

# WebSphere Enterprise Service Bus lab

## Dynamic endpoints

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 a mediation module .....	8
Part 3: Generate mediation flow implementation for the mediation.....	18
Part 4: Test dynamic end points .....	30
Part 5: Save the work and clean up server .....	35
What you did in this exercise .....	36
Solution instructions .....	37
Task: Adding remote server to WebSphere Integration Developer test environment .....	38

## What this exercise is about

The objective of this lab is to provide an understanding of how to create a Mediation Flow that will set a dynamic endpoint address in a Service Message Object (SMO) and invoke a service using the address.

## Lab requirements

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

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

## What you should be able to do

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

- Import the project interchange file into the WebSphere Integration Developer V6.1 development environment
- Create and edit a mediation module and mediation flow
- Navigate the Properties View for mediation information
- Generate implementation and binding from the development environment
- Work with the Mediation Flow Editor and build XSL Transformation request and response flows
- Test by running a JSP on the WebSphere Enterprise Service Bus V6.1 server with the Dynamic End Point property enabled (default) for the Callout
- Test by running a JSP on the WebSphere Enterprise Service Bus V6.1 server with the Dynamic End Point property disabled for the Callout

---

## Introduction

The concept of Dynamic End Points provide the runtime support, in fact it provides an enhancement to the WebSphere Enterprise Service Bus (and WebSphere Process Server) runtime to:

- Allow the mediation programming model to select (or to influence the selection of) service endpoints at the runtime
- Allow the selection at runtime of a service endpoint that has not been predefined in the mediation flow

With the support provided by Dynamic End Points, the mediation flow designer can use a Callout Node in the flow in a dynamic way. For a dynamic callout, the associated reference provided on the component need not be wired to an import. At runtime, the dynamic callout retrieves an element from the Service Message Object (SMO) which provides the endpoint Address to be used by the callout as the service endpoint.

The mediation flow primitives that are wired in the flow can provide logic which affects the contents of the endpoint Address held in the SMO, and the logic which can control the selection of the callout to be used. For example, a Database Lookup primitive might supply an Endpoint Reference (EPR) used to set endpoint information into the transient context of the SMO, and a Message Filter primitive might then route the request (based on the characteristics of the endpoint information) through XSLT Transformation primitives which can tailor the final endpoint Address set in the request SMO. The SMO is then passed by way of a dynamic callout to invoke the required external service.

The runtime implementation of Dynamic End Points is based on the SCA runtime support for Dynamic References. The support offered by Dynamic End Points is therefore consistent with SCA Dynamic Reference support and is subject to the same limitations. For example, there is no direct support for dynamic invocation of a service using JMS Bindings, although a predefined SCA import which specifies JMS Bindings might be selected at runtime.

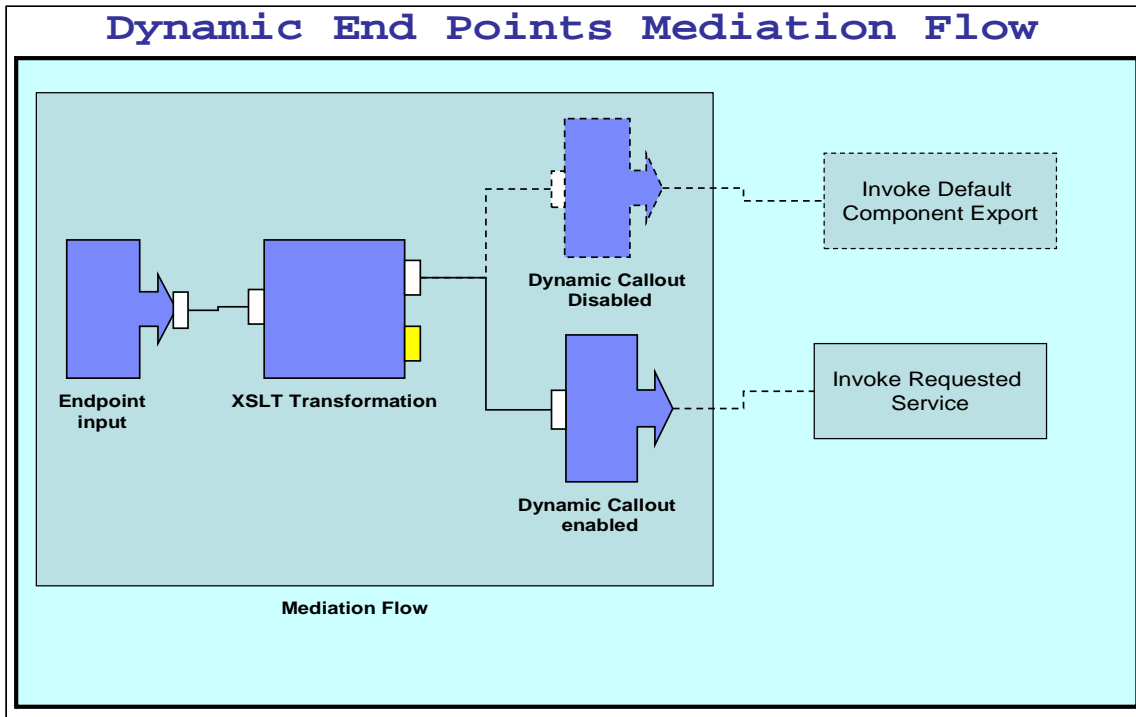
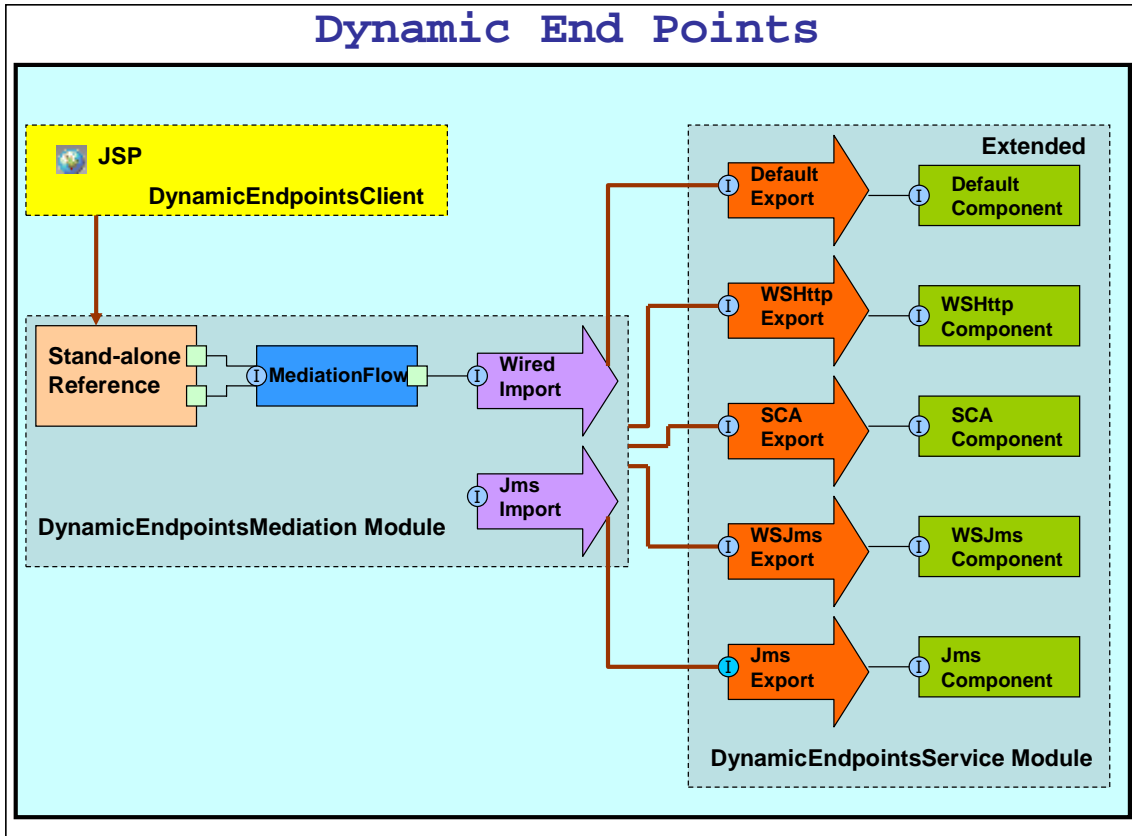
For use of the Dynamic End Points, an endpoint Address representation is carried in the Service Message Object (SMO) that is passed to the Callout node in a mediation flow. A new property of a Callout node controls whether the Address held in the SMO header should be used to dynamically invoke the service.

If the Callout property specifies that the dynamic Address should be used, and the SMO header at runtime holds an endpoint Address, then this Address is used by the runtime when invoking the service. The default endpoint (if any) is overridden by the Address located in the header of the SMO.

In this lab, you construct a Mediation Flow that includes a Callout to a service. Within the body of the message received by the Mediation Flow, there is an endpoint address. An XSLT primitive in the Mediation Flow moves the endpoint address from the SMO body to the dynamic endpoint address field in the SMO header.

The actual endpoint used by the Callout is either:

- The dynamic endpoint address in the SMO header (if Callout property **“Use dynamic endpoint”** enabled)
- The endpoint specified by the wired import (if Callout property **“Use dynamic endpoint”** disabled)



## 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\DynamicEndpoints	/tmp/Labfiles61/WESB/DynamicEndpoints
<WORKSPACE>	C:\Labfiles61\WESB\DynamicEndpoints\workspace	/tmp/Labfiles61/WESB/DynamicEndpoints /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\ be 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:

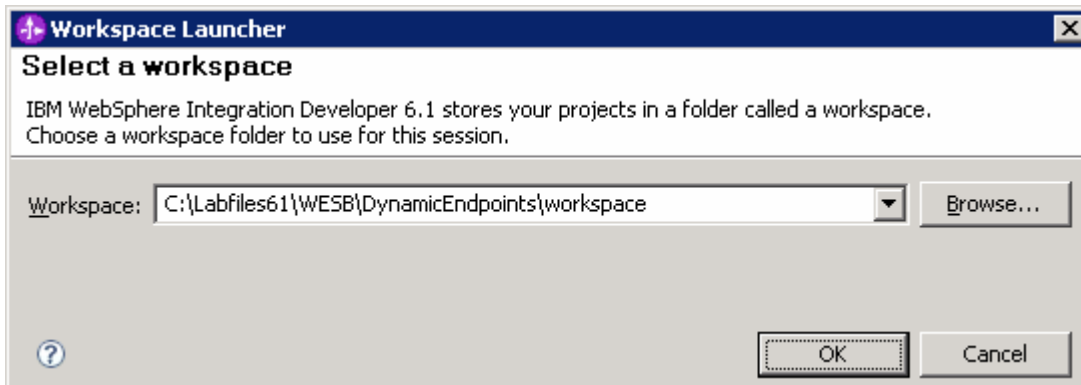
Reference variable	Example: Remote Windows test server location	Example: Remote z/OS® test server location	Input your values for the remote location of the test server
<SERVER_NAME>	server1	sssr011	
<WAS_HOME>	C:\Program Files\IBM\WebSphere\AppServer	/etc/sscell/AppServer	
<HOSTNAME>	localhost	mvsxxx.rtp.raleigh.ibm.com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

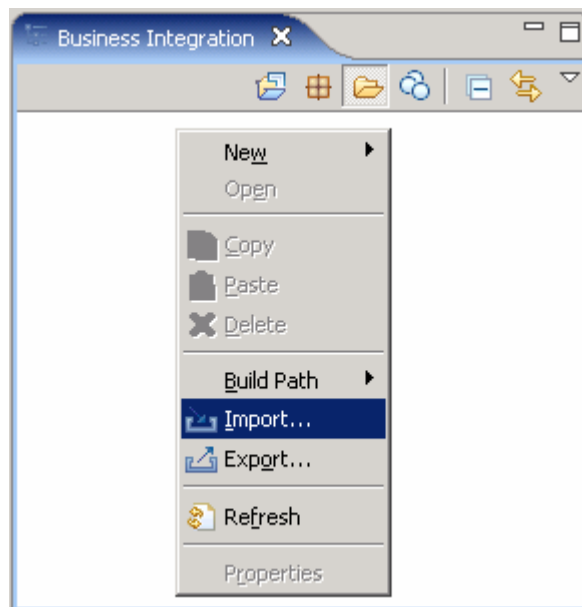
## Part 1: Prepare environment for the lab

In this section of the lab, all the projects that are part of **WESB\_DynamicEndpoints\_PI.zip** project interchange file are imported into a new workspace. Remember this is the sample SCA application to which the ESB specific mediation is added as the lab progresses.

