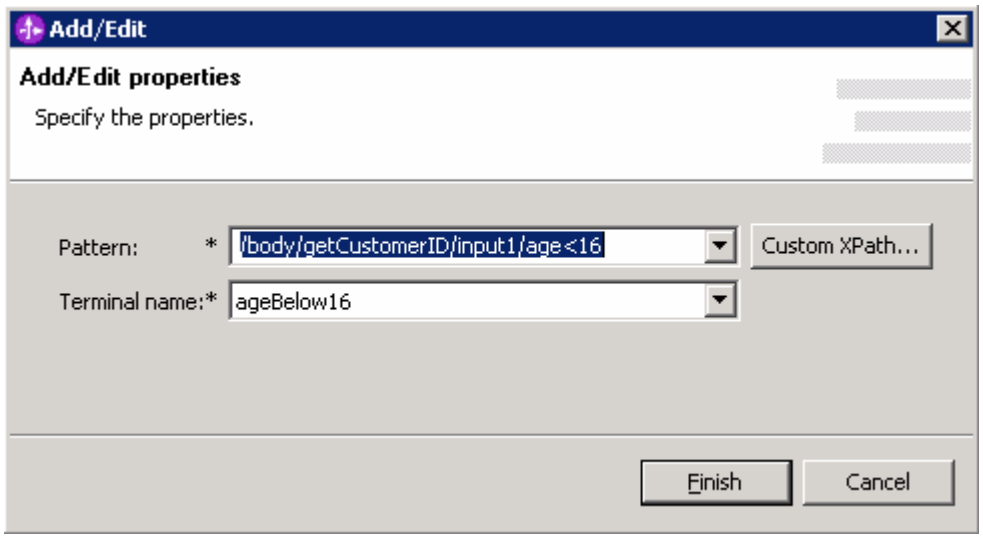
Note: To Alter the XPath location, type **<16** next to **/body/getCustomerID/input1/age** for the XPath Location field.



__ l. Now select the check box next to **Override** under the Full XPath Expression section as shown below:

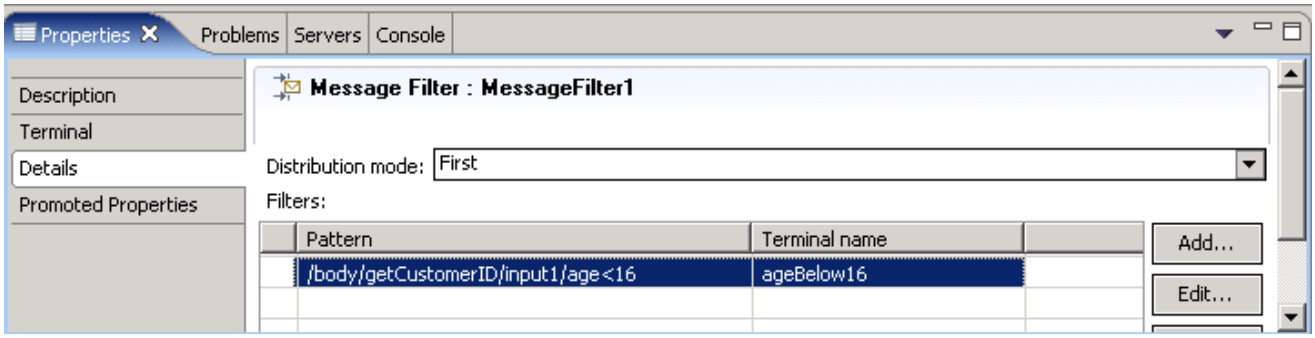


__ m. Click **OK**

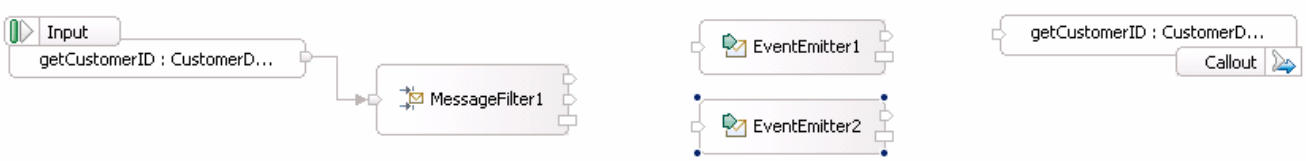


__ n. Click **Finish** over the **Add/Edit properties**

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

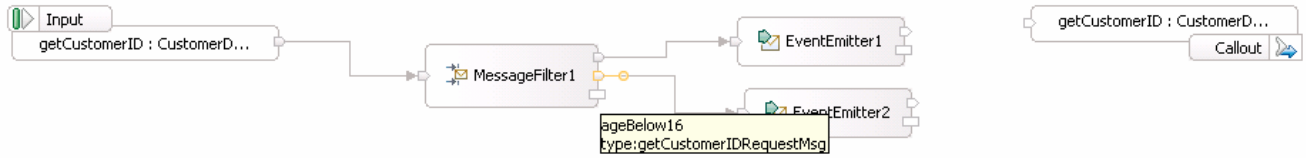


__ p. In the **Mediation Flow Editor**(middle window), click on the **Event Emitter** icon ( ) to select the Event Emitter primitive from the pallet on left-hand side of view and drop **two** of them into the canvas between the, **MessageFilter1** and the Callout Request Node as shown below:

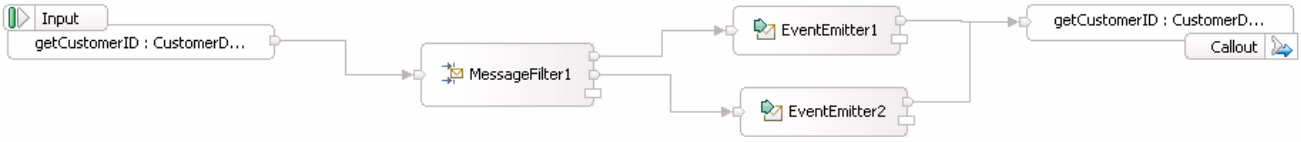


__ q. Wire the Message Filter primitive, MessageFilter1 and the Event Emitter primitives, EventEmitter1 and EventEmitter2 as shown below:

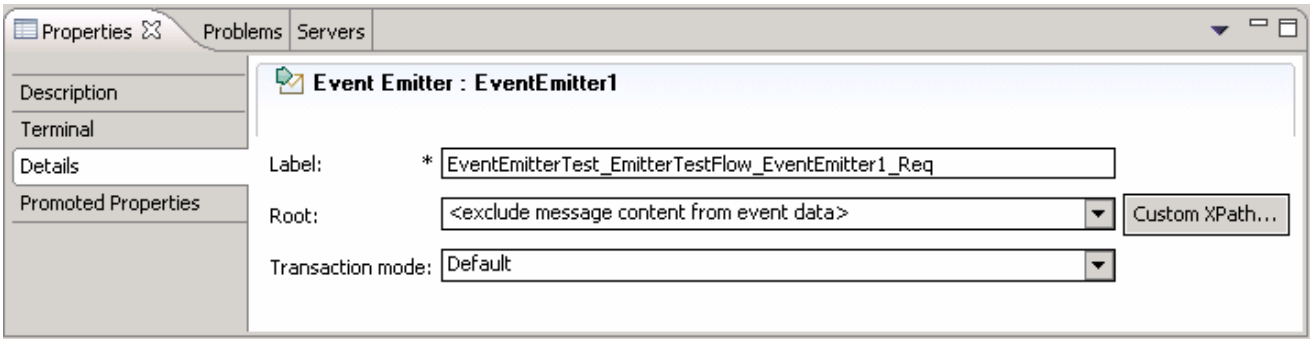
Note: The **ageBelow16** output terminal of the Message Filter is wired to the input terminal of **EventEmitter2** primitive. The **default** output terminal is wired to the input terminal of **EventEmitter1** primitive.



__ 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 choose **Details** under its properties view. The property values for this primitive are the default values and should match the diagram below:

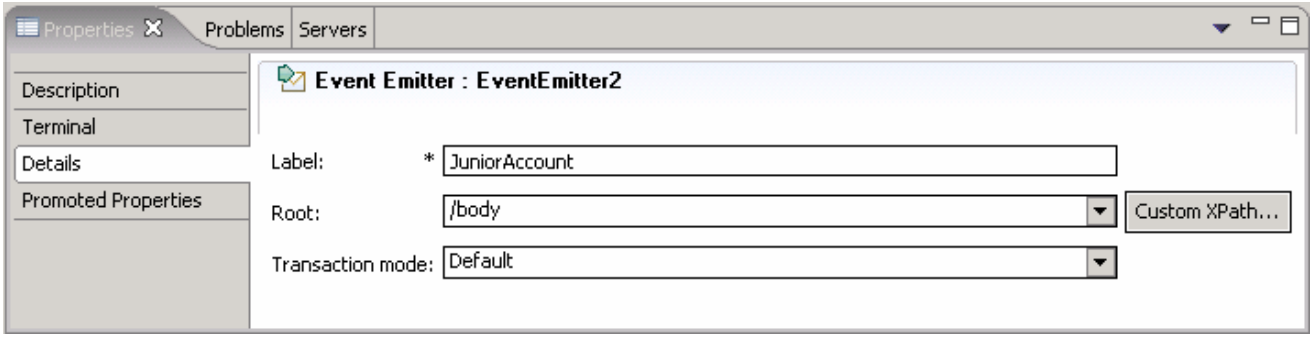


__ t. Select the Event Emitter primitive, **EventEmitter2** and choose **Details** under its properties view

1) Enter **JuniorAccount** for the event **Label**

2) Select **/body** from the **Root** drop down list

3) Leave the **Transaction mode** as default with the value, **Default**



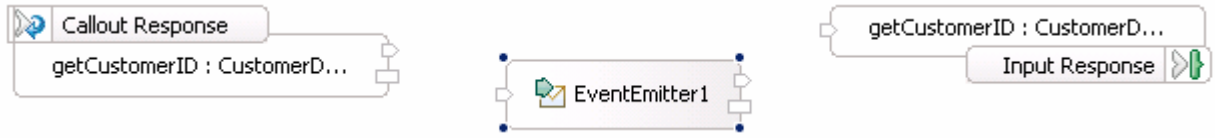
__ u. Save all work by choosing **File > Save All** or **Crtl + Shift + S**

Create a Response Flow:

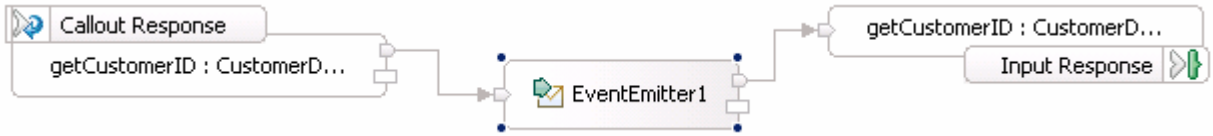
____ 5. Create the Request Flow by completing the following steps:

__ a. While still in Mediation Flow Editor(middle section), select the **Response tab**

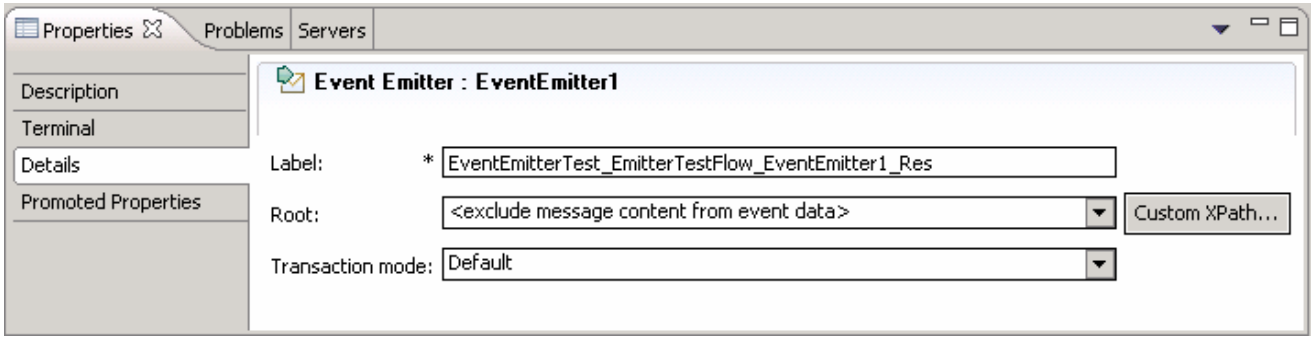
__ b. In the **Mediation Flow Editor**(middle), click on the **Event Emitter** icon ( ) to select the Event Emitter primitive from the pallet on left-hand side of view and drop into the canvas as shown below



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

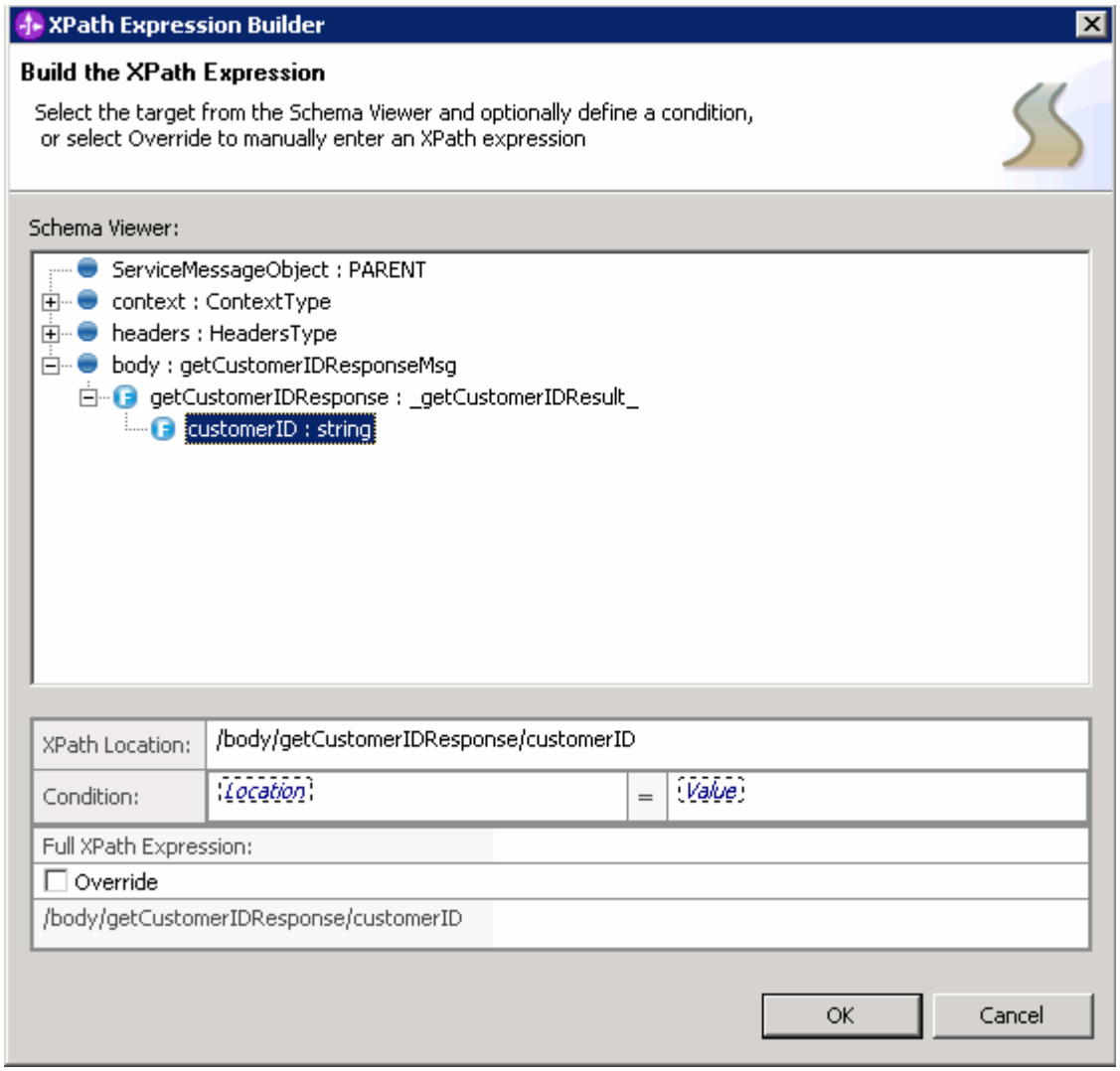


__ d. Select the Event Emitter primitive, **EventEmitter1** in the Response Flow and choose **Details** under its properties view



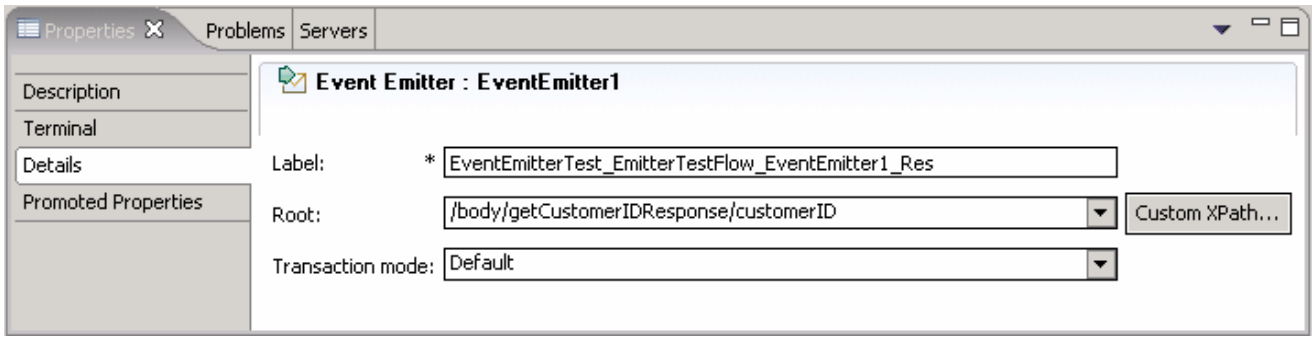
__ e. Click the **Custom XPath** button for the **Root** property. The **XPath Expression Builder** window opens

__ f. In the **XPath Expression Builder** window, navigate to the **customerID** element; select it within the SMO body



__ g. Click **OK**

__ h. Leave all the other properties as default values. The properties view for the **EventEmitter1** primitive of the Response Flow must look like the diagram below:



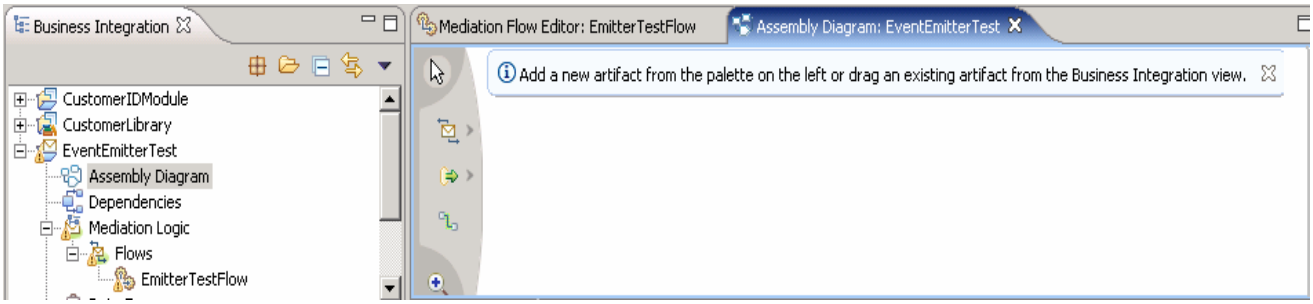
__ i. Save all work by choosing **File > Save All** or **Crtl + Shift + S**

Visually compose the Mediation Module:

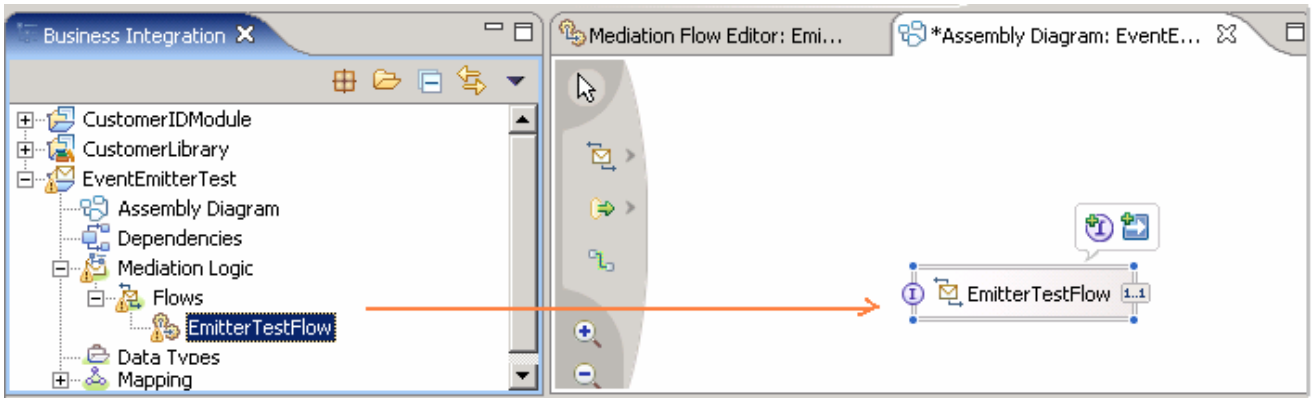
____ 6. To Visually compose the Mediation Module, complete the following steps:

__ 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

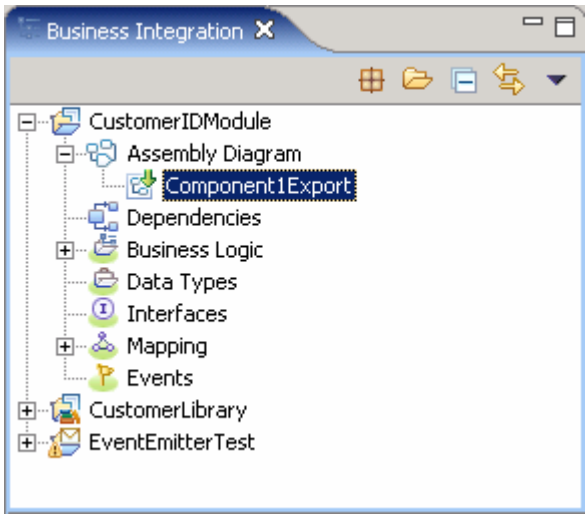
__ b. An empty assembly editor for the **EventEmitterTest** mediation module is opened, as shown below:



__ c. In the Business Integration view's tree, expand the **EventEmitterTest → Mediation Logic → Flows**; select **EmitterTestFlow** mediation flow**,** drag it over the assembly editor's canvas

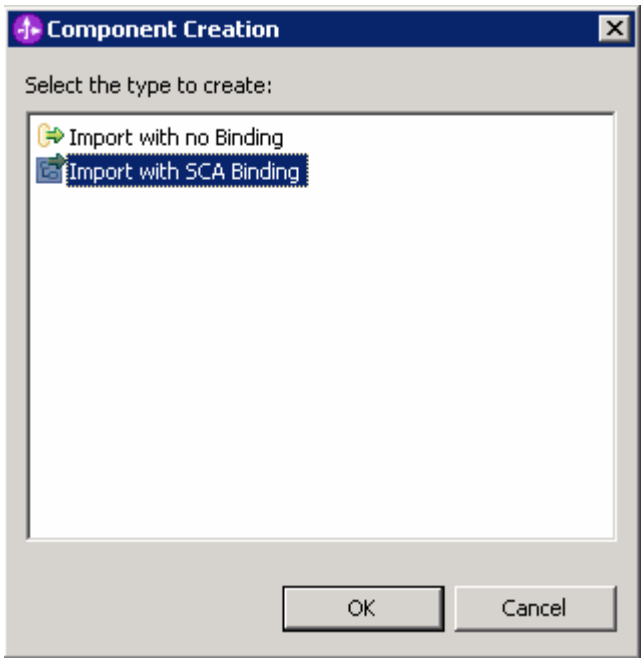


__ d. In the Business Integration view's tree, expand the **CustomerIDModule → Assembly Diagram**; select **Component1Export**



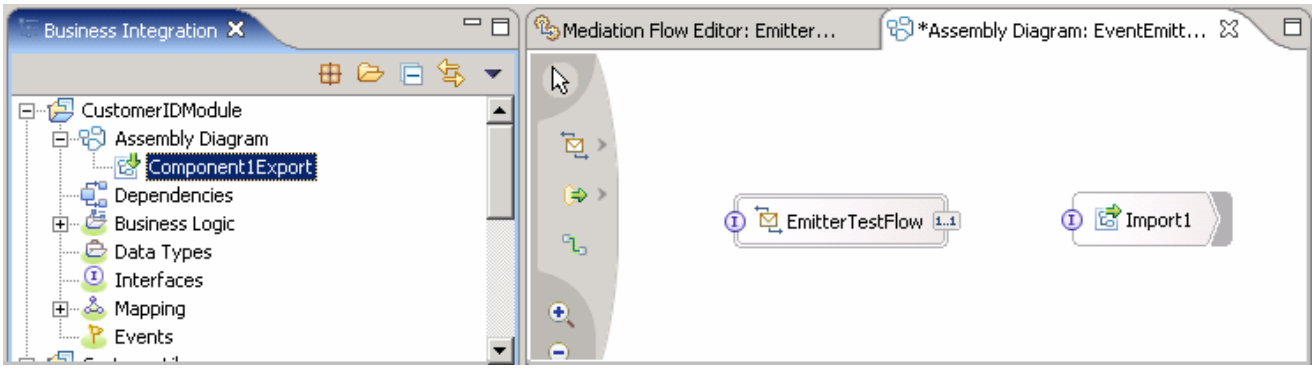
__ e. Drag it over the assembly editor's canvas

__ f. Select **Import with SCA Binding** from the **Component Creation** window that pops-up upon dragging the **Component1Export**.

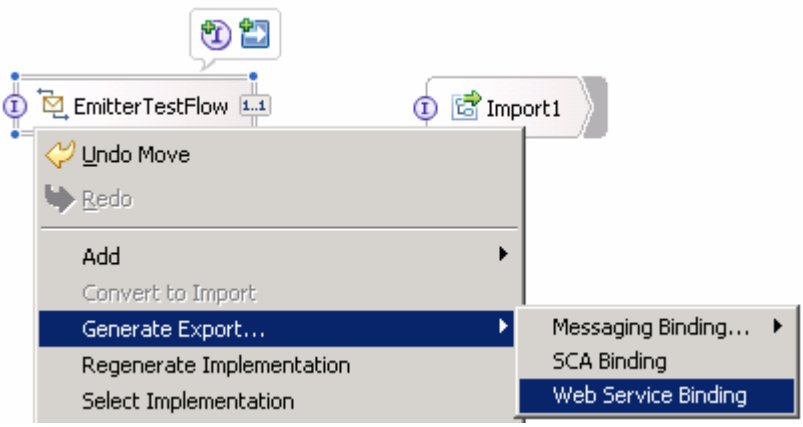


__ g. Click **OK**

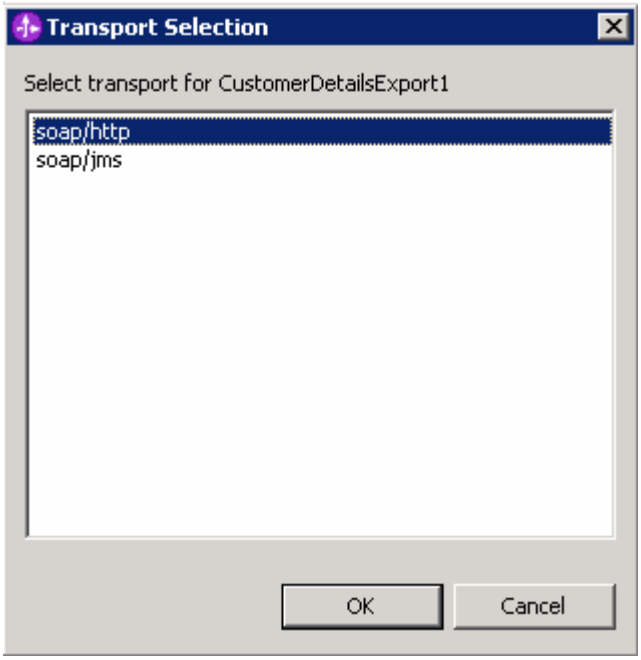
__ h. The assembly diagram must look like the diagram shown below:



__ i. Right click the **EmitterTestFlow** component and select **Generate Export → Web Service Binding** from the context menu



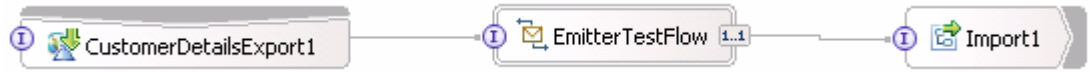
__ j. Select **soap/http** from the Transport Selection pop-up window



__ k. Click **OK**

__ l. The assembly diagram must look like the diagram shown below:

__ m. Wire the mediation Flow, **EmitterTestFlow** and **Import1**



__ n. Save all work by choosing **File > Save All** or **Crtl + Shift + S**

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

Part 3: Deploy the modules and configure Web services explorer

In this section of the lab, the Mediation Module is installed (or deployed) to the WebSphere ESB Server 6.0 test server and the Web services explorer is configured to prepare the ground for testing.