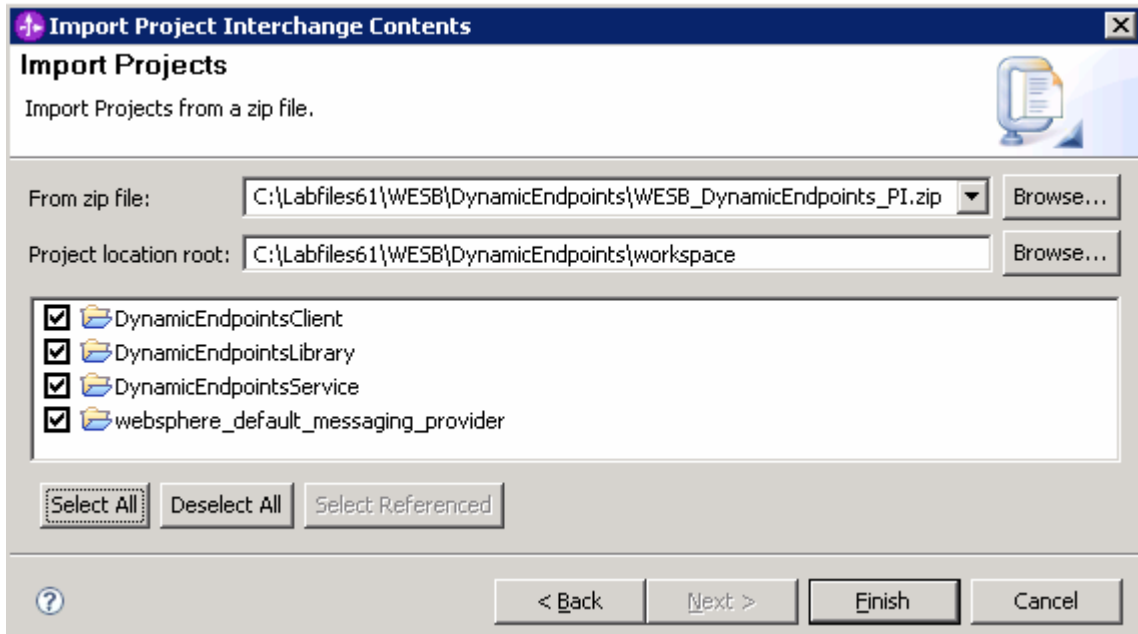
- \_\_\_ 1. Start WebSphere Integration Developer V6.1 with the workspace location as **C:\LabFiles61\WESB\DynamicEndpoints\workspace**



- \_\_\_ 2. On the welcome screen, click the curved arrow at the top right to '**Go to the business integration perspective**' (  ), to close the 'Welcome' screen.
- \_\_\_ 3. Import the Project Interchange file, **WESB\_DynamicEndpoints\_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

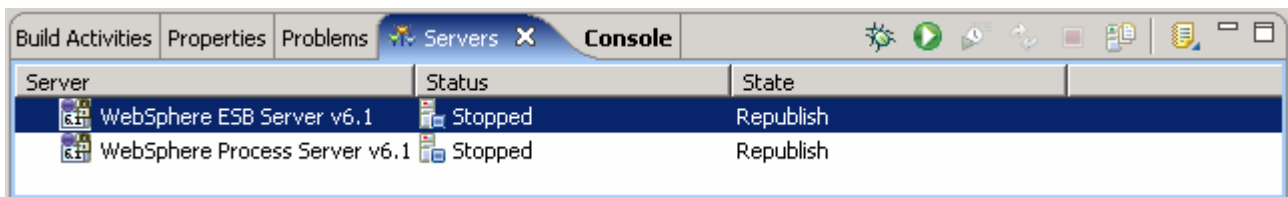


- \_\_\_ c. From the **Import** dialog, expand '**Other**' and select '**Project Interchange**'
- \_\_\_ d. Click **Next**.
- \_\_\_ e. Click the **Browse** button for '**From zip file**' and navigate to <LAB\_FILES>/WESB\_DynamicEndpoints\_Pi.zip and hit **Open**.



- \_\_\_ f. Click the **Select All** button to ensure all projects listed are selected.
- \_\_\_ g. Click the **Finish** button (projects are imported and auto-build is run)

\_\_\_ 4. Verify that the WebSphere ESB Server V6.1 is listed in the **Servers** view (bottom right window)

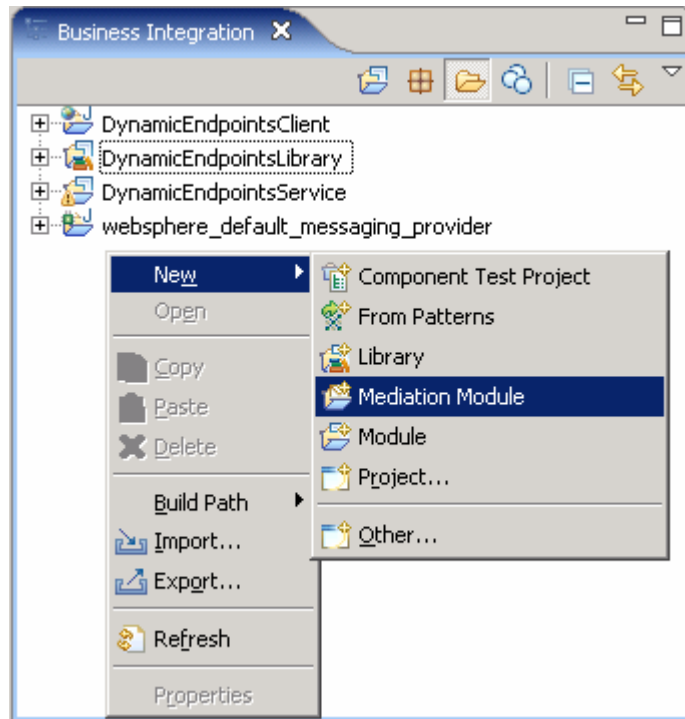


## Part 2: Create a mediation module

In this section of the lab, a new mediation module is created. There can only be one mediation module for each deployable project.

\_\_\_ 1. To create the mediation module, complete these steps:

\_\_\_ a. In the Business Integration view, right-click and select **New → Mediation Module** from the pop-up menu.



\_\_\_ b. In the 'New Mediation Module' panel, enter these parameters:

- Module Name: **DynamicEndpointsMediation**
- Ensure the target runtime is '**WebSphere ESB Server v6.1**'
- Select the check box for '**Create mediation component**'
- Change the name of the module from '**DynamicEndpointsMediation**' to '**Mediation1**'



**New Mediation Module**

**Mediation module**

Create a new mediation module. A mediation module is a project that is used for development, version management, organizing resources, and deploying to the

Module name:

Use default location

Location:

Target runtime:

Create mediation component

Name:

Mediation modules can be deployed and run on WebSphere Enterprise Service Bus or WebSphere Process Server. They contain flows, which link together operations for modifying and routing messages between service consumers and service providers.

\_\_ c. Click **Next**

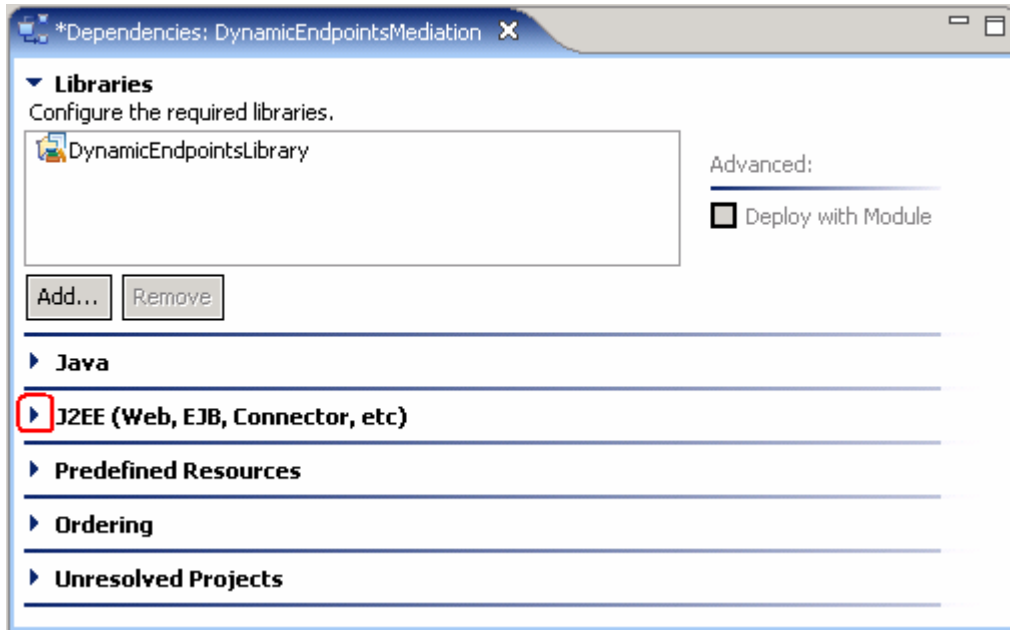
\_\_ d. In the 'Select Required Libraries' panel, select **DynamicEndpointsLibrary**

\_\_ e. Click **Finish**

\_\_ f. A mediation module called **DynamicEndpointsMediation** is created. Also a mediation flow component called **Mediation1** is created in the module's assembly diagram.

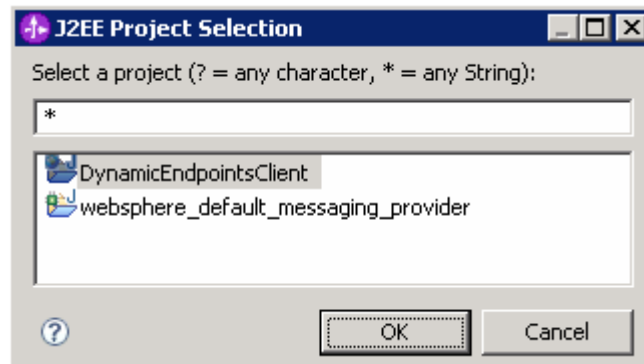
\_\_\_ 2. Add **DynamicEndpointsClient** Web project and **websphere\_default\_messaging\_provider** connector project to **DynamicEndpointsMediation** as a J2EE dependency projects

\_\_ a. In the Business Integration view, expand **DynamicEndpointsMediation** module (if not already expanded) and double-click on Dependencies (  Dependencies ) to open the Dependency Editor



\_\_\_ b. In the Dependency Editor, expand **J2EE** and click the **Add** button to add **DynamicEndpointsClient** and **websphere\_default\_messaging\_provider** as J2EE dependent projects for the mediation module

\_\_\_ c. From the **J2EE Project Selection** window, select the projects listed one at a time and click **OK**

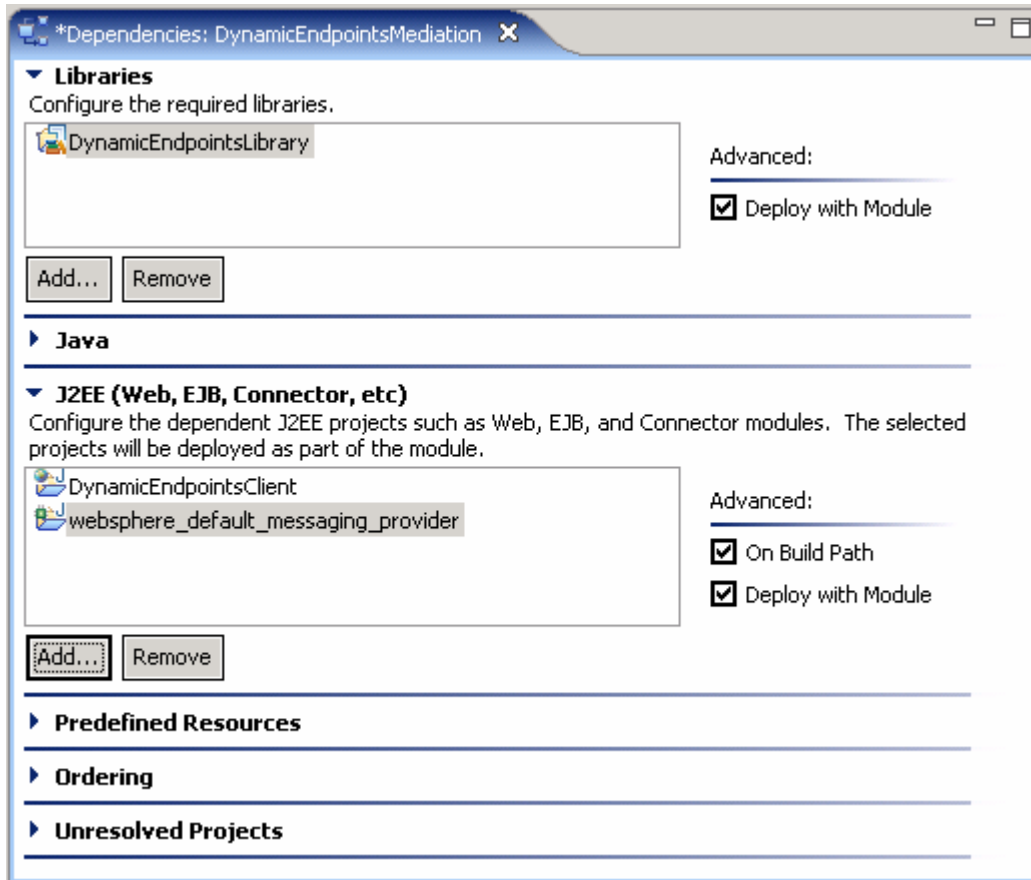


\_\_\_ d. Ensure that the check box next to **Deploy with Module** is selected by selecting each of the modules

---

**Note:** The required configured **DynamicEndpointsLibrary** was added as a dependent library while creating the mediation module.


---



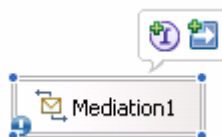
\_\_\_ e. Save the changes (**File → Save or Ctrl + S**) and close the Dependency Editor

\_\_\_ 3. Open the **DynamicEndpointsMediation** module assembly diagram in an Assembly Editor to visually compose the Mediation Module


\_\_\_ a. Open **DynamicEndpointsMediation** module assembly diagram

1) In the Business Integration view, expand the **DynamicEndpointsMediation** module (if not already expanded) and double-click on Assembly Diagram (  Assembly Diagram ) to open it with the assembly editor

2) Notice that a default mediation flow was created and is named as **Mediation1**

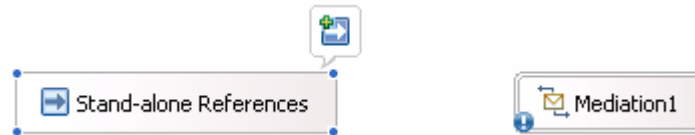



\_\_\_ b. Add a **Stand-alone Reference** to the Mediation in the Assembly Diagram

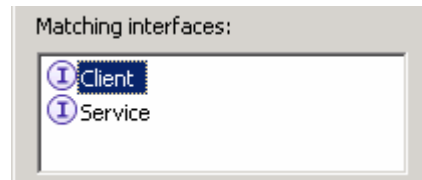
1) Add a **Stand-alone Reference** from the component palette tray (  References ) and drop it on the left side of **Mediation1**

a) Select the **Stand-alone Reference** icon (  ) from component palette tray

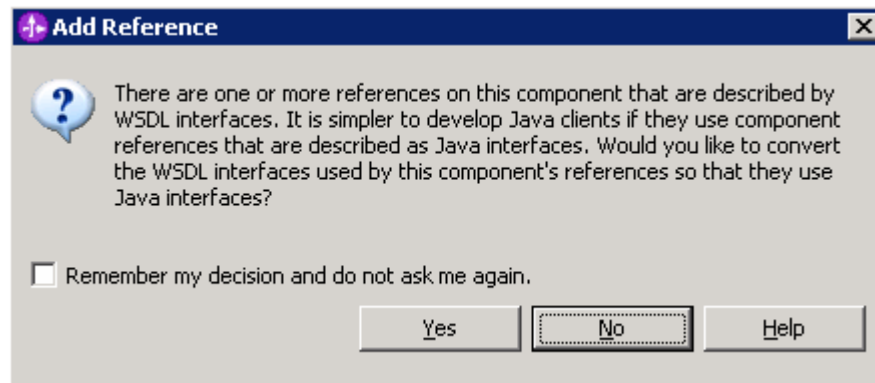
b) Click to it to the left side of **Mediation1**



- 2) Hover over the Stand-alone Reference or click on the Stand-alone Reference so the Add Reference icon appears. Click the icon (  ) to add a reference
- 3) From the **Add Reference** window, select **Client** (if not already selected) from the list of matching interfaces and accept the default Name





- 4) Click **OK**




- 5) Click **NO** to the Question on the pop-up dialog
- 6) The **Stand-alone Reference** must look as shown below:

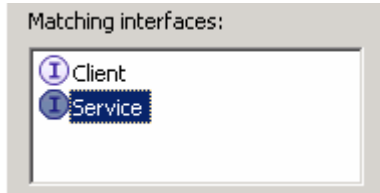


\_\_\_ c. Add an Import to the Mediation in the Assembly Diagram

- 1) Add an **Import** from the component palette tray (  ) and drop it on the right side of **Mediation1**
  - a) Click on **Import** icon (  ) from the component palette tray
  - b) Click to drop it to right side of **Mediation1**
- 2) Right click over the import and select **Rename** from the pop-up menu to change default import name from **Import1** to **WiredImport**





- 3) Hover over or click on **WiredImport** so the add interface icon appears. Click the icon (  ) to add an interface
- 4) Select **Service** from the list of matching interfaces




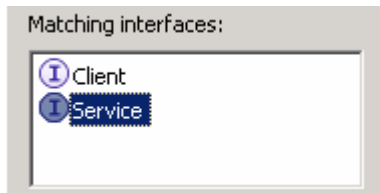
\_\_ d. Click **OK**

\_\_ e. Add another Import to Mediation in the Assembly Diagram

- 1) Add an **Import** from the component palette tray (  ) and drop it on the right side of **Mediation1**
  - a) Click on **Import** icon (  ) from component palette tray
  - b) Click to drop it to the right side of **Mediation1**
- 2) Right click over the import and select **Rename** from the pop-up menu to change the default import name from **Import1** to **JmsImport**



- 3) Hover over or click on the **JmsImport** so the add interface icon appears. Click the icon (  ) to add an interface
- 4) Select **Service** from the list of matching interfaces



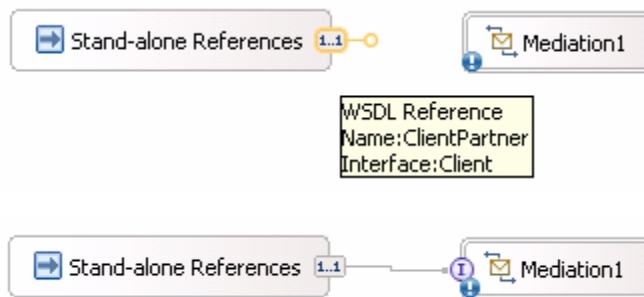
5) Click **OK**

6) The Assembly Diagram looks like the one shown below:



\_\_\_ f. Wire the components together

- 1) Click on **Stand-alone Reference** and ensure the **ClientPartner Reference** node is selected and drag a wire and connect to **Mediation1**. Click **OK** to any pop-up dialog



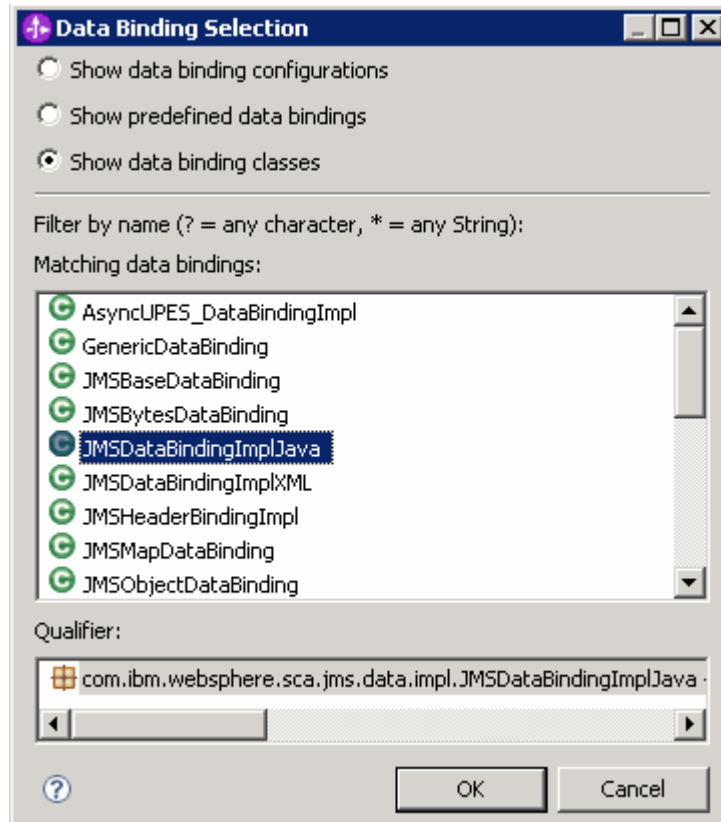
- 2) Click on **Mediation1** and drag a wire and connect to **WiredImport**. Click **OK** to any pop-up dialog



- 3) Leave the **JmsImport** component alone

\_\_\_ g. Generate bindings for the Imports

- 1) Right-click on **WiredImport** and select **Generate Binding → SCA Binding** from the pop-up menu
- 2) Right-click on **JmsImport** and select **Generate Binding → Message Binding → JMS Binding** from the pop-up menu.
- 3) In the **JMS Import Binding** panel, click the **Browse** button for '**Data binding configuration**' under the '**Data Format**' section. The '**Data Binding Selection**' panel is launched
  - a) In the '**Data Binding Selection**' panel, select the radio button for '**Show data binding classes**'. This action should list the matching data bindings



b) Select '**JMSTDataBindingImplJava**' from the matching data bindings and then click **OK**

4) Accept the defaults for the remaining parameters from the '**JMS Import Binding**' panel

5) Click **OK**

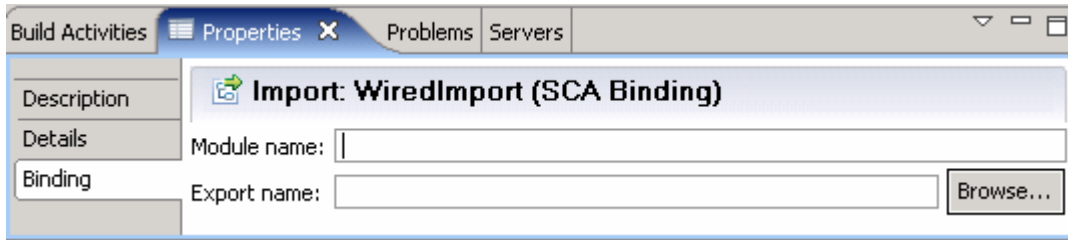
6) This is how the Assembly Diagram looks after generating the bindings:



\_\_\_ h. Update Binding for the **WiredImport**

1) Select the **WiredImport** component in the Assembly Diagram

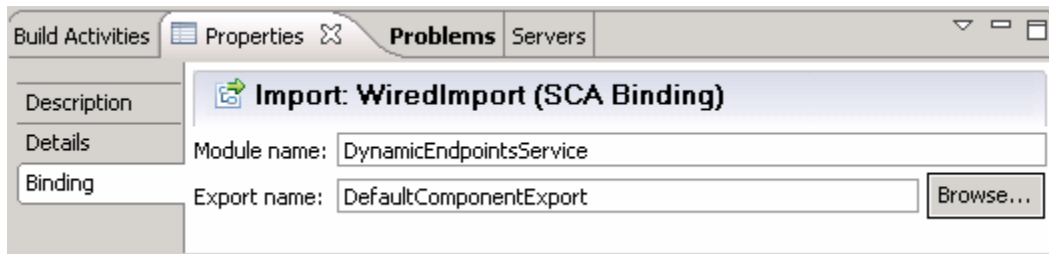
2) In the Properties view (bottom right window) select the **Binding** tab



- 3) Click the browse button next to **Export name** field and select **DefaultComponentExport**



- 4) Click **OK**
- 5) Set the **Module name** to **DynamicEndpointsService** and the **Export name** to **DefaultComponentExport**

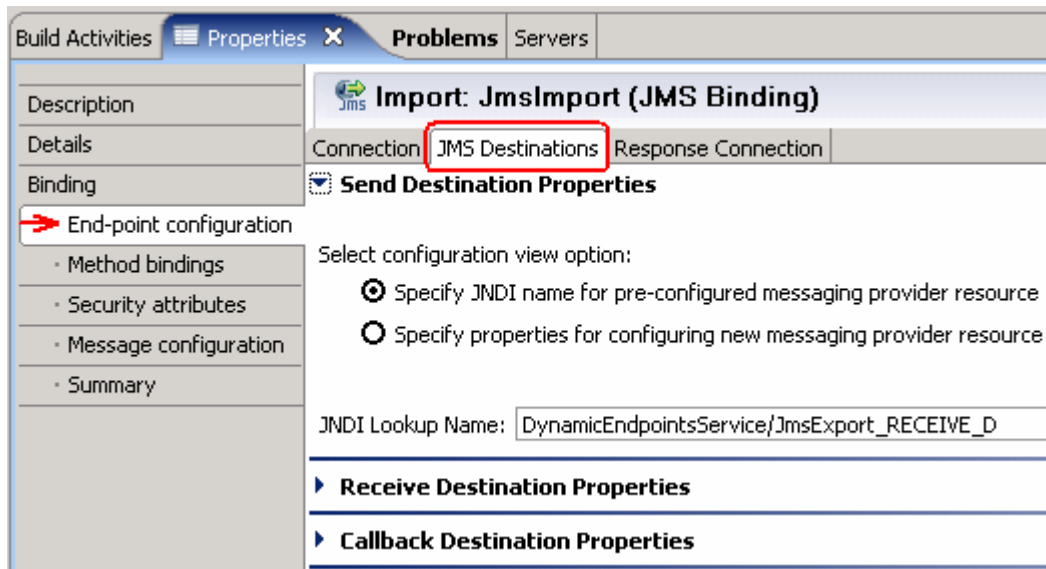


- 6) Save all work (**File** → **Save All** or **Ctrl + Shift + S**)