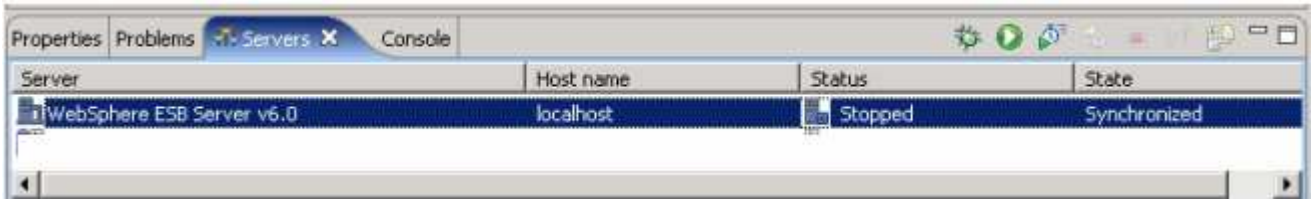
____ 1. Start the **WebSphere ESB Server** if not started and **add modules** to the server

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:

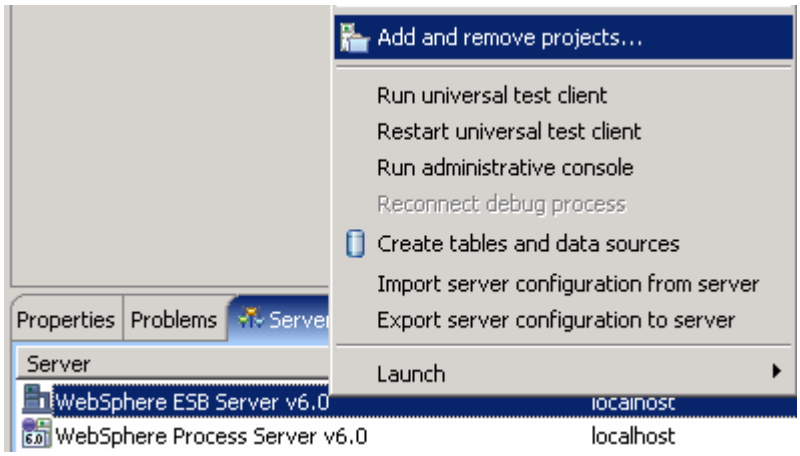
__ a. Open **Servers View**

__ b. Select the WebSphere ESB Server v6.0 and right-click to select "( ) **Start**" from the context menu



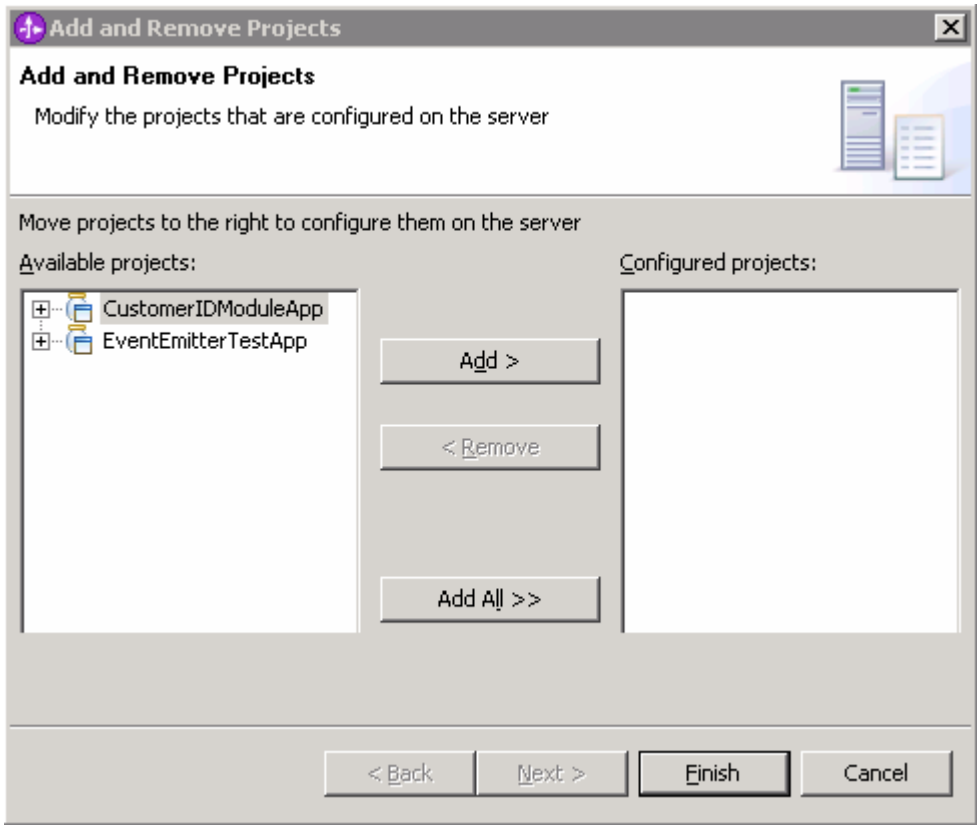
__ c. This takes some time. Wait for the server to start

__ d. Add the **projects** to WebSphere ESB Server. In the servers view, right-click on WebSphere Process Server v6.0 and select "**Add and Remove Projects...**" from the context menu

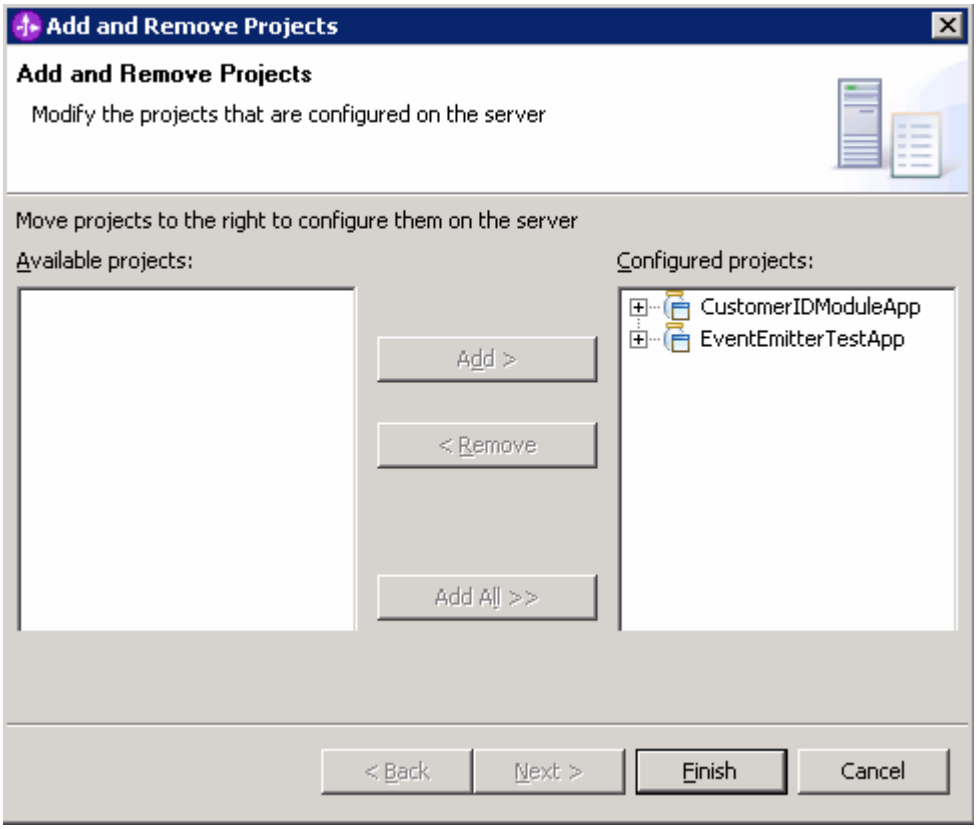


***NOTE*:** Note that the WebSphere ESB server that is being used is configured with an ESB profile that is part of the installation and not part of the workspace. Therefore, if there are any projects deployed to the server from a different workspace, there may be some naming conflicts or other problems. If this occurs, open the Administrative Console and stop/uninstall those projects before adding these projects. This should avoid any potential errors.

__ e. The available projects are listed as shown below:

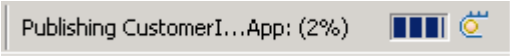


__ f. Click the **Add-All>>** button to move all projects to server



__ g. Click **Finish**

__ h. Wait for the deployment to finish. While the project is deploying you will see something like the following in the lower right corner of WebSphere Integration Developer

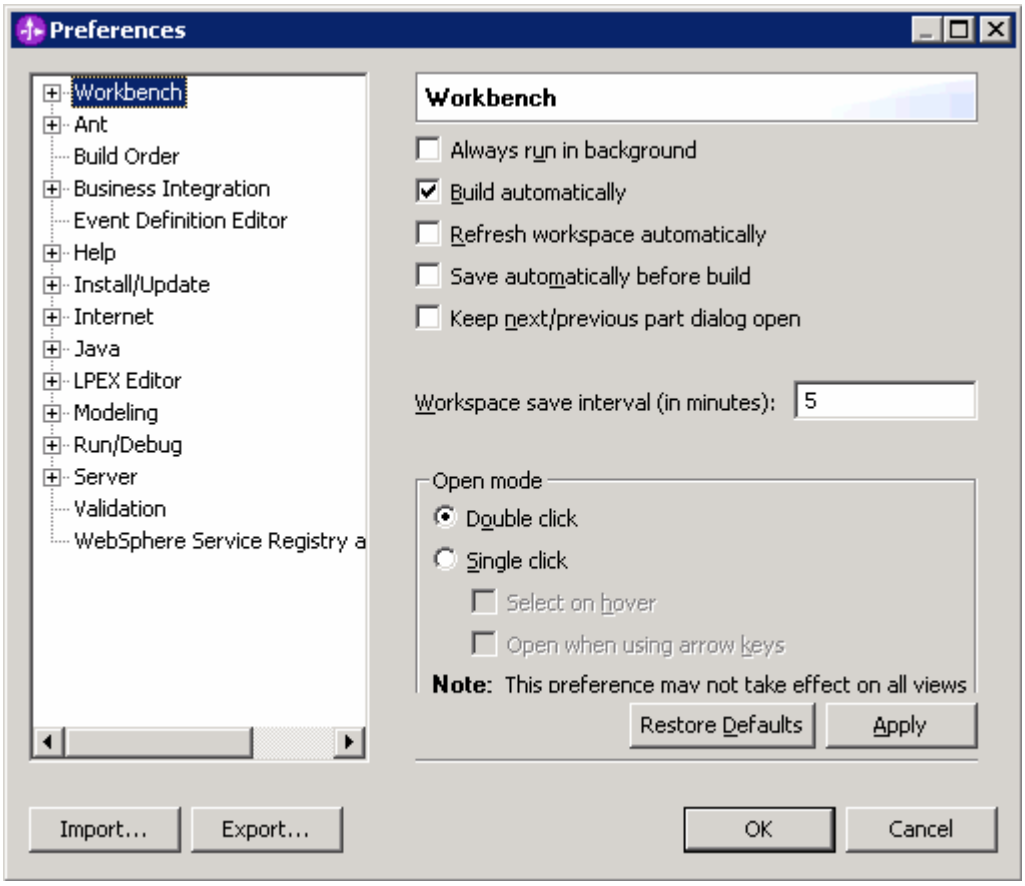


Configure WebServices Explorer:

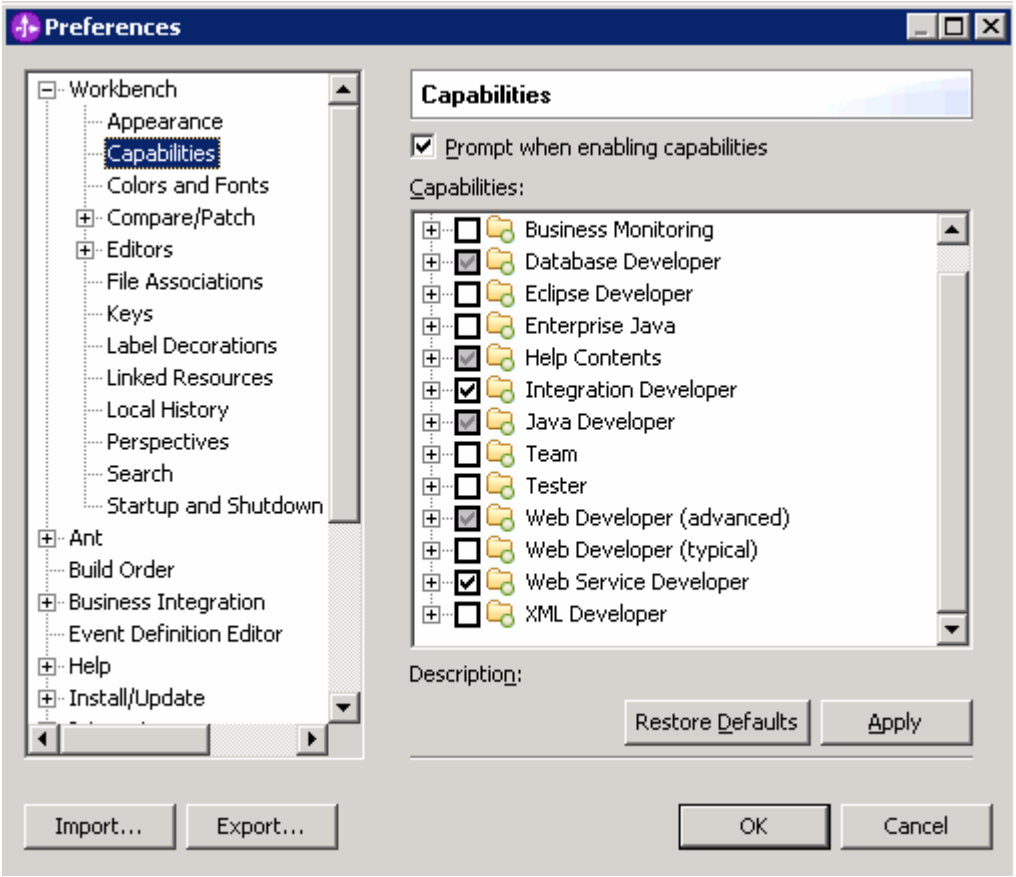
____ 3. To Configure WebServices Explorer in the WebSphere Integration Developer, complete the steps below:

__ a. In the WebSphere Integration Developer's main menu, select **Window > Preferences**

__ b. The **Preferences** window opens



__ c. From the **Preferences** window, expand **Workbench** and select **Capabilities**. Select the check box next to **Web Service Developer** in the Capabilities frame



__ d. Click **OK**

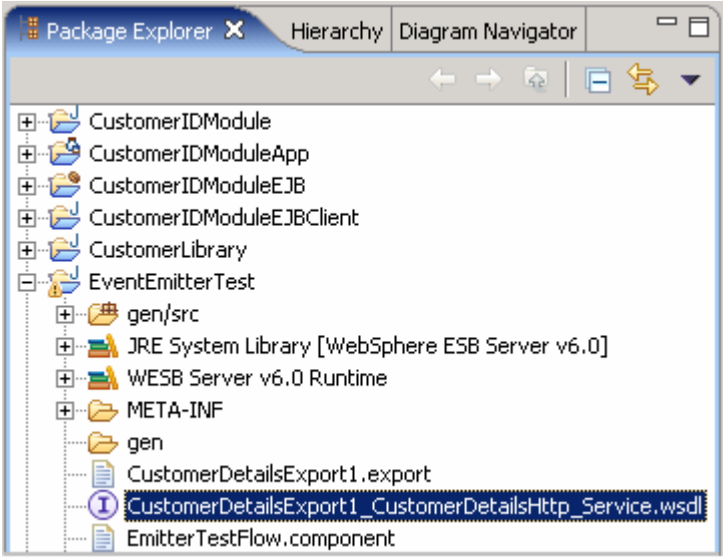
__ e. The WebServices Explorer is configured

Copy the EventEmitterTest *Service.wsdl* **file into** *CustomerLibrary*:

____ 4. For the WebServices Explorer to resolve the business object type correctly, copy the **CustomerDetailsExport1_CustomerDetailsHttp_Service.wsdl** file into the **CustomerLibrary** project. To achieve this, complete the steps below:

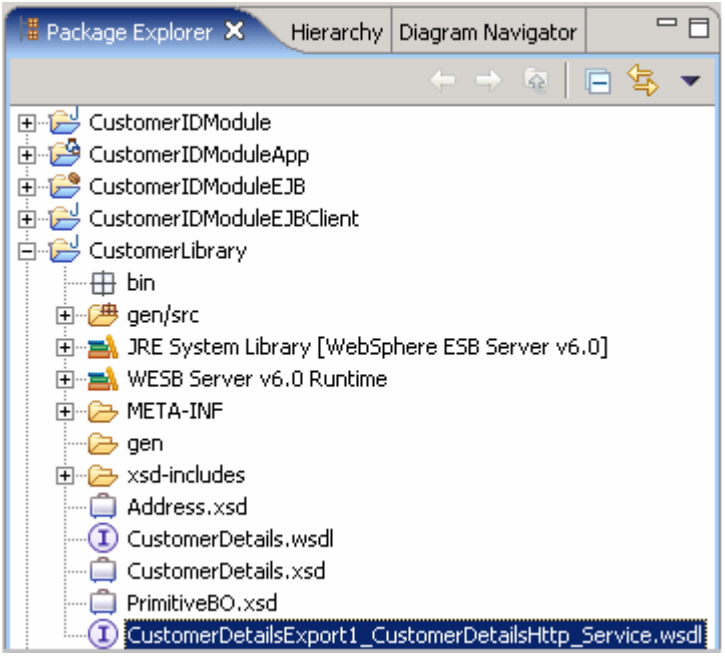
__ a. Open the Java Perspective in the WebSphere Integration Developer. (**Window → Open Perspective → Other..** and select **Java** from the **Select Perspective** window and click **OK**

__ b. In the Java Perspective, expand the EventEmitterTest project and locate the **CustomerDetailsExport1_CustomerDetailsHttp_Service.wsdl** file. Right-click on this WSDL file and select **Copy** from the context menu

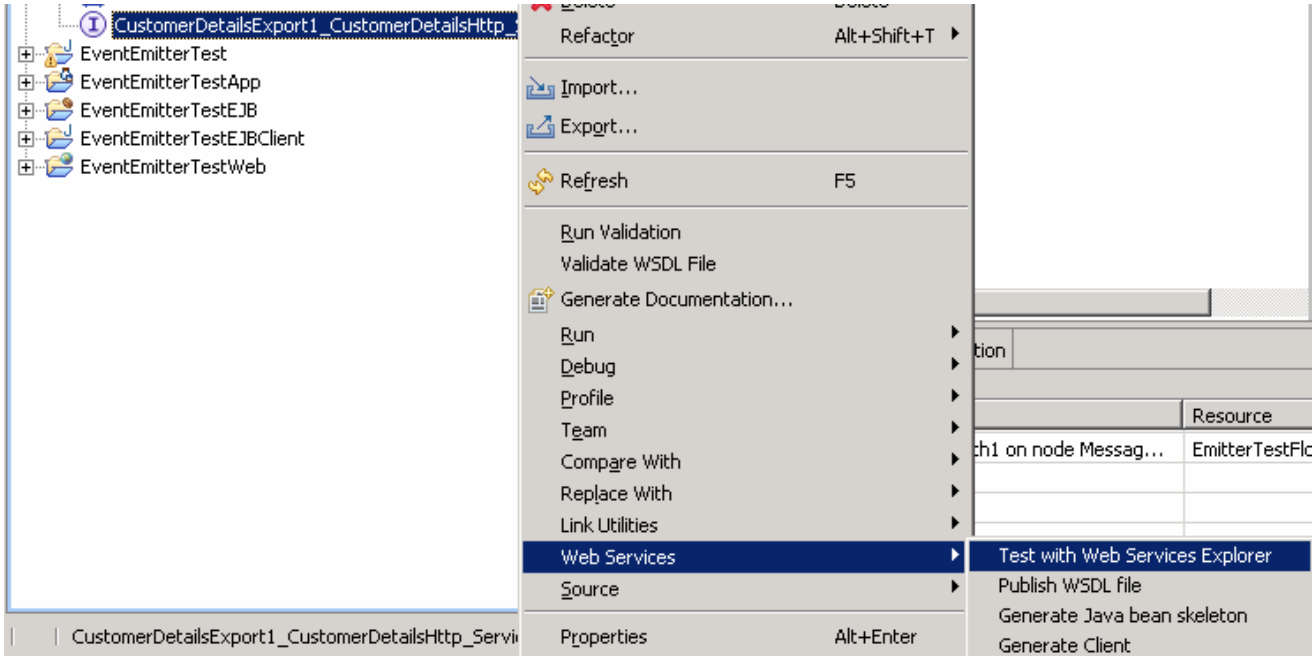


__ c. Now, right-click the **CustomerLibrary** project and select **Paste** from the context menu

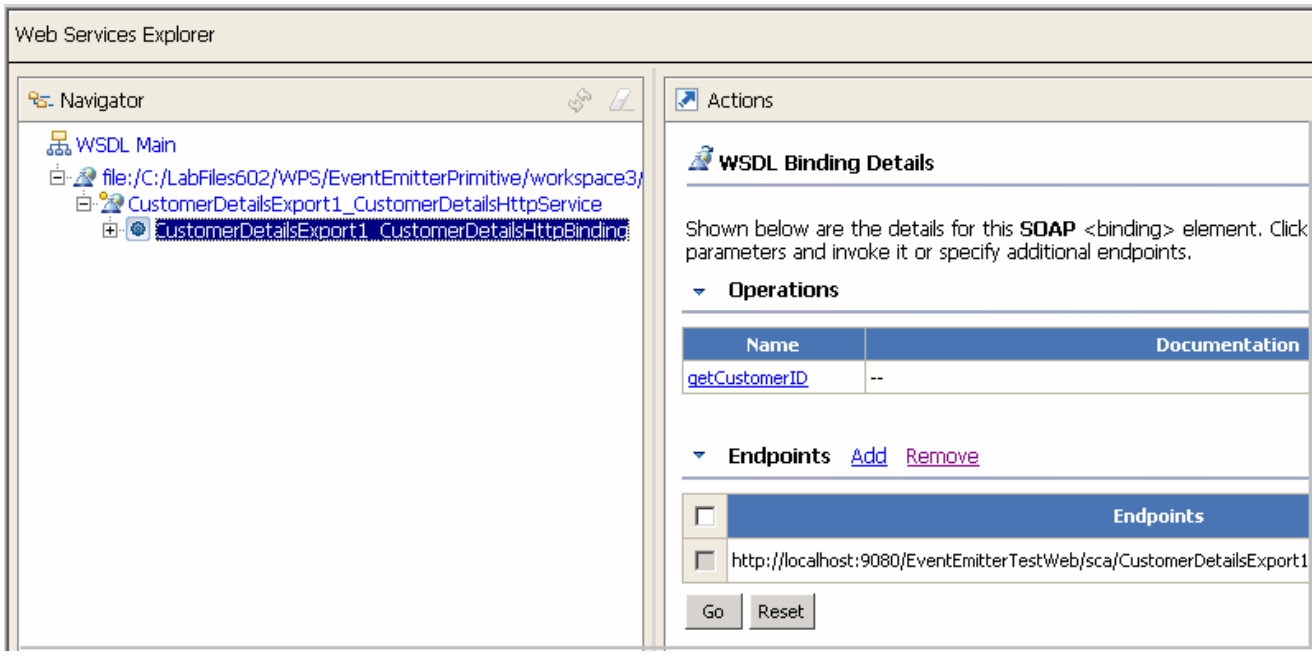
Ignore any warnings on the problems tab that pop up.



__ d. Now expand the **CustomerLibrary** project; right-click the **CustomerDetailsExport1_CustomerDetailsHttp_Service.wsdl** file and select **Web Services → Test with Web Services Explorer** from the context menu

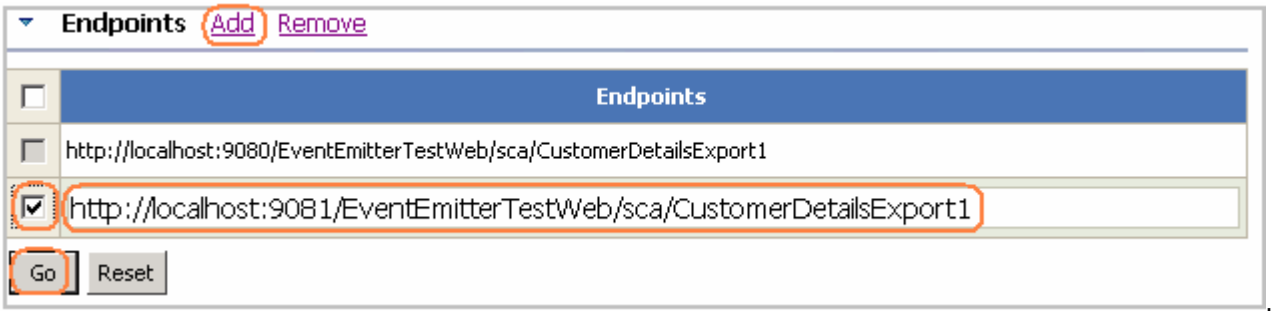


__ e. The Web Services Explorer opens as shown below:



** Note that Status may show errors encountered while validating XML. Do not worry about them for now.

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.

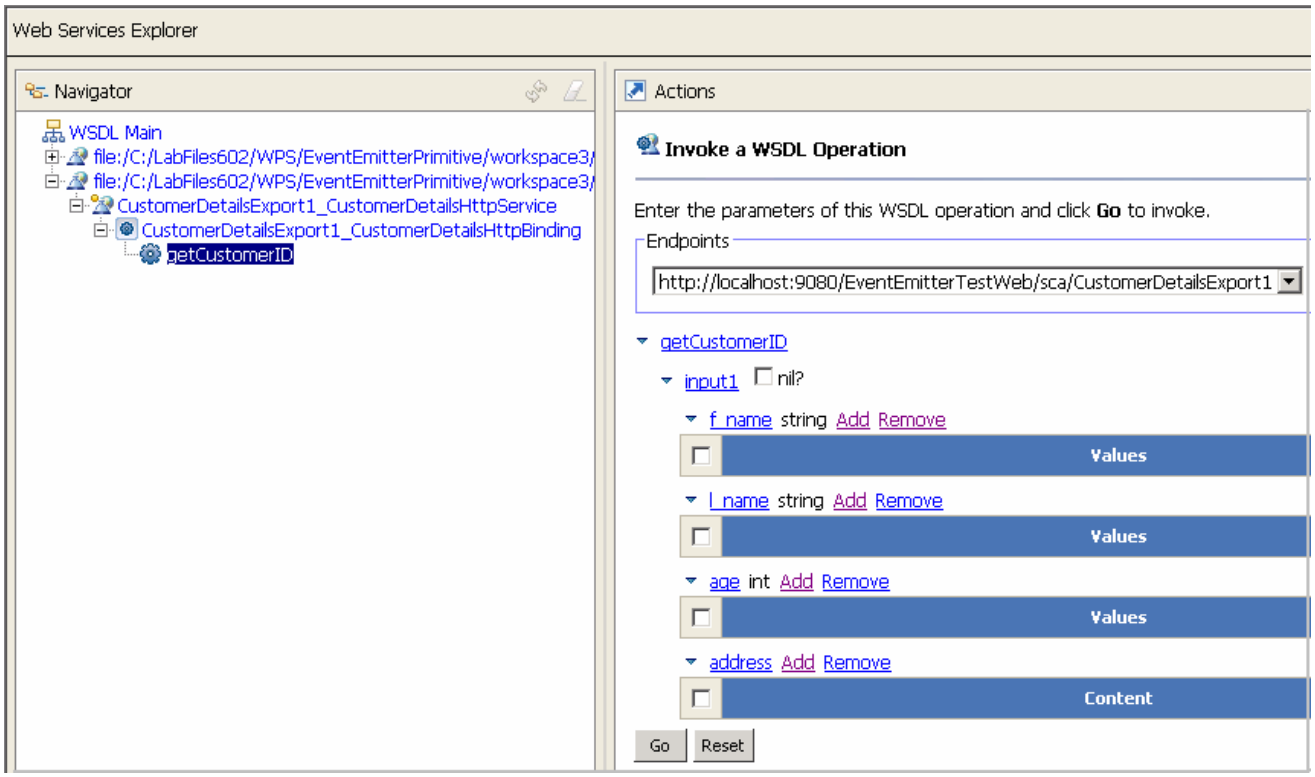


Part 4: Event generation test A and B using the Web services explorer

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:

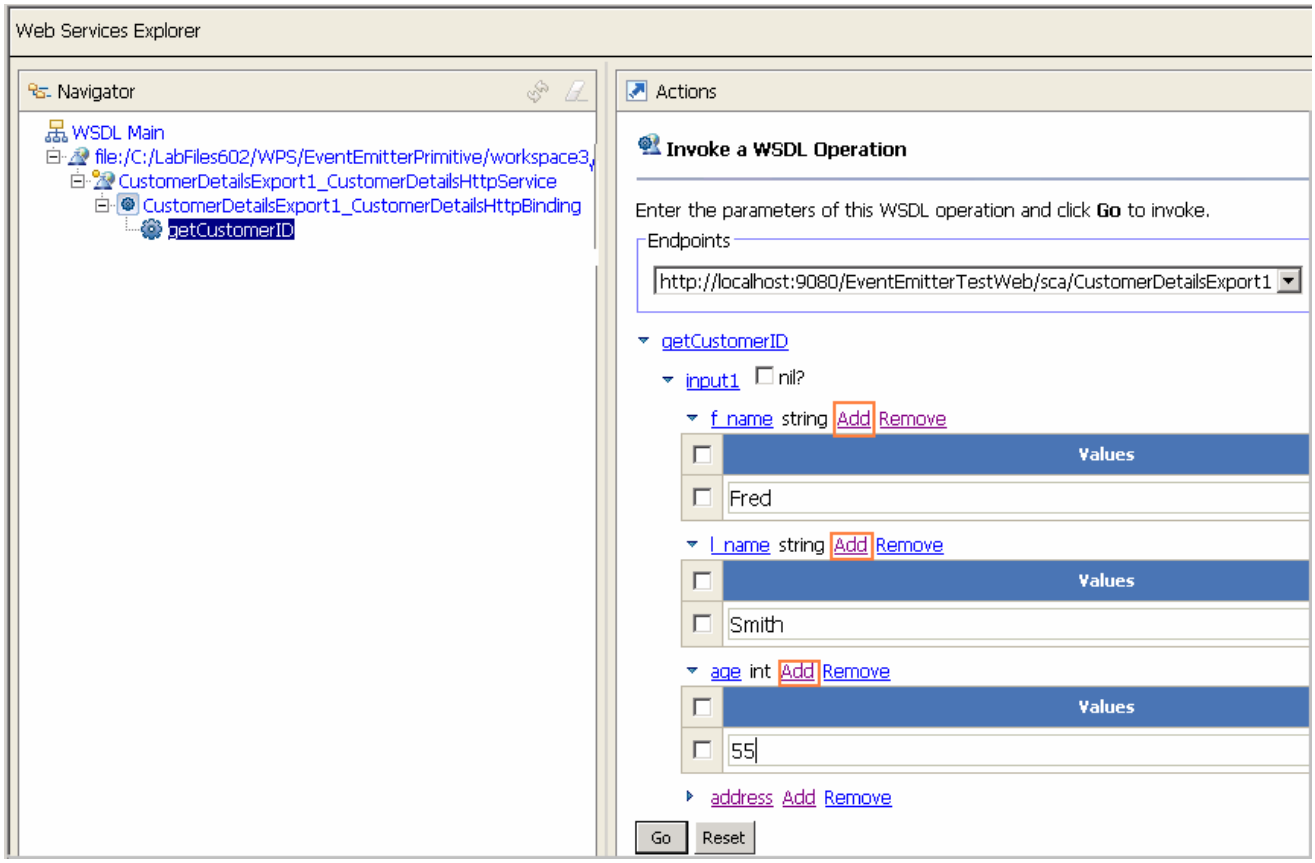


____ 2. Enter the following values in the following fields on the Actions pane of the explorer by clicking on the **Add** link to add a blank field for each of the values below:

__ a. f_name: **Fred**

__ b. l_name: **Smith**

__ c. age : **55**

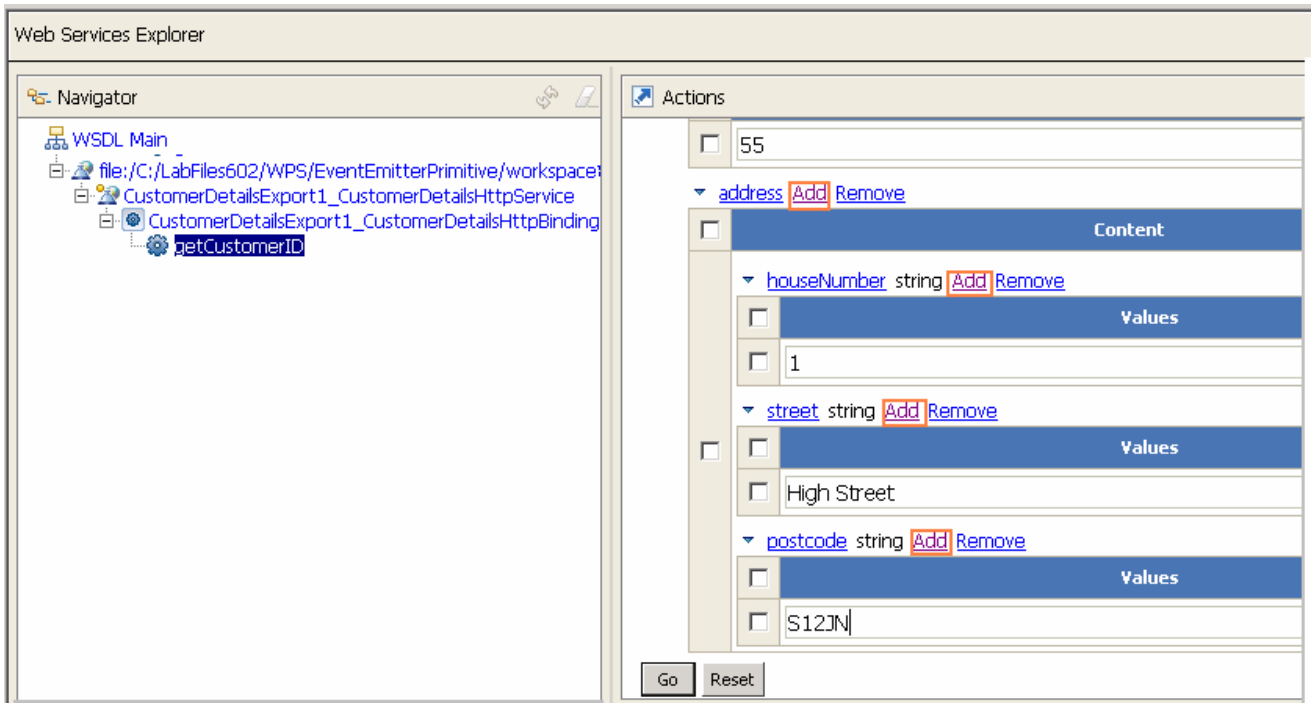


____ 3. Scroll down and click the **Add** link next **address** to add the address fields. Enter the following values for the address fields:

__ a. houseNumber: **1**

__ b. street : **High Street**

__ c. postcode : **S12JN**



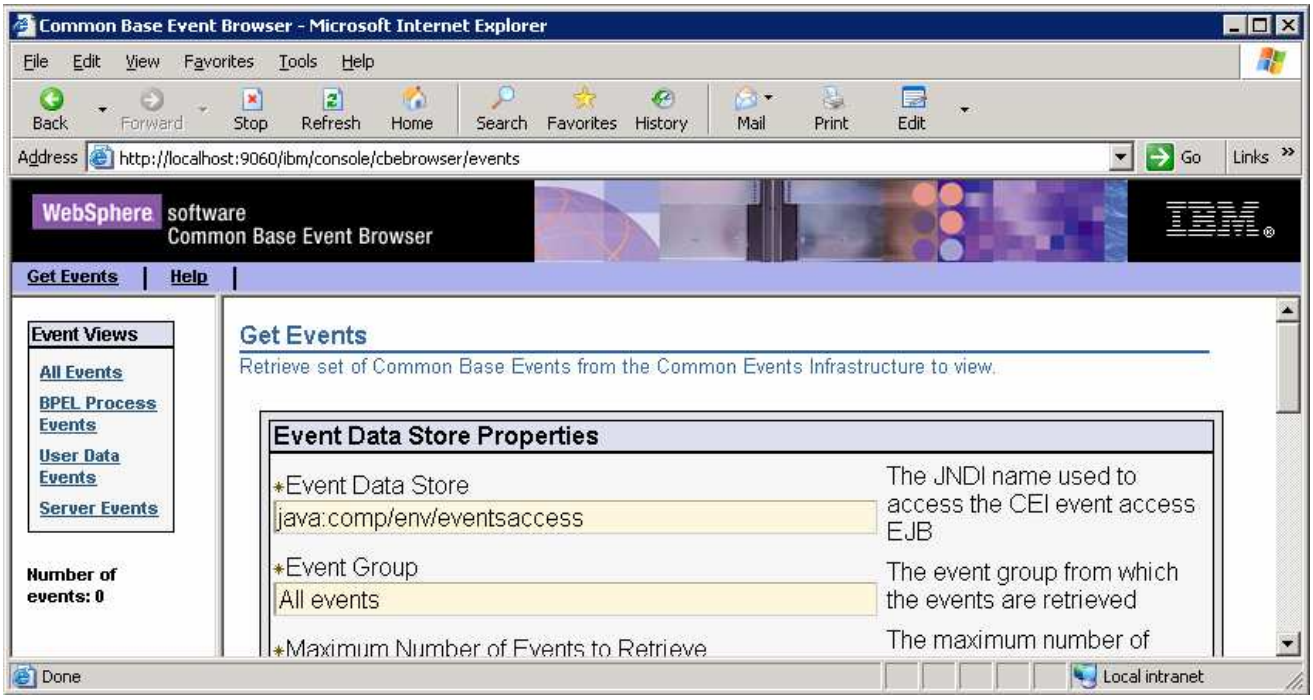
____ 4. Click on the **Go** button to generate the Events

____ 5. Ensure that there are no errors displayed in the **Status** pane at the bottom

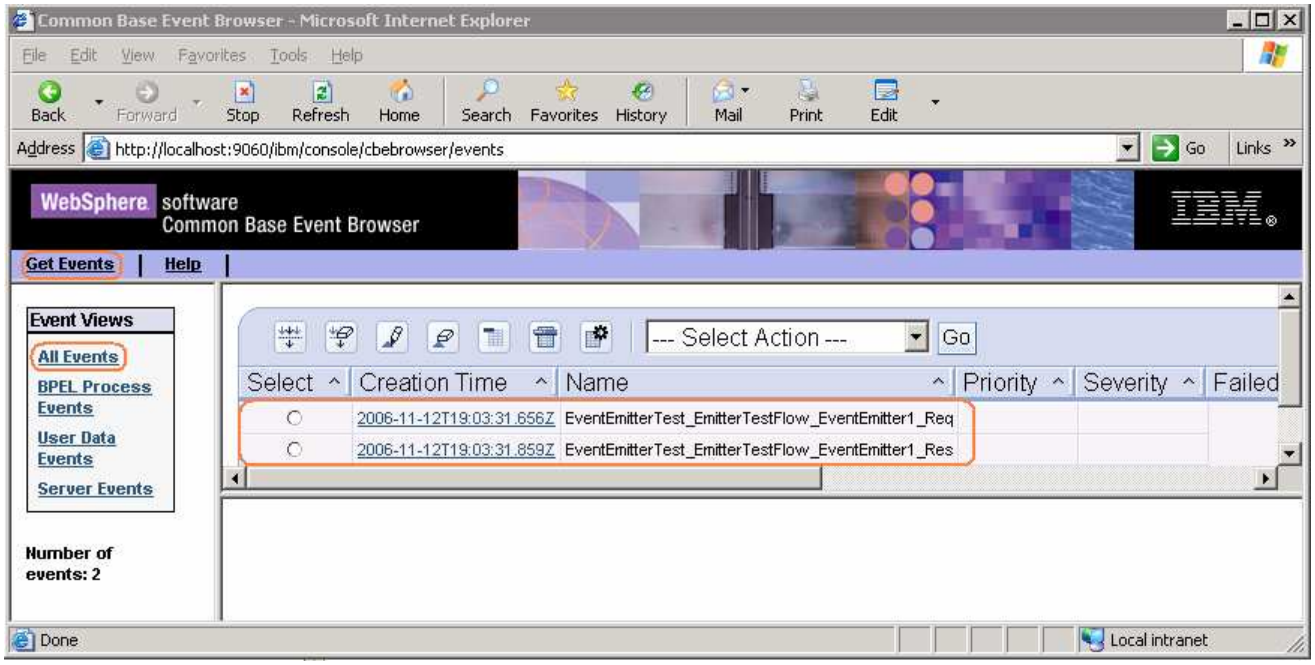
____ 6. To confirm that the Events are generated, open the **Common Base Event browser** at http://hostname:<port>/ibm/console/cbebrowser/events/ as shown in the diagram below:

Note: The **hostname** is a fully qualified name of the machine and **port** is the **default_http** port of the ESB server. Ex: http://localhost:9080/ibm/console/cbebrowser/events/

Also you can open the common base event browser by switching to the Business Integration perspective and in the server view, right-click on the server and select **launch->Common Base Event Browser**



____ 7. From the Event Browser, on the top left of the navigation menus, 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 will display the events in the Event Data pane as shown here:

Event Data

List of all properties associated with the selected event.