\_\_\_ i. Configure End-point Configuration binding for the **JmsImport**

- 1) Select the **JmsImport** component in the Assembly Diagram
- 2) In the Properties view select **Binding tab** → **End-point configuration**
- 3) Select the **JMS Destinations** tab
- 4) Expand **Send Destination Properties**
- 5) Select the **Specify JNDI Name for pre-configured messaging provider resource** for the configuration view option
- 6) Enter **DynamicEndpointsService/JmsExport\_RECEIVE\_D** for JNDI Lookup name



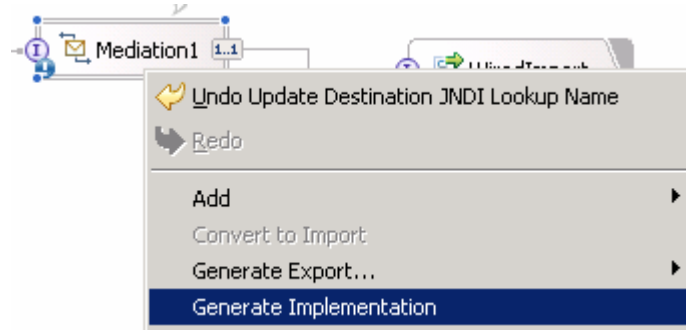


7) Save all work (**File** → **Save All** or **Ctrl + Shift + S**)

## Part 3: Generate mediation flow implementation for the mediation

In this section of the lab, the Mediation component is selected for generating the implementation and such an action opens the implemented Module in a Mediation Flow Editor. On connecting the source and target operations for the Operations Connections view, the mediation primitive, that is, an XSL Transformation primitive is added to the Request and Response flows.

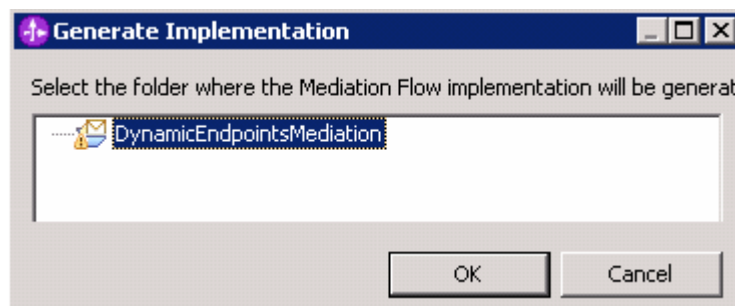
- \_\_\_ 1. To open the Mediation Flow Editor for the mediation module, **Mediation1**, follow the steps below:
  - \_\_\_ a. Open the mediation module in the assembly editor
  - \_\_\_ b. In the Mediation Module's Assemble diagram, right click on **Mediation1** and choose **Generate Implementation** from the pop-up menu



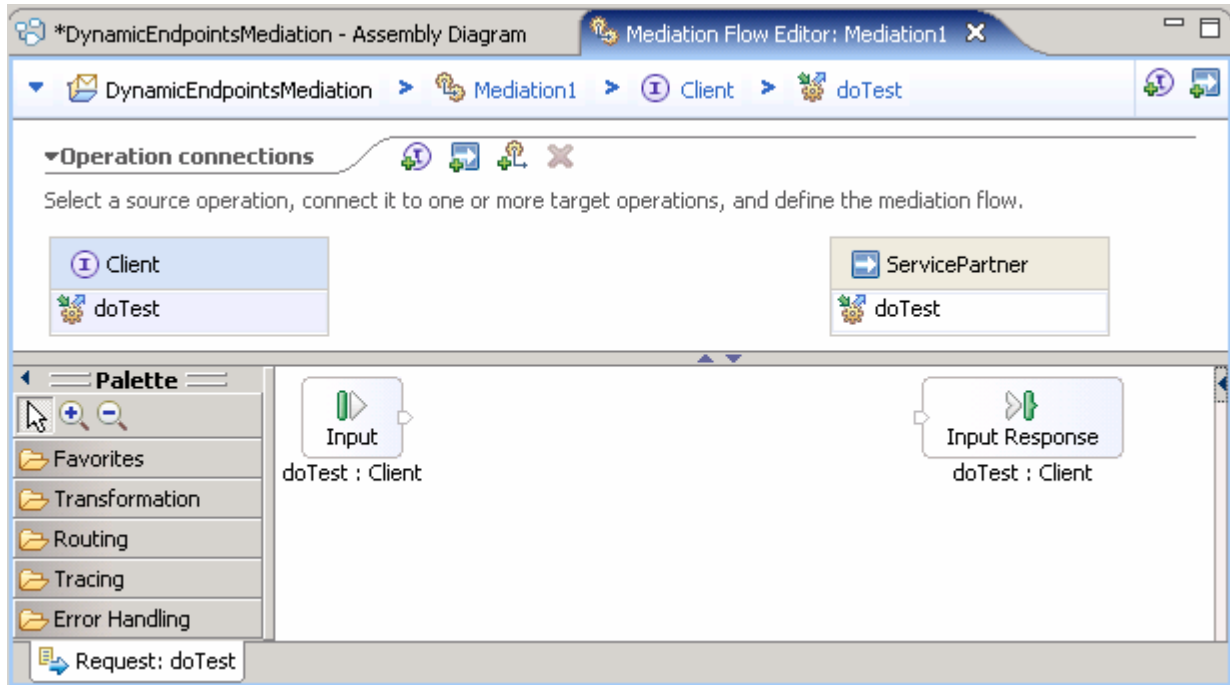
**Alternative 1:** Double-click on the mediation flow component, **Mediation1**. If an implementation exists, it is opened in the mediation flow editor. If the component is not implemented, a dialog will pop up asking to confirm for implementation. Click **Yes**, and select the target mediation module, **DynamicEndpointsMediation**. Click **Finish**

**Alternative 2:** In the Business Integration View, expand the mediation module, **DynamicEndpointsMediation** and then expand the **Flows** category. Select the mediation flow, **Mediation1**, right-click and select **Open**

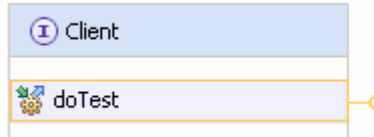
- \_\_\_ c. Select the target mediation module, **DynamicEndpointsMediation** from the Generate Implementation dialog



- \_\_\_ d. Click **OK**
- \_\_\_ e. The **Mediation Flow Editor** will open after generating implementation for the mediation module, **Mediation1**



- \_\_\_ 2. Connect the **source to target operations** in Operation Connections view
  - \_\_\_ a. Click anywhere on the source operation, **Client/doTest** on the left side of the Operation Connections view



- \_\_\_ b. Drag to the target operation, **ServicePartner/doTest** on the right side of the Operation Connections view and release the mouse click

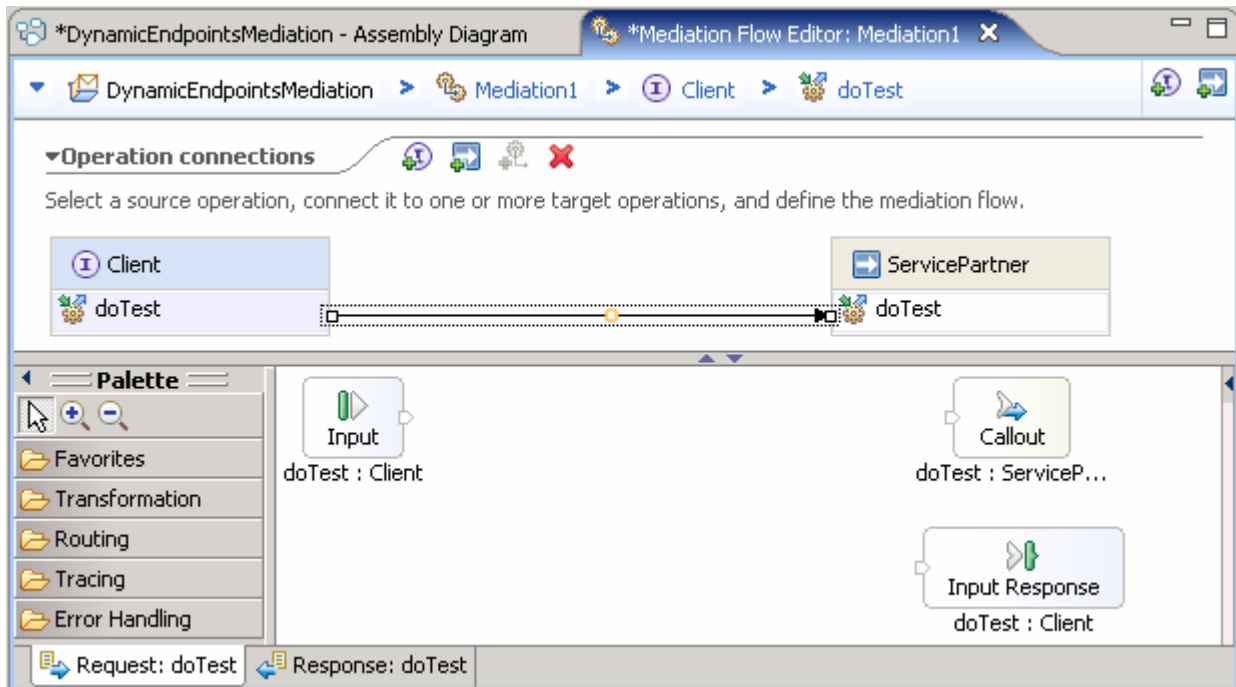
---

**Alternative:** Right-click on the source component, **Client/doTest** and select **Create an operation connection** and then click on the target operation, **ServicePartner/doTest**

---



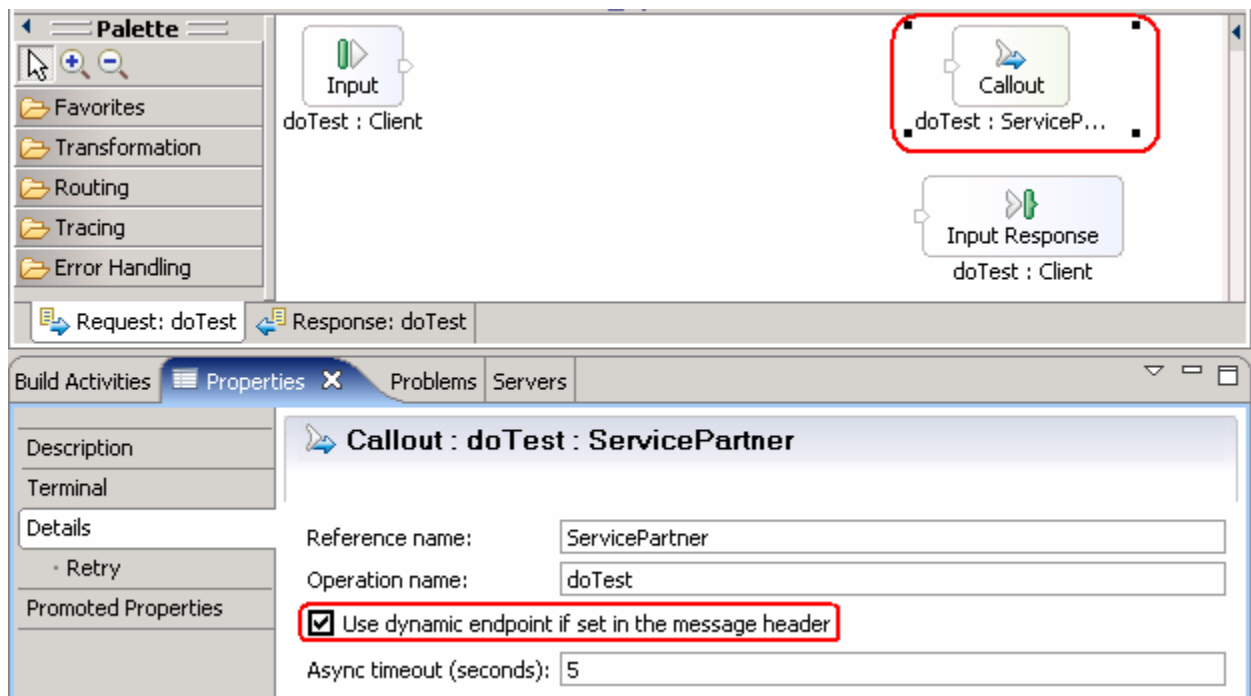
- \_\_\_ 3. Click on the black line (wire) to view the Mediation Flow View and ensure the **Request** tab is selected to build the Request flow as shown below



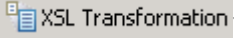
4. Ensure that **Callout** property is enabled

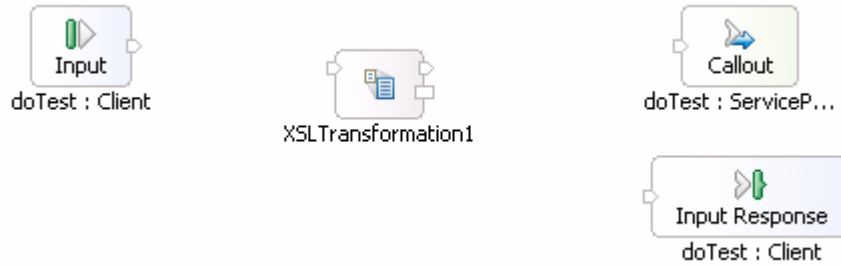
- a. In the Mediation Flow Editor (Request flow - middle window (doTest:ServicePartner)), select the **Callout node** and select the **Details** tab under its Properties view (bottom window) to ensure the '**Use dynamic endpoint if set in the message header**' is selected

**Note:** By default the 'Use dynamic endpoint if set in the message header' property is enabled.

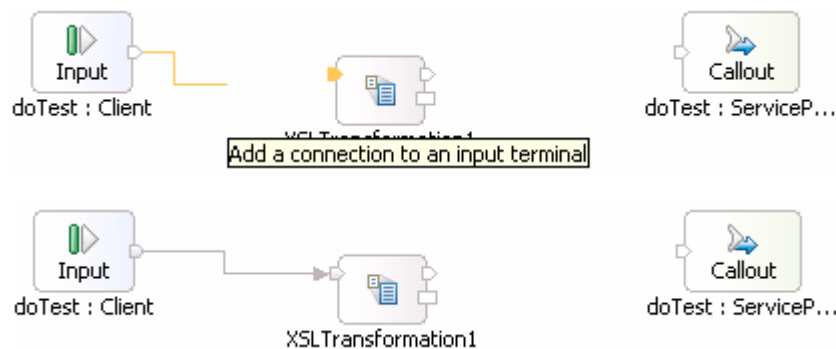


5. Add a **XSL Transformation** primitive to the Request Mediation Flow diagram

- \_\_\_ a. In the **Mediation Flow Editor**(middle), click on **XSL Transformation** icon (  ) to select the XSL Transformation primitive from the transformation palette tray on left side and drop it into the canvas between the Input Request node and the Callout Request Node as shown below:



- \_\_\_ b. Hover the mouse over **Input node's** output terminal and drag the handle that appears to the input terminal of the XSL Transformation primitive, **XSL Transformation1**

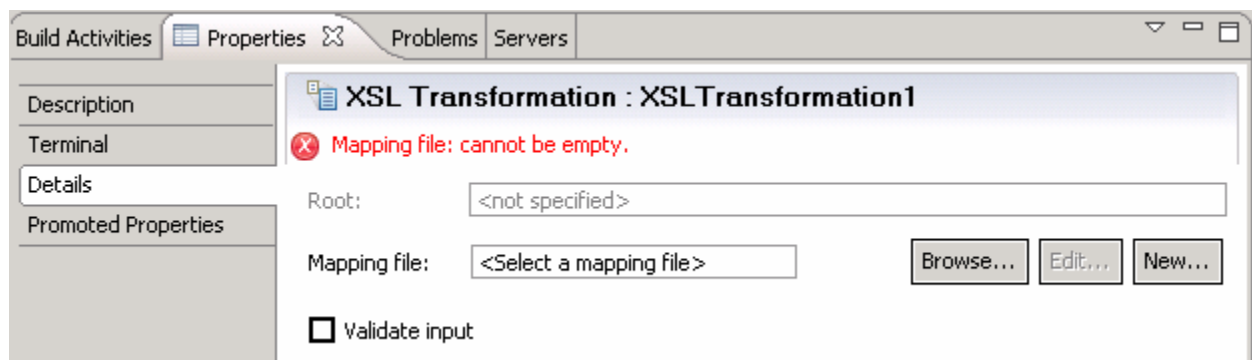


- \_\_\_ c. Hover the mouse over XSL Transformation primitive, **XSL Transformation1's** output terminal and drag the handle that appears to the input terminal of the **Callout node**



- \_\_\_ 6. Set the Properties for the XSL Transformation primitive, **XSL Transformation1**

- \_\_\_ a. In the Mediation flow editor, select the XSL Transformation primitive, **XSL Transformation1** and choose the **Details** tab under its properties view



\_\_ b. Click the 'New...' button on the Mapping File row to launch the 'XML Mapping' wizard

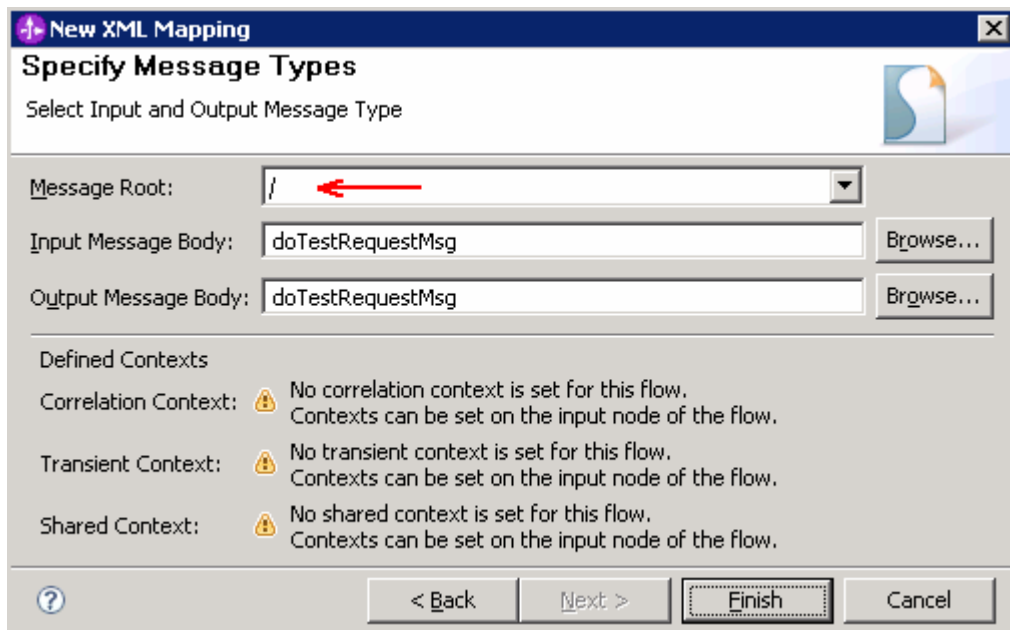
1) Enter these parameters in the 'XML Mapping' panel

- Folder : **xsIt** (default)
- Name : **XSLTransformation1\_req\_1** (default)

2) Click **Next**

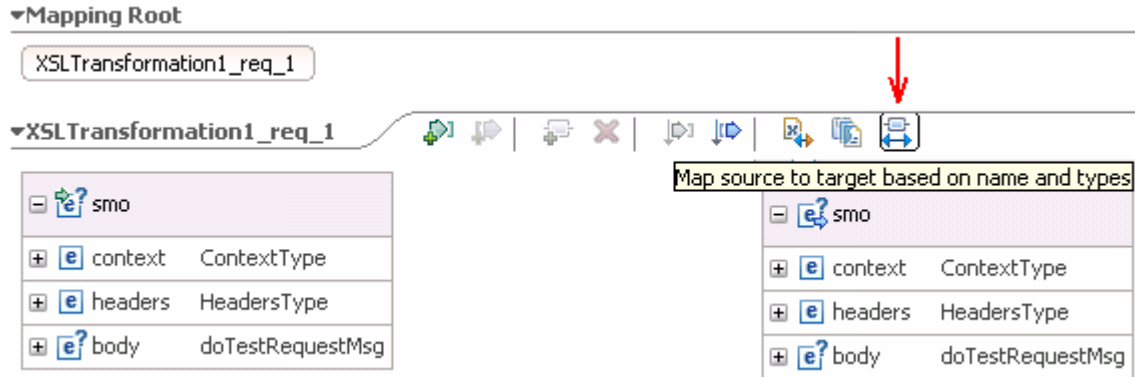
3) In the 'Specify Message Types' panel, enter these parameters:

- Select / (root) for the Message Root field
- Since you wired the Input node to XSLTransformation1, the input message body is already assigned as **doTestRequestMsg**
- Since you wired the XSLTransformation1 to Callout node, the output message body is already assigned as **doTestRequestMsg**



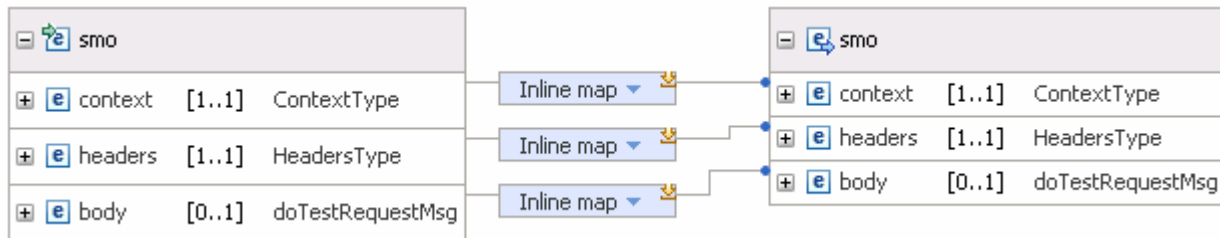
\_\_ c. Click **Finish**

\_\_ d. The XSL Transformation mapping editor opens



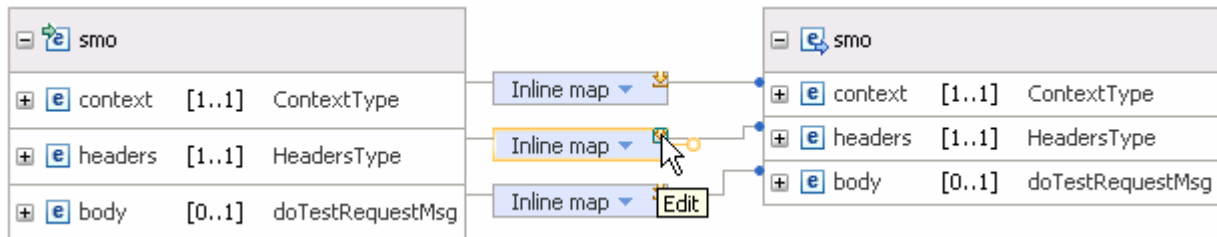
\_\_\_ e. Click the toolbar icon (↔) **'Map source to target based on name and types'** which will map the context and headers using inline maps as shown below:

**Note:** This creates a nested set of inline maps for all matching elements in the source and target. In this case, nested inline maps are created for the context, headers and body which have the same elements in source and target.