Name	Value
version	1.0.1
globalInstanceId	CE7CFF2326CFB5B58CA1DB72807D098D80
extensionName	EventEmitterTest_EmitterTestFlow_EventEmitter1_Req
localInstanceId	
creationTime	2006-11-12T19:03:31.656Z
severity	
msg	
priority	
sequenceNumber	9
repeatCount	
elapsedTime	
contextDataElement / WBISESSION_ID / contextValue	9.3.75.99;EventEmitterTest;;getCustomerID;1163358211359;1007133850
contextDataElement / ECSCurrentID / contextValue	9.3.75.99;EventEmitterTest;null;;getCustomerID;1163358211359;1007133850
contextDataElement / ECSParentID / contextValue	9.3.75.99;EventEmitterTest;;getCustomerID;1163358211359;1007133850
extendedDataElement / EventNature	CUSTOM
extendedDataElement / PayloadType	full
extendedDataElement / ModuleName	EventEmitterTest
extendedDataElement / MediationName	EventEmitter1
extendedDataElement / Root	No Data

____ 9. Look for the following elements in the event to ensure the correct data has been generated:-

__ a. extensionName → **EventEmitterTest_EmitterTestFlow_EventEmitter1_Req**

__ b. extendedDataElement/ModuleName → **EventEmitterTest**

__ c. extendedDataElement/MediationName → **EventEmitter1**

__ d. extendedDataElement/Root → **No Data**

____ 10. 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:

Event Data

List of all properties associated with the selected event.

Name	Value
version	1.0.1
globalInstanceId	CE7CFF2326CFB5B58CA1DB72807D261630
extensionName	EventEmitterTest_EmitterTestFlow_EventEmitter1_Res
localInstanceId	
creationTime	2006-11-12T19:03:31.859Z
severity	
msg	
priority	
sequenceNumber	10
repeatCount	
elapsedTime	
contextDataElement / WBISESSION_ID / contextValue	9.3.75.99;EventEmitterTest;;getCustomerID;1163358211359;1007133850
contextDataElement / ECSCurrentID / contextValue	9.3.75.99;EventEmitterTest;null;;getCustomerID;1163358211359;1007133850
contextDataElement / ECSParentID / contextValue	9.3.75.99;EventEmitterTest;;getCustomerID;1163358211359;1007133850
extendedDataElement / EventNature	CUSTOM
extendedDataElement / PayloadType	full
extendedDataElement / ModuleName	EventEmitterTest
extendedDataElement / MediationName	EventEmitter1
extendedDataElement / Root	/body/getCustomerIDResponse/customerID
extendedDataElement / Message	ABCDE12345

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

__ a. extensionName → **EventEmitterTest_EmitterTestFlow_EventEmitter1_Res**

__ b. extendedDataElement/ModuleName → **EventEmitterTest**

__ c. extendedDataElement/MediationName → **EventEmitter1**

__ d. extendedDataElement/Root → **/body/getCustomerIDResponse/customerID**

__ e. extendedDataElement/Message → **ABCD12345**

____ 12. The Event Generation Test A is complete

Event Generation Test B

____ 13. 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 may 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**

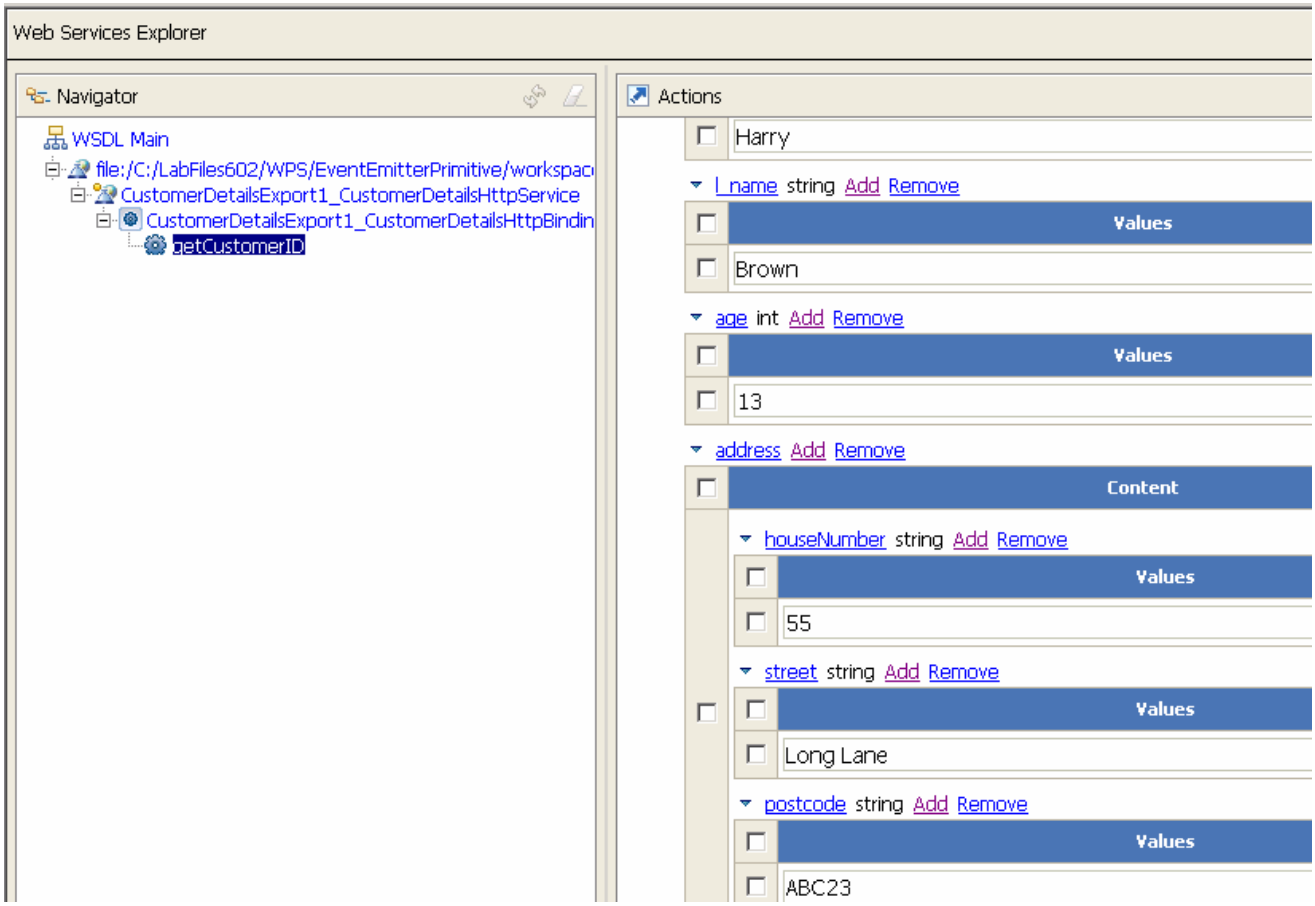
Address

__ d. houseNumber : **55**

__ e. street : **Long Lane**

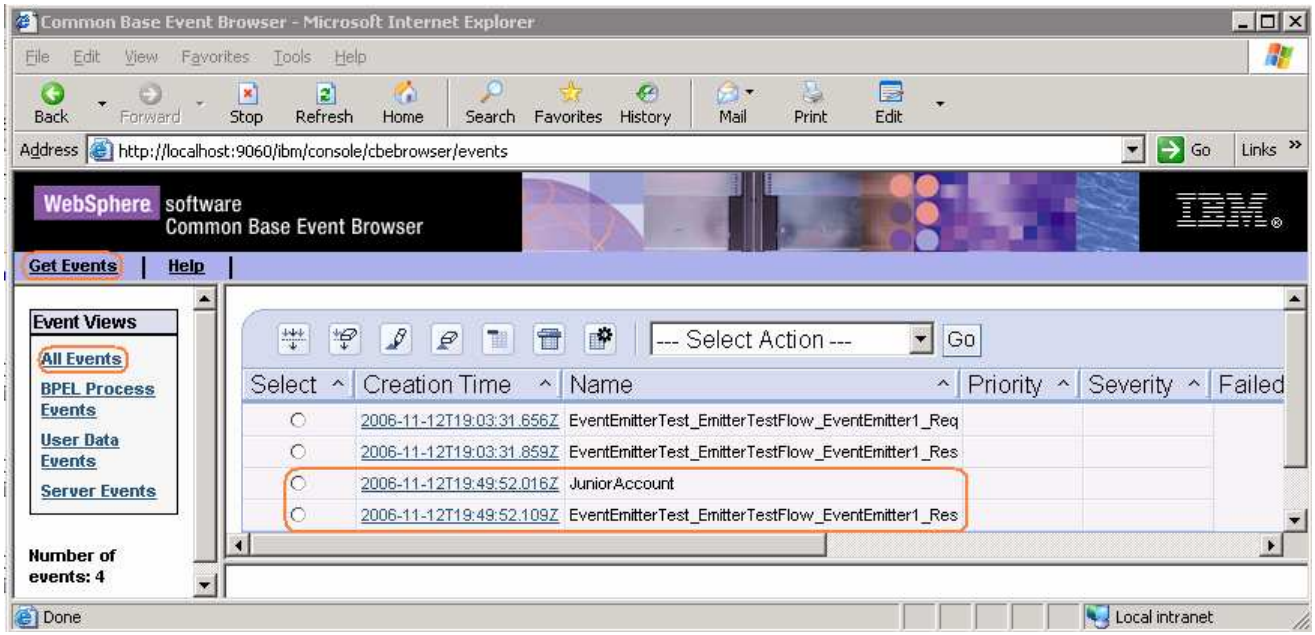
__ f. postcode : **ABC23**

____ 14. The Explorer with the entered values must look as shown below:



____ 15. Click on the **Go** button to generate the Events

____ 16. 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:



____ 17. 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:

extensionName	JuniorAccount
localInstanceId	
creationTime	2006-11-12T19:49:52.016Z
severity	
msg	
priority	
sequenceNumber	11
repeatCount	
elapsedTime	
contextDataElement / WBISESSION_ID / contextValue	9.3.75.99;EventEmitterTest;;getCustomerID;1163360992000;1252975837
contextDataElement / ECSCurrentID / contextValue	9.3.75.99;EventEmitterTest;null;;getCustomerID;1163360992000;1252975837
contextDataElement / ECSParentID / contextValue	9.3.75.99;EventEmitterTest;;getCustomerID;1163360992000;1252975837
extendedDataElement / EventNature	CUSTOM
extendedDataElement / PayloadType	full
extendedDataElement / ModuleName	EventEmitterTest
extendedDataElement / MediationName	EventEmitter2
extendedDataElement / Root	/body
extendedDataElement / Message	
extendedDataElement / Message / getCustomerID	
extendedDataElement / Message / getCustomerID / input1	
extendedDataElement / Message / getCustomerID / input1 / f_name	Harry
extendedDataElement / Message / getCustomerID / input1 / l_name	Brown
extendedDataElement / Message / getCustomerID / input1 / age	13
extendedDataElement / Message / getCustomerID / input1 / address	
extendedDataElement / Message / getCustomerID / input1 / address / houseNumber	55
extendedDataElement / Message / getCustomerID / input1 / address / street	Long Lane

____ 18. Look for the following elements in the event to ensure the correct data has been generated:-

__ a. extensionName → **JuniorAccount**

__ b. extendedDataElement/ModuleName → **EventEmitterTest**

__ c. extendedDataElement/MediationName → **EventEmitter2**

__ d. extendedDataElement/Root → **/body**

__ e. extendedDataElement / Message / getCustomerID / input1 / f_name : **Harry**

__ f. extendedDataElement / Message / getCustomerID / input1 / l_name : **Brown**

__ g. extendedDataElement / Message / getCustomerID / input1 / age : **13**

__ h. extendedDataElement / Message / getCustomerID / input1 / address / houseNumber : **55**

__ i. extendedDataElement / Message / getCustomerID / input1 / address / street : **Long Lane**

__ j. extendedDataElement / Message / getCustomerID / input1 / address / postcode : **ABC23**

____ 19. 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:

Event Data

List of all properties associated with the selected event.

Name	Value
version	1.0.1
globalInstanceId	CE7CFF2326CFB5B58CA1DB7286F64E91D0
extensionName	EventEmitterTest_EmitterTestFlow_EventEmitter1_Res
localInstanceId	
creationTime	2006-11-12T19:49:52.109Z
severity	
msg	
priority	
sequenceNumber	12
repeatCount	
elapsedTime	
contextDataElement / WBISESSION_ID / contextValue	9.3.75.99;EventEmitterTest;;getCustomerID;1163360992000;1252975837
contextDataElement / ECSCurrentID / contextValue	9.3.75.99;EventEmitterTest;null;;getCustomerID;1163360992000;1252975837
contextDataElement / ECSParentID / contextValue	9.3.75.99;EventEmitterTest;;getCustomerID;1163360992000;1252975837
extendedDataElement / EventNature	CUSTOM
extendedDataElement / PayloadType	full
extendedDataElement / ModuleName	EventEmitterTest
extendedDataElement / MediationName	EventEmitter1
extendedDataElement / Root	/body/getCustomerIDResponse/customerID
extendedDataElement / Message	ABCDE12345

____ 20. Look for the following elements in the event to ensure the correct data has been generated:-

__ a. extensionName → **EventEmitterTest_EmitterTestFlow_EventEmitter1_Res**

__ b. extendedDataElement/ModuleName → **EventEmitterTest**

__ c. extendedDataElement/MediationName → **EventEmitter1**

__ d. extendedDataElement/Root → **/body/getCustomerIDResponse/customerID**

__ e. extendedDataElement/Message → **ABCDE12345**

____ 21. The Event Generation Test B is complete

____ 22. 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.

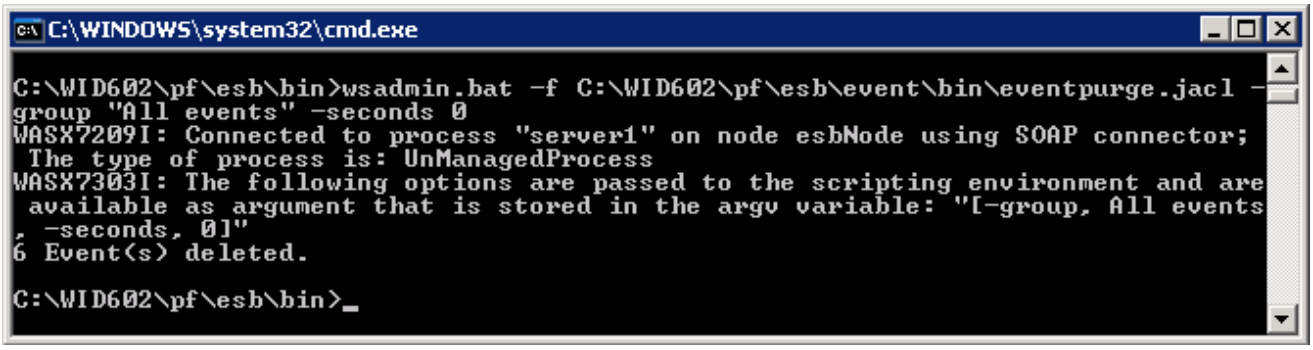
Note: To avoid confusion, delete all the events generated as result of the actions during the course of Part 4. To achieve this:

→ Open a command line window and change directory to <WID_HOME>\pf\esb\bin> that is, E:\WID602\pf\esb\bin

> ***Note:*** If you are testing on a remote server, go to the <WAS_HOME>bin directory of the server and issue the command as shown (possibly substituting **.sh** for the file extension) and substituting **<WAS_HOME>/profiles/<PROFILE_NAME>event/bin/eventpurge.jacl** for the jacl script name.