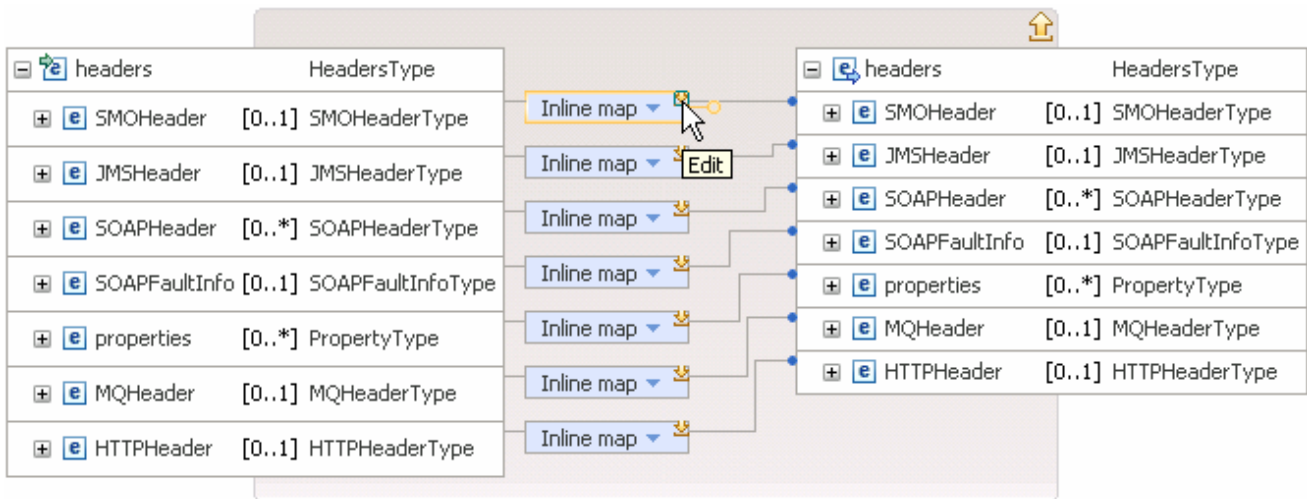


\_\_\_ f. Navigate down into the headers/SMOHeader inline map and remove the SMOHeader/Target inline map

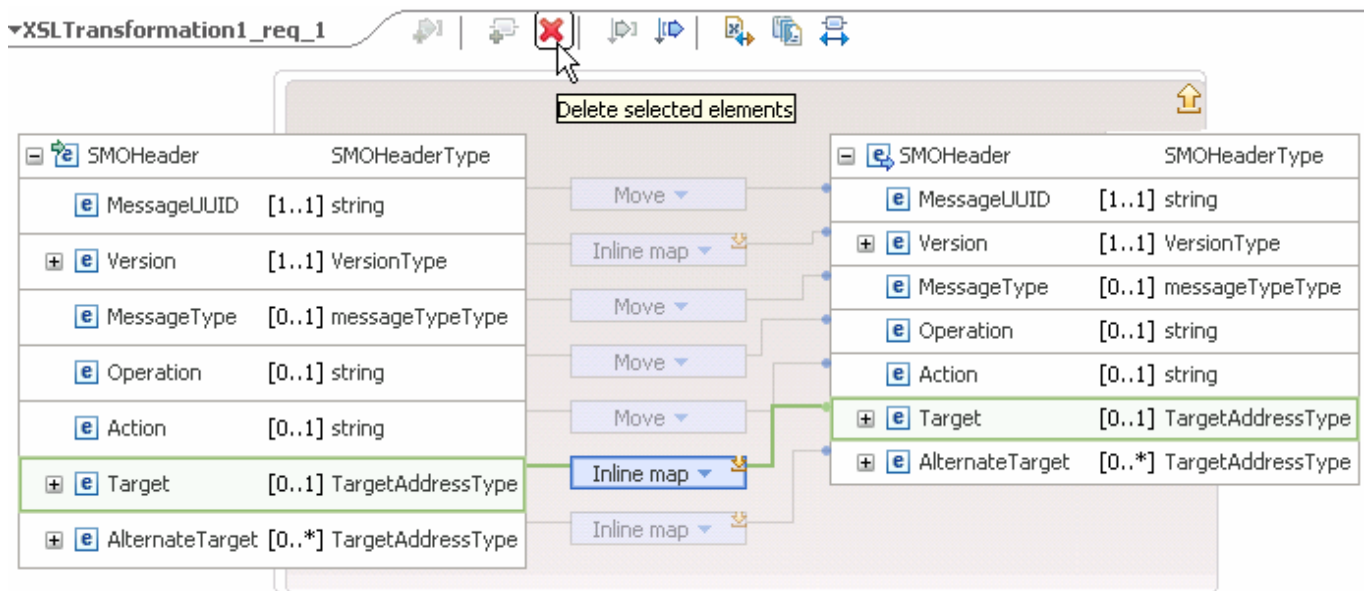
1) Click on **Edit** (🔧) icon on the Inline map connecting **headers** attributes



2) This opens all the transformations defined under **'headers'**. Click on **Edit** (🔧) icon on the Inline map connecting **SMOHeader** attributes:



3) This opens all the transformations defined under '**SMOHeader**'. Select the 'Inline Map' connecting the 'Target' attributes and then click '**Delete selected elements**' icon (✖) from the tool bar



4) Click on '**Up a level**' (⬆) icon which will bring you one level up to the headers mapping

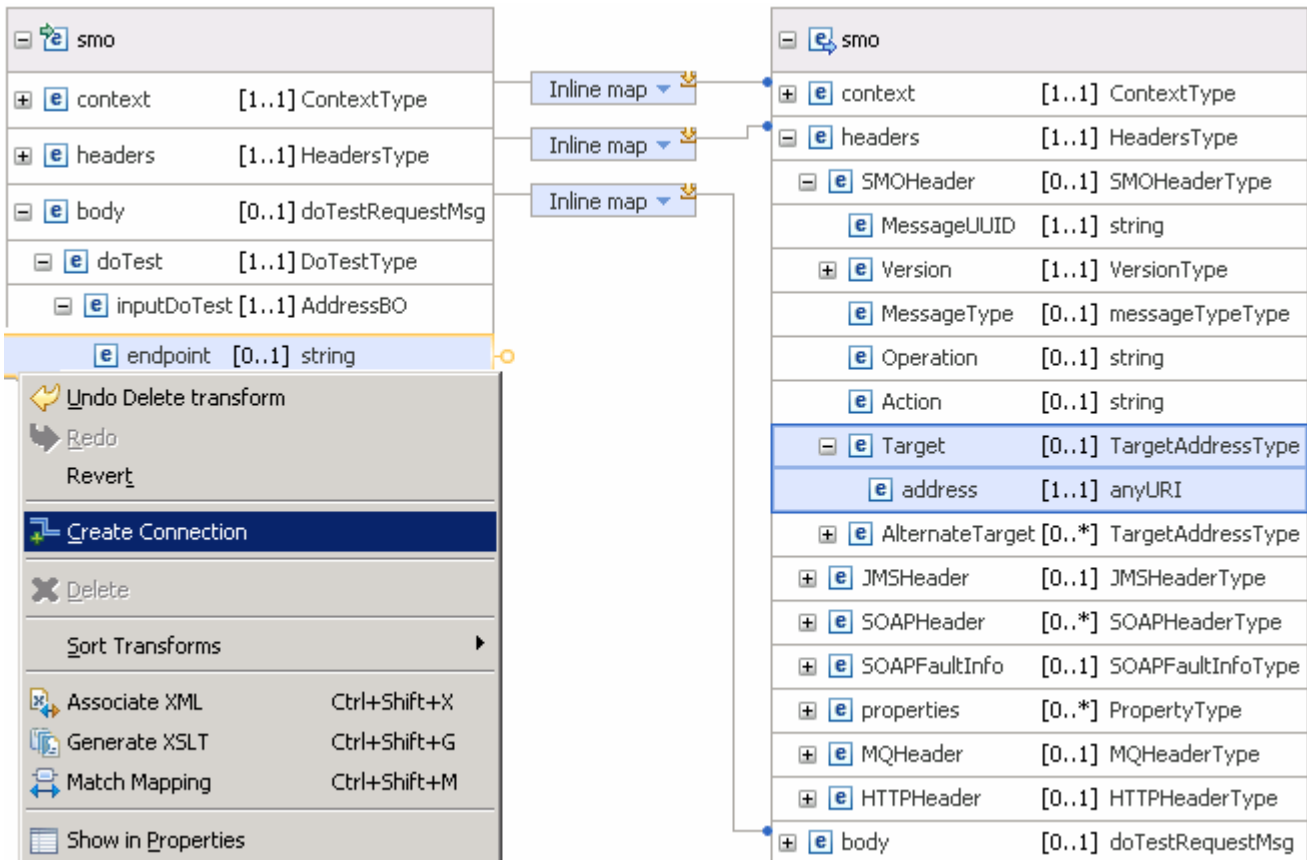


5) Click on '**Up a level**' (⬆) icon one more time which will now bring you back the **smo** mapping level

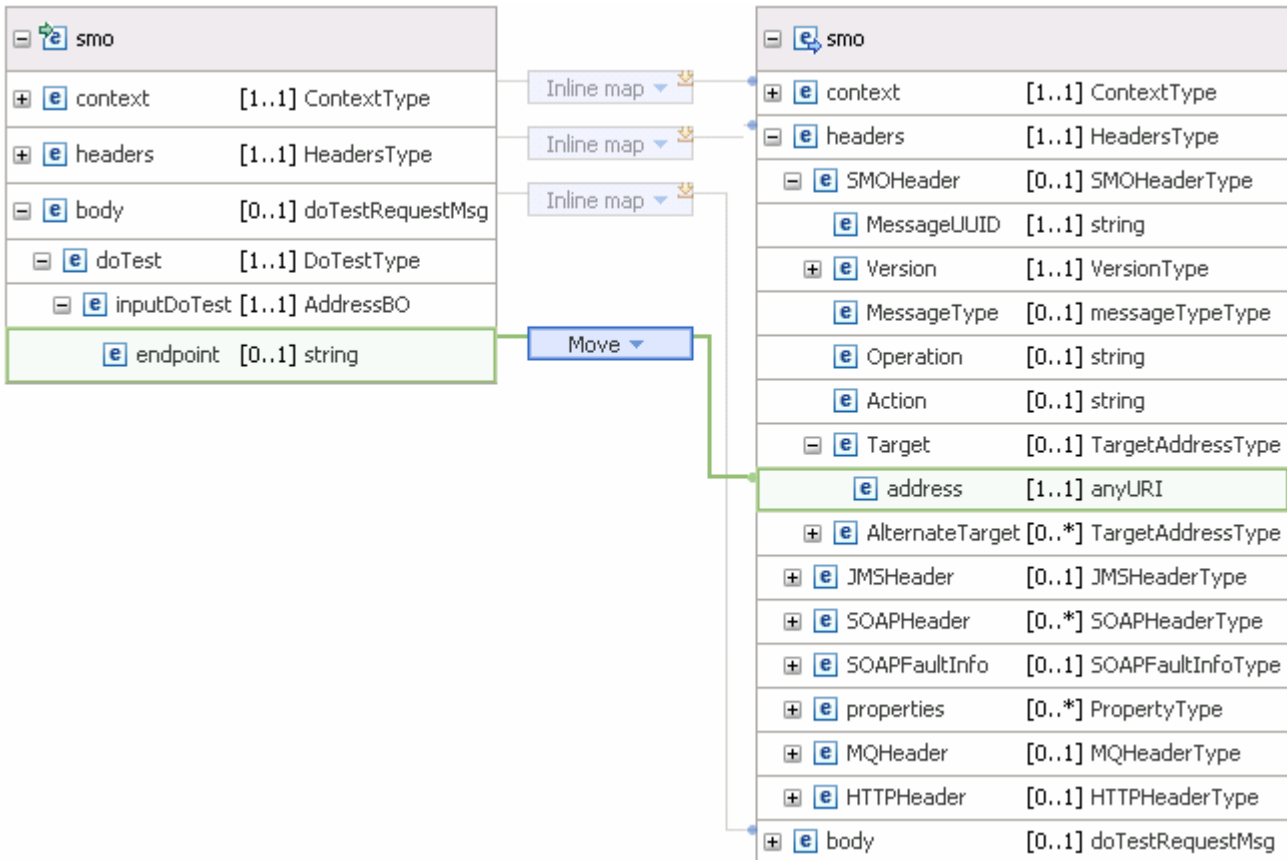
\_\_\_ g. Move **body/doTest/inputDoTest/endpoint** to **headers/SMOHeader/Target/address**



- 1) Expand **body** → **doTest** → **inputDoTest** → **endpoint** on the source side and **headers** → **SMOHeader** → **Target** → **address** on the target side as shown below:



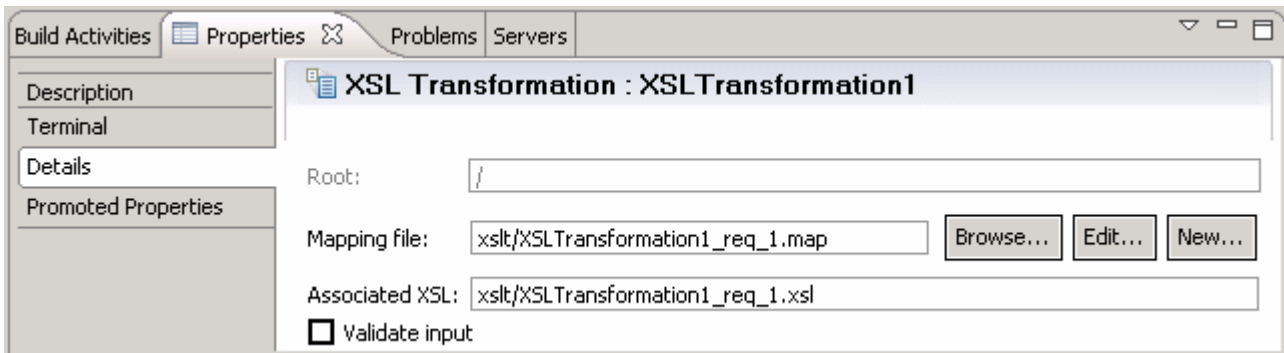
- 2) Right click over the 'endpoint' attribute on the source side and select 'Create Connection' from the pop-up menu as shown above. The connection wire appears
- 3) Drag the wire to 'address' attribute on the target side as shown below:



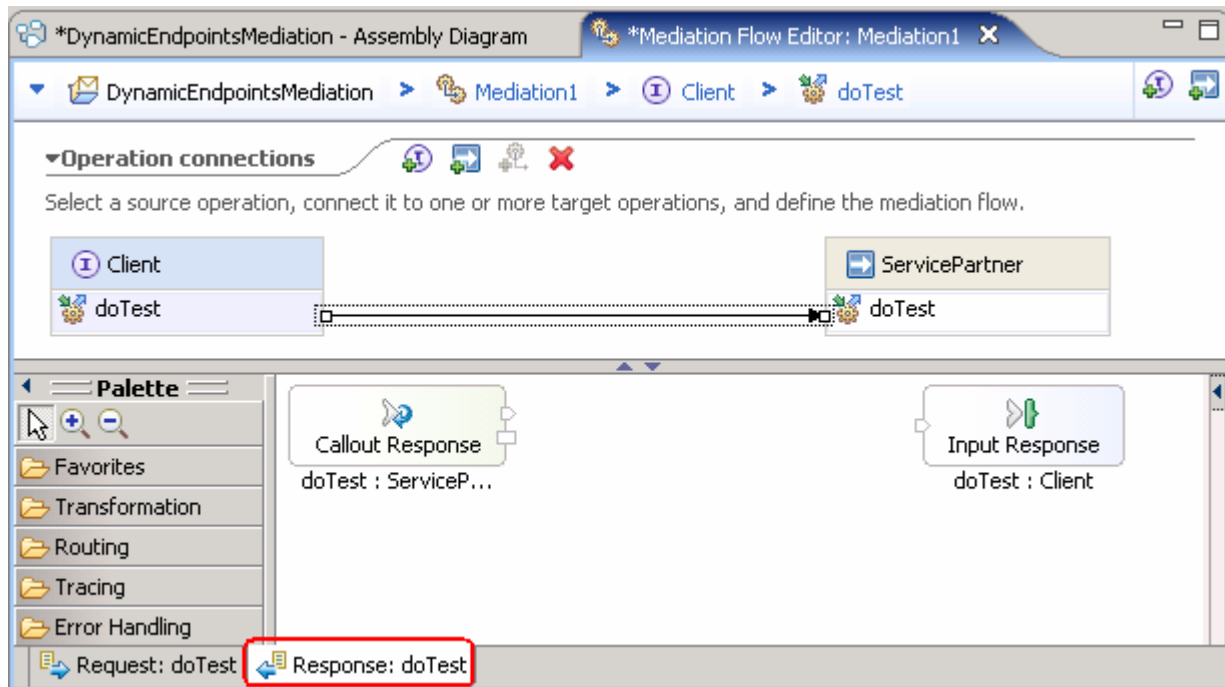
\_\_ h. Save all work (**File → Save All** or **Ctrl + Shift + S**)

\_\_ i. Close the XSL transformation editor

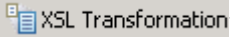
\_\_ j. On the Mediation Flow Editor, the XSL Transformation Binding must look as shown below:



\_\_\_ 7. Click on the black line (wire) to view the Mediation Flow View and select the **Response** tab to build the Response flow as shown below:



8. Add **XSL Transformation** primitive to Response Mediation Flow diagram

- a. In the **Mediation Flow Editor** (middle), click on **XSL Transformation** icon (  ) to select the XSL Transformation primitive from the transformation palette tray on the left side of view and drop it into the canvas between the Callout Response node and the Input Response Node



- b. Hover the mouse over **Callout Response** node's output terminal and drag the handle that appears to the input terminal of the XSL Transformation primitive, **XSL Transformation1**

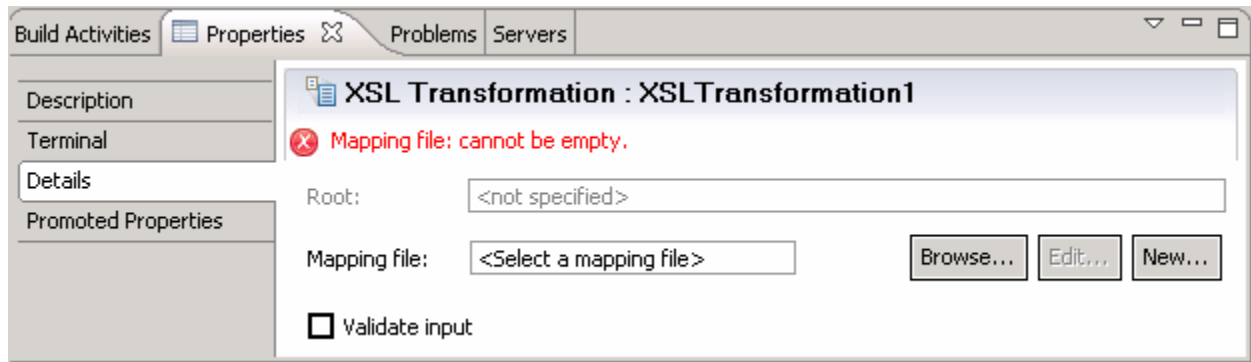


- c. Hover the mouse over XSL Transformation primitive, **XSL Transformation1**'s output terminal and drag the handle that appears to the input terminal of the **Input Response node**



9. Set the Properties for the XSL Transformation primitive, **XSL Transformation1**

- a. In the Response Mediation flow editor, select the XSL Transformation primitive, **XSL Transformation1** and choose **Details** under its properties view



\_\_ b. Click the **'New...'** button on the Mapping File row to launch the **'XML Mapping'** wizard

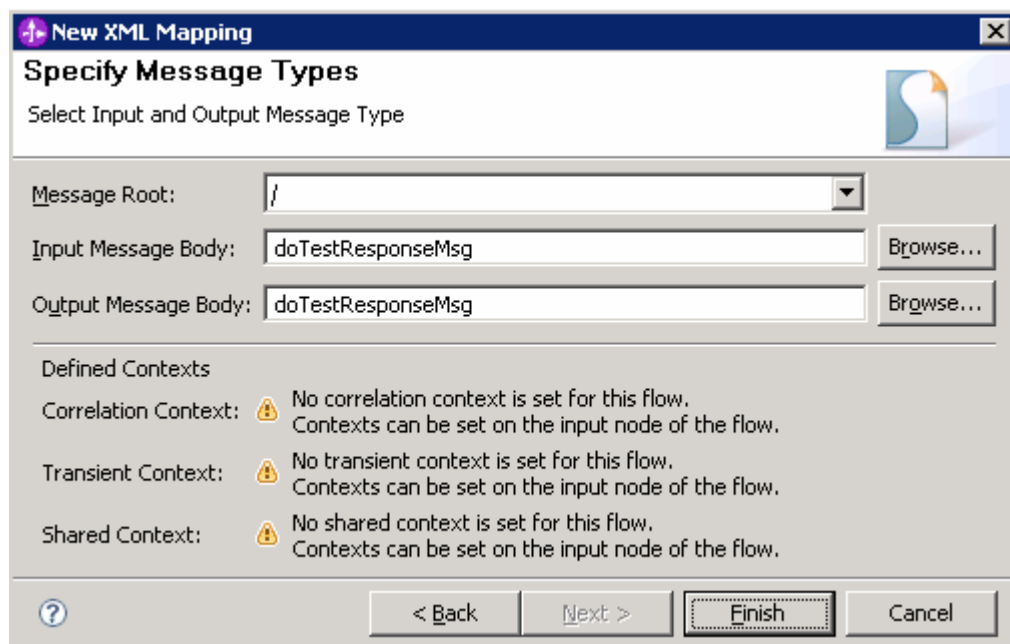
1) Enter these parameters in the **'XML Mapping'** panel

- Folder : **xslt** (default)
- Name : **XSLTransformation1\_res\_1** (default)

2) Click **Next**

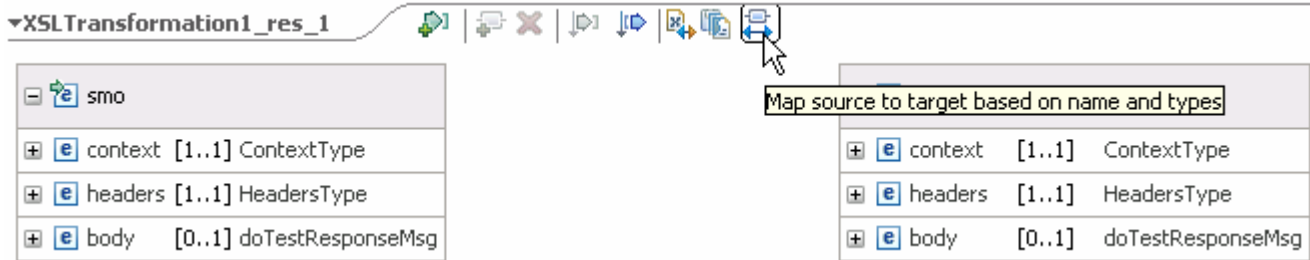
3) In the **'Specify Message Types'** panel, enter these parameters:


- Select **/** (root) for the Message Root field
- Since you wired the callout response node to XSLTransformation1, the input message body is already assigned as **doTestResponseMsg**
- Since you wired the XSLTransformation1 to input response node, the input message body is already assigned as **doTestResponseMsg**

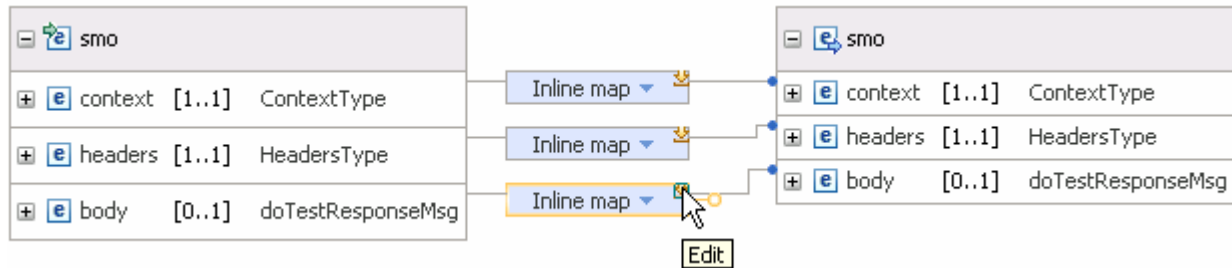


4) Click **Finish**

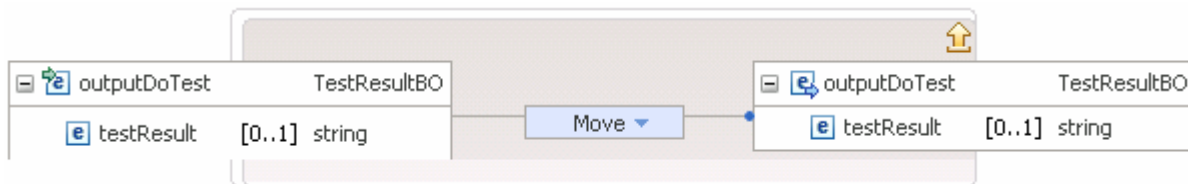
\_\_ c. The XSL Transformation mapping editor opens



\_\_\_ d. Click the toolbar icon (  ) **'Map source to target based on name and types'** which will map the context, headers and body using inline maps as shown below:



\_\_\_ e. Navigate down into the body/doTestResponse/outputDoTest inline map and review the **'testResult'** move transform



\_\_\_ f. Save all work (**File → Save All** or **Ctrl + Shift + S**)

\_\_\_ g. Close XSL transformation Editor

\_\_\_ 10. **Close** the Mediation Flow Editor

## Part 4: Test dynamic end points

In this section of the lab, a JSP client is used test various service endpoints with the Dynamic End Point property enabled and disabled for the Callout

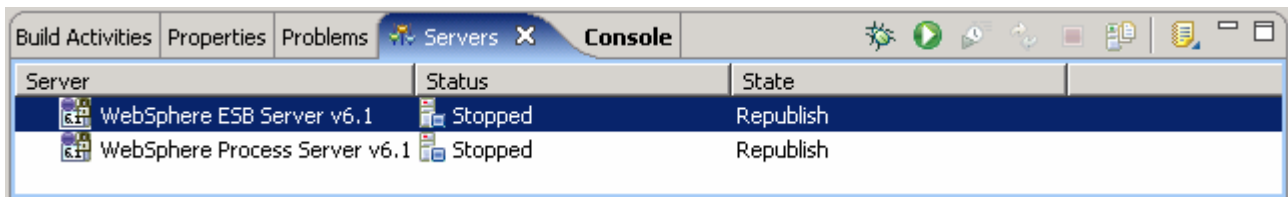
- \_\_\_ 1. Start **WebSphere Enterprise Service Bus Server** and **add modules** to server

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

If using a local testing environment:

- \_\_\_ a. Open **Servers View**

- \_\_\_ b. Select the WebSphere ESB Server V6.1 and right-click to select “(  ).