→Run the following JACL script to delete all the events:

wsadmin.bat -f E:\WID602\pf\esb\event\bin\eventpurge.jacl -group "All events" -seconds 0

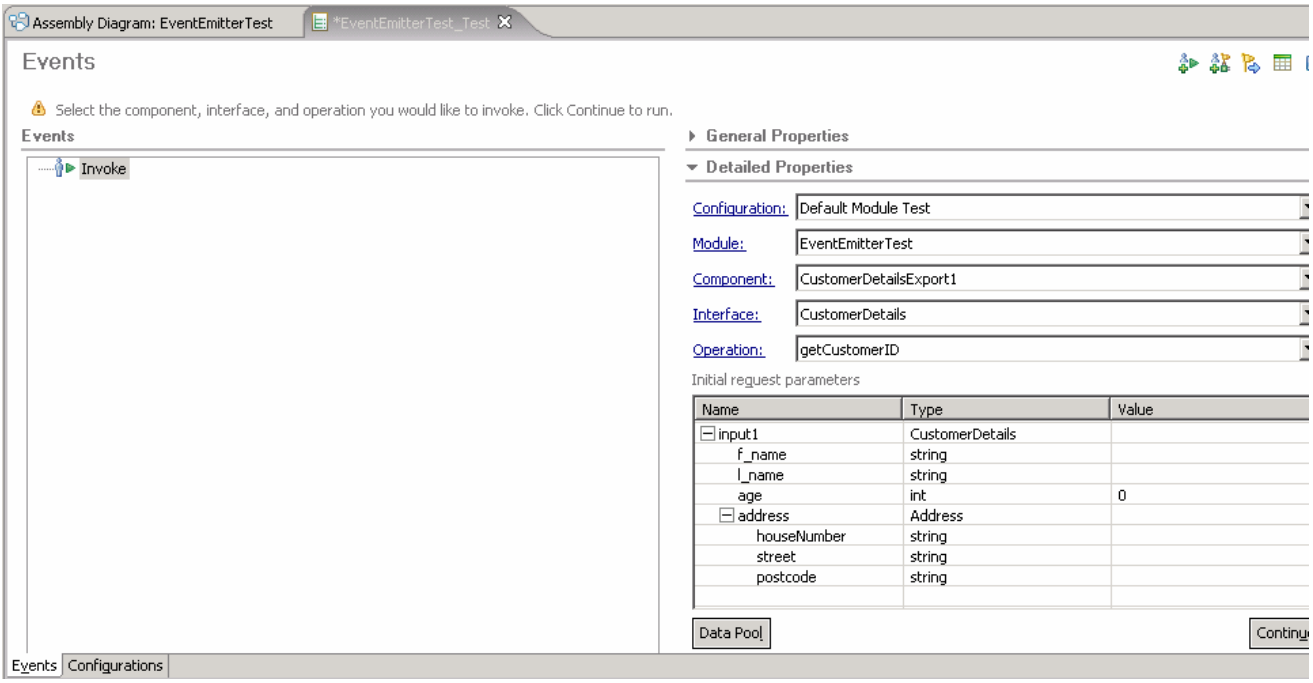


____ 1. On WebSphere Integration Developer, switch to the Business Integration Perspective. To switch to this perspective: **Window → Open Perspective → Other...** and select **Business Integration** from the **Select Perspective** window and click **OK**

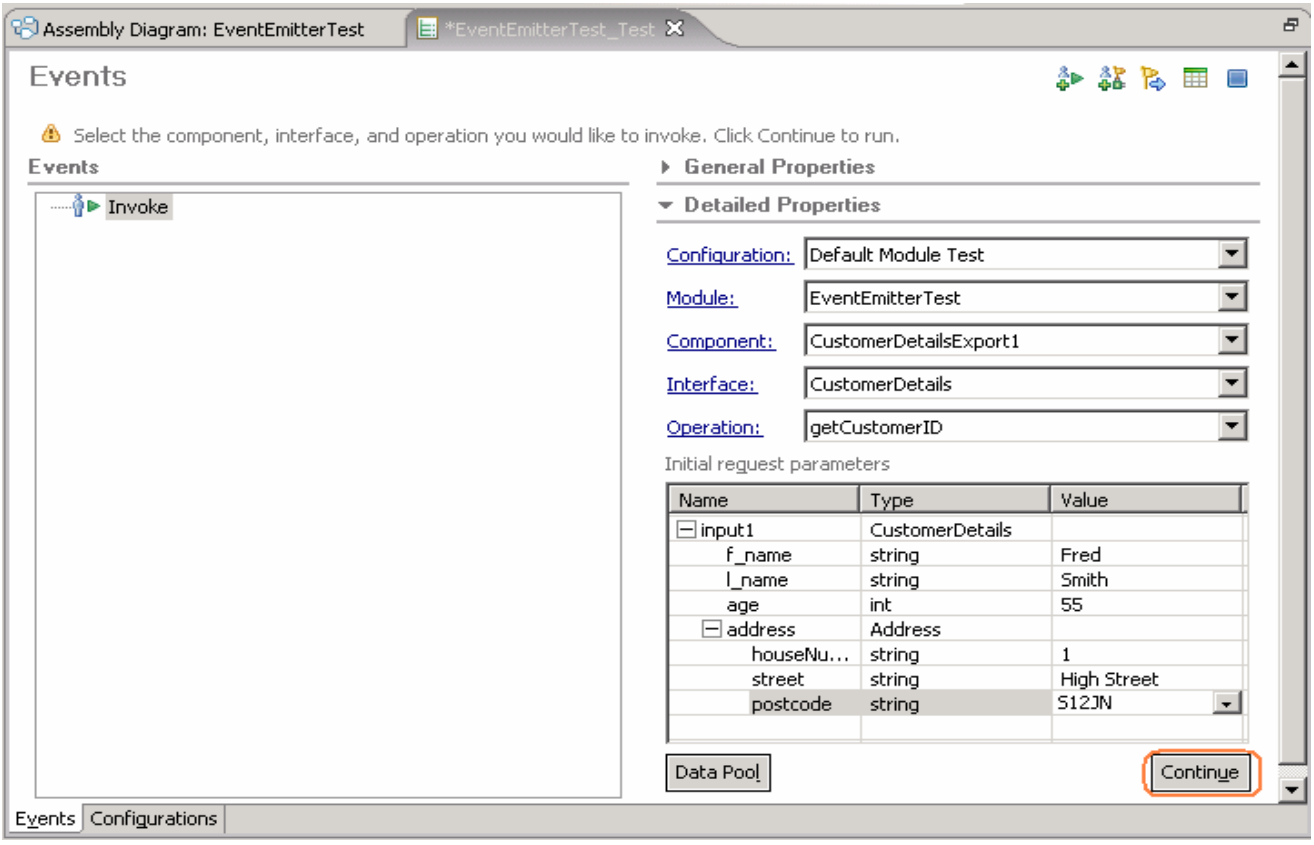
____ 2. In the Business Integration view's tree, expand the **EventEmitterTest** mediation module and double-click the mediation module assembly (Assembly Diagram) to open it with the assembly editor

____ 3. Right-click anywhere on the white space on the canvas and select **Test Module** from the context menu

____ 4. The Test client for **EventEmitterTest** mediation module is opened



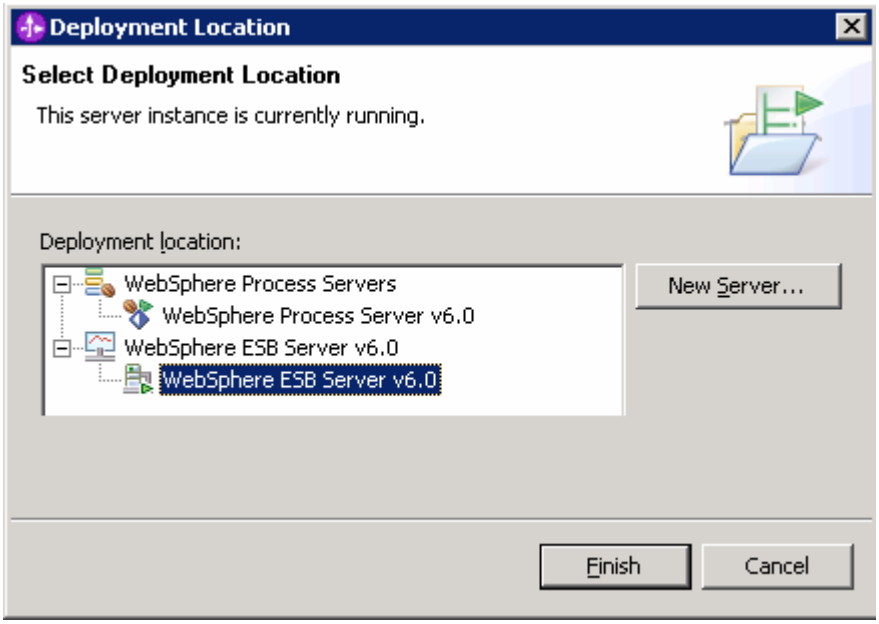
____ 5. Enter the same values in the fields, used for **Event Generation Test A** of Part4, as shown below:



__ a. Click **Continue**

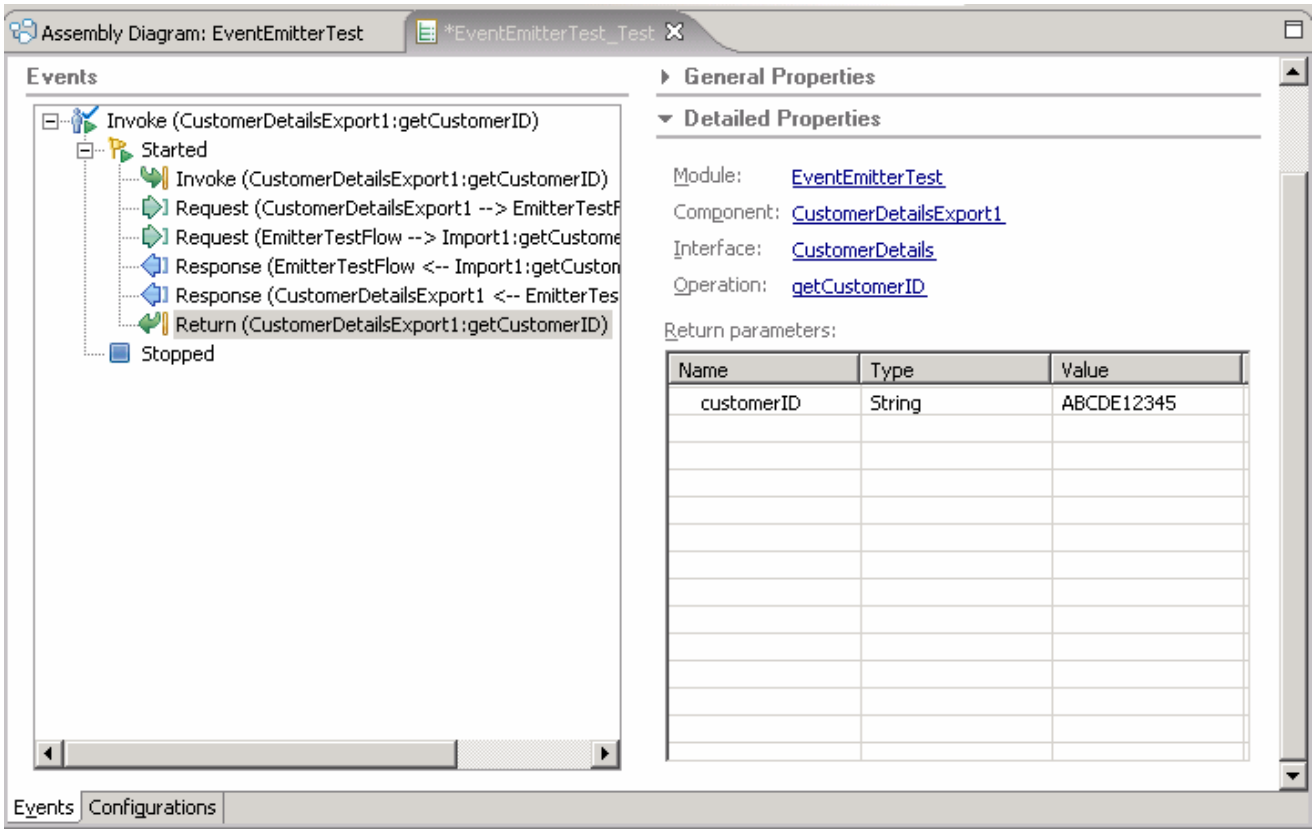
__ b. Select WebSphere ESB Server 6.0 as the deployment location from the pop-up window when prompted

Note: By default WebSphere Process Server 6.0 is selected as the deployment location. Ensure the WebSphere ESB Server 6.0 is selected as a deployment location.



__ c. Click **Finish**

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



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

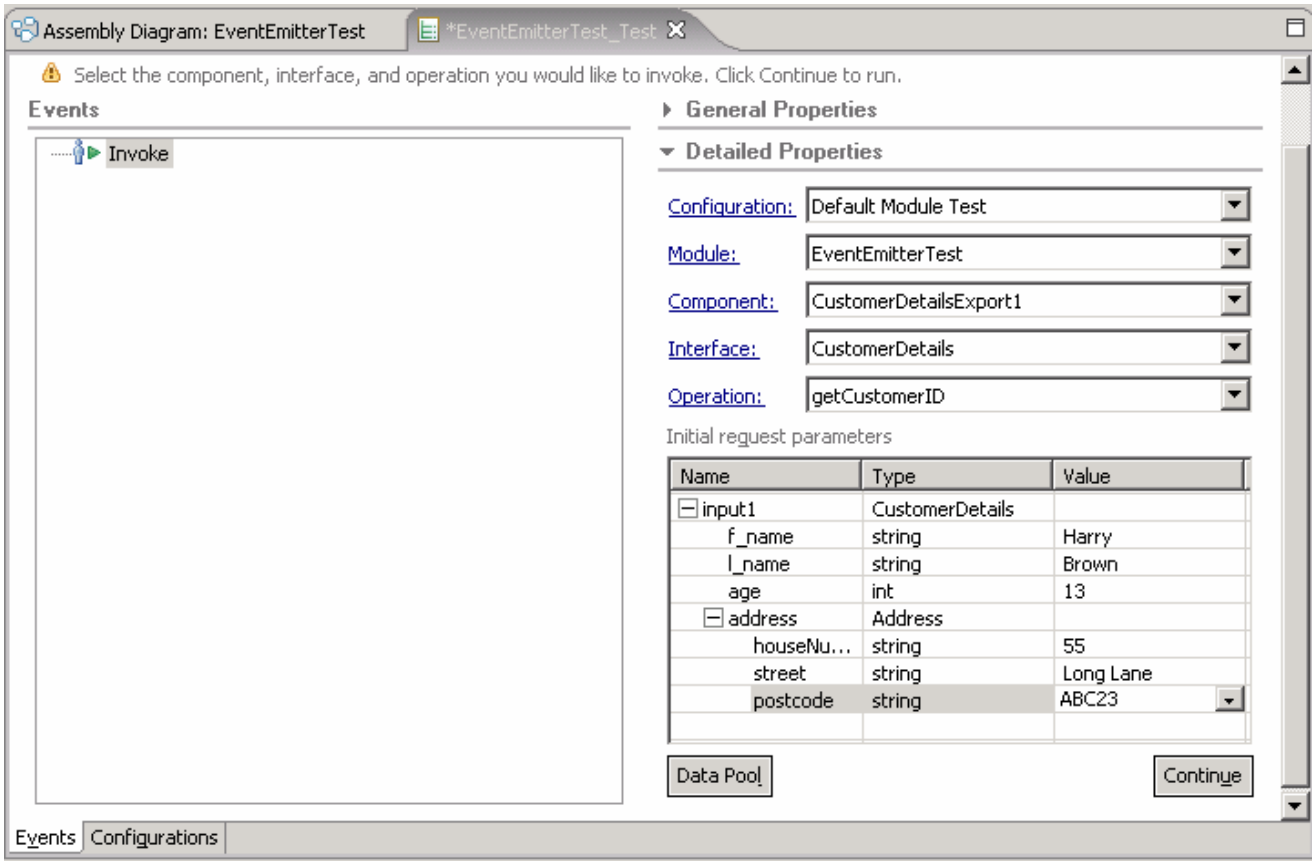
____ 8. Close the Test Client and say no to save changes when prompted

____ 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:

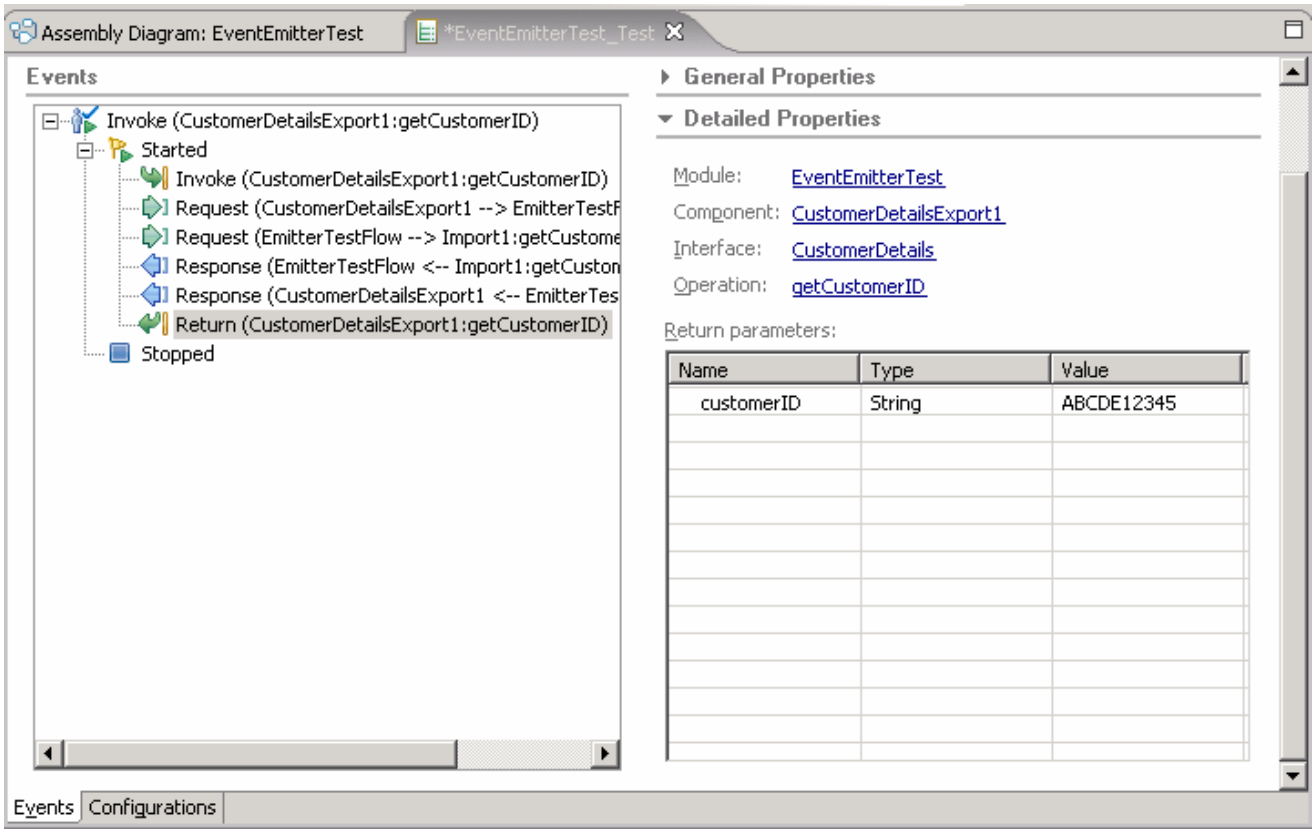


__ d. Click **Continue**

__ e. Select WebSphere ESB Server 6.0 as the deployment location from the pop-up window when prompted

__ f. Click **Finish**

__ g. Ensure the following is resulted on a successful execution:



____ 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:/LabFiles602/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 v6.0 (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 v6.0.

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:/LabFiles602/WESB/EventEmitterPromitive/ WESB_EventEmitterPrimitive_Solution_PI.zip**

__ c. Click **Finish** button

____ 2. Continue with **Part 3, Part 4 & Part 5** of this LAB

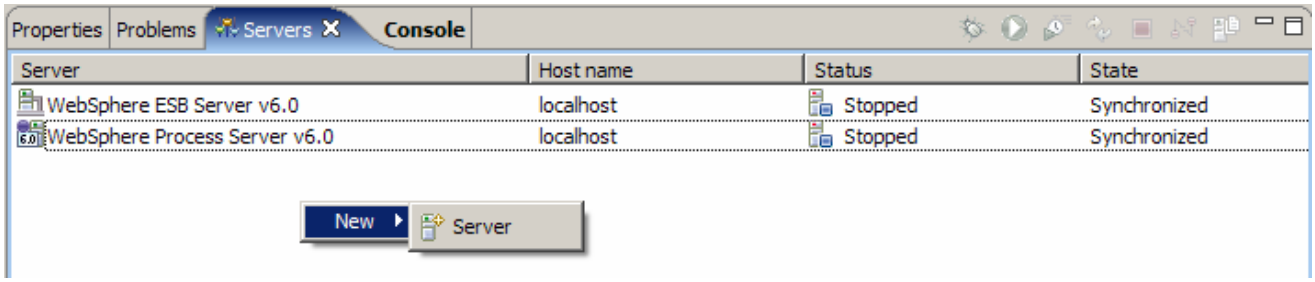
Task: Adding remote server to WebSphere Integration Developer test environment

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

____ 1. Create a new remote server

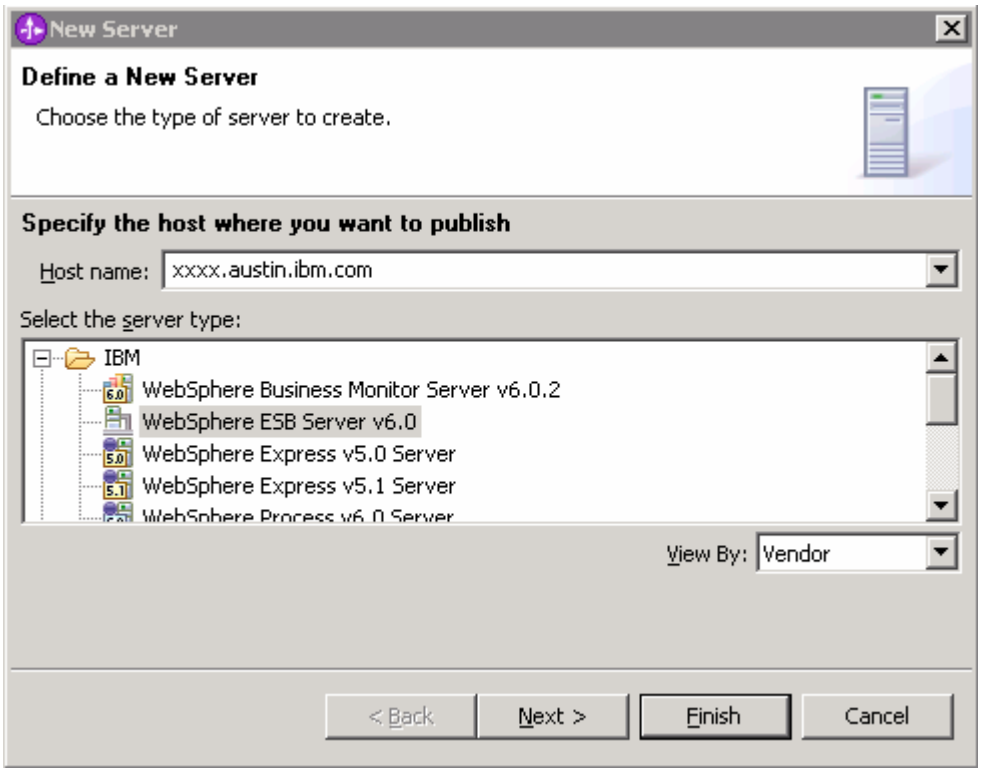
__ a. Right click on the background of the Servers view to access the pop-up menu

__ b. Select **New > Server**



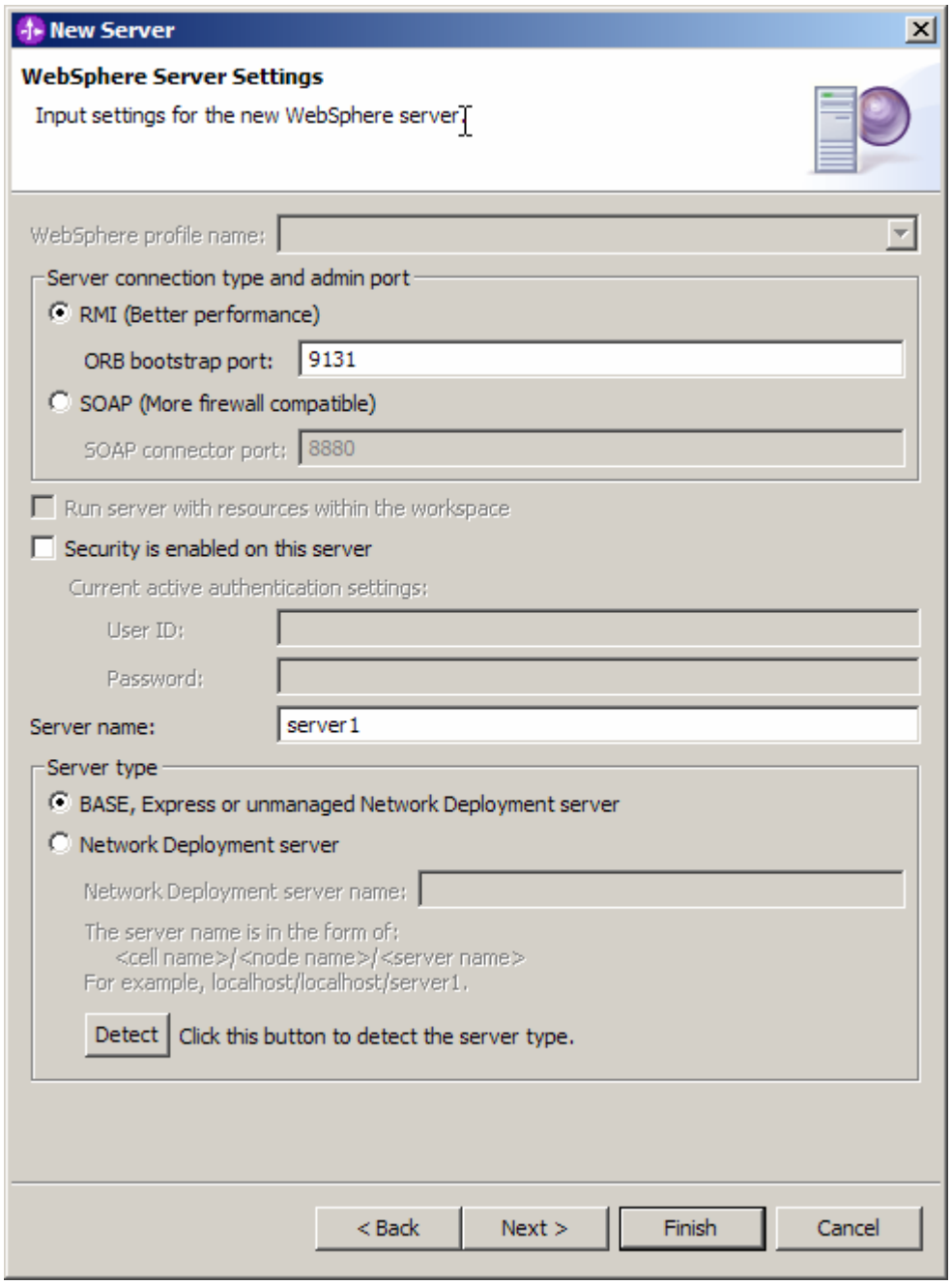
__ c. Specify hostname to the remote server, **<HOSTNAME>**

__ d. Ensure that '**WebSphere ESB 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

____ 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 cl1sr01 open for e-business; process id is 0000012000000002
```

This page is left intentionally blank.