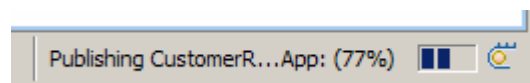
- \_\_\_ c. This takes some time. Wait for the server to start

- \_\_\_ 3. Add **projects** to WebSphere Enterprise Service Bus Server (once the ESB Server is started, not before).

- \_\_\_ a. In Servers view, right-click on '**WebSphere ESB Server v6.1**' and select '**Add and Remove Projects...**'

- \_\_\_ b. Click **Add All>>** button to move all projects to server and click **Finish** button

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



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



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

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

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

Ex: <ESB\_PROFILE\_HOME>\config\cells\esbCell\nodes\esbNode.

\_\_ f. Select the radio button next to each binding type in turn, and press the **Submit** button

<input type="radio"/> SCA Export, WebServices Binding - Soap/HTTP	http://localhost:9080/DynamicEndpointsServiceWeb/sca/WShttpExport
<input type="radio"/> SCA Export, WebServices Binding - Soap/JMS	jms:/queue? destination=jms/WSjmsComponentExport&connectionFactory=jms/WSjmsCompor
<input type="radio"/> SCA Export, SCA Binding	sca://DynamicEndpointsService/SCAExport
<input type="radio"/> SCA Import, JMS Binding	DynamicEndpointsMediation/JmsImport
<input checked="" type="radio"/> Enter an endpoint address:	<input type="text"/>

Submit

**Note:** The SOAP/HTTP endpoint address is the **default\_http** port of the WebSphere Enterprise Service Bus profile. If other than default 9080, provide the correct HTTP port and use the text box and the radio button provided to request the URL. For example:  
http://localhost:9081/DynamicEndpointsServiceWeb/sca/WShttpExport

Enter an endpoint address:

Submit

\_\_ g. Verify that the endpoint requested matches the endpoint invoked

1) Invoking **“SCA Export, WebServices Binding – SOAP/HTTP”**

```
Requested URL --> http://localhost:9081/DynamicEndpointsServiceWeb/sca/WShttpExport
Export/Component invoked --> WShttpExport/WShttpComponent
```

2) Invoking **“SCA Export, WebServices Binding – SOAP/JMS”**

```
Requested URL --> jms:/queue?
destination=jms/WSjmsComponentExport&connectionFactory=jms/WSjmsComponentExportQCF&
Export/Component invoked --> WSjmsExport/WSjmsComponent
```

### 3) Invoking “SCA Export, SCA Binding”

Requested URL --> sca://DynamicEndpointsService/SCAExport

Export/Component invoked --> SCAExport/SCAComponent

### 4) Invoking “SCA Import, JMS Binding”

Requested URL --> DynamicEndpointsMediation/JmsImport

Export/Component invoked --> JmsExport/JmsComponent

\_\_\_ h. Close the browser

\_\_\_ i. Remove the projects from the ESB server. To remove the projects, right-click over the ESB server in the Server view and select “**Add and Remove projects...**” from the context menu. From the Add and Remove dialog click over the **Remove All** and then click **Finish**

\_\_\_ 4. Return to the Mediation Flow Editor and update the **Callout** property to disable the dynamic endpoints property.

\_\_\_ a. In the Business Integration view, expand **DynamicEndpointsMediation -> Mediation Logic -> Flows** and double click on **Mediation1** to open Mediation Flow Editor.

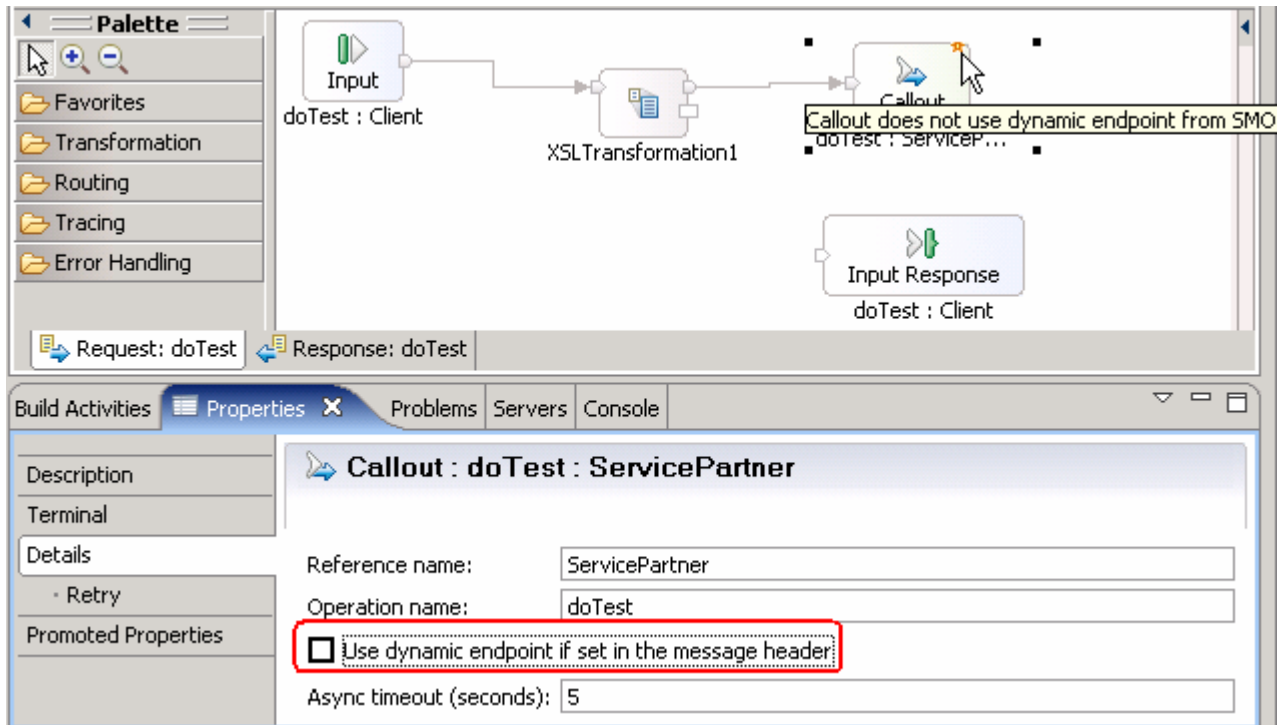
\_\_\_ b. In the Mediation Flow Editor, click on the Mediation Flow line which will open the middle window

\_\_\_ c. In the middle window, select the Request **Callout** node and select the **Details** tab under its properties view to modify the callout property

The screenshot displays the Mediation Flow Editor interface. On the left is a Palette with categories like Favorites, Transformation, Routing, Tracing, and Error Handling. The main workspace shows a flow diagram with nodes: Input (doTest: Client), XSLTransformation1, Callout (doTest: ServiceP...), and Input Response (doTest: Client). Below the workspace is a Properties view for the selected 'Callout : doTest : ServicePartner' node, showing fields for Reference name (ServicePartner), Operation name (doTest), a checked checkbox for 'Use dynamic endpoint if set in the message header', and Async timeout (seconds) set to 5.



\_\_ d. Unselect the check box next to “Use dynamic endpoint if set in the message header”



\_\_ e. Save all work by choosing **File > Save All** or **Ctrl + Shift + S**

\_\_ f. Add all the projects again to run on the ESB server

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



\_\_ h. Enter [http:// <HOSTNAME>:<PORT>/DynamicEndpointsClientWeb/index.jsp](http://<HOSTNAME>:<PORT>/DynamicEndpointsClientWeb/index.jsp) . Where hostname is the name of the system where the WebSphere Enterprise Service Bus server is located. Port is the **default\_http** port of the WebSphere Enterprise Service Bus profile.

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

\_\_ i. Select the radio button next to each binding type in turn, and press the **Submit** button

\_\_ j. Verify that the requested endpoint invoked is **DefaultComponentExport** for each request

1) Invoking “**SCA Export, WebServices Binding - Soap/HTTP**”

```
Requested URL --> http://localhost:9081/DynamicEndpointsServiceWeb/sca/WShttpExport
Export/Component invoked --> DefaultComponentExport/DefaultComponent
```

2) Invoking “**SCA Export, WebServices Binding – Soap/JMS**”

```
Requested URL --> jms:/queue?  
destination=jms/WSjmsComponentExport&connectionFactory=jms/WSjmsComponentExportQCF&  
Export/Component invoked --> DefaultComponentExport/DefaultComponent
```

3) Invoking “**SCA Export, SCA Binding**”

```
Requested URL --> sca://DynamicEndpointsService/SCAExport  
Export/Component invoked --> DefaultComponentExport/DefaultComponent
```

4) Invoking “**SCA Import, JMS Binding**”

```
Requested URL --> DynamicEndpointsMediation/JmsImport  
Export/Component invoked --> DefaultComponentExport/DefaultComponent
```

\_\_\_ k. Close browser.

\_\_\_ l. Remove the projects from the ESB Server

\_\_\_ 5. The exercise is complete

---

## Part 5: 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:
    - DynamicEndpointsClient
    - DynamicEndpointsLibrary
    - DynamicEndpointsMediation
    - DynamicEndpointsService
    - Websphere\_default\_messaging\_provider
  - \_\_\_ d. Save in **C:/LabFiles61/WESB/DynamicEndpoints/**
  - \_\_\_ e. Name the project interchange **WESB\_DynamicEndpoints\_Solution\_PI.zip**
  - \_\_\_ f. Click **Finish** to save the file
- \_\_\_ 2. Remove all the projects and **clean** up the ESB Server if not already done.
  - \_\_\_ a. Right-click on WebSphere ESB Server V6.1 (once started) and select **Add and Remove projects...** from the context menu
  - \_\_\_ 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 were provided with an understanding of how to create a mediation module and mediation flow in WebSphere Integration Developer V6.1. A XSLT primitive was added to the Request and Response flow of the Mediation and the Service Message Object Header was mapped from source to target using the XSL editor. You also tested the application by deploying it to the integrated WebSphere Enterprise Service Bus test server with Callout property enabled and then disabled

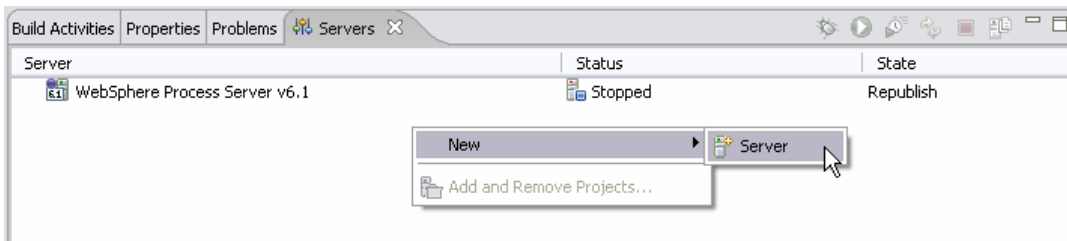
## Solution instructions

- \_\_\_ 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/DynamicEndpoints/WESB\_DynamicEndpoints\_Solution\_Pi.zip**
  - \_\_\_ c. Select All Projects and click the **Finish** button. Ignore any warnings reflected in the Problems view
- \_\_\_ 2. Start with **Part 4: Test Dynamic End Points**

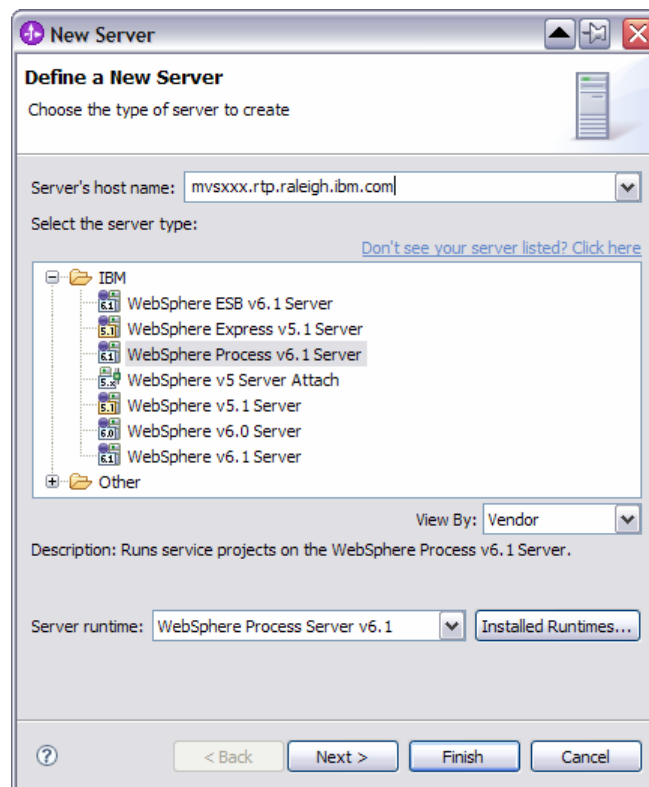
## Task: Adding remote server to WebSphere Integration Developer test environment

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

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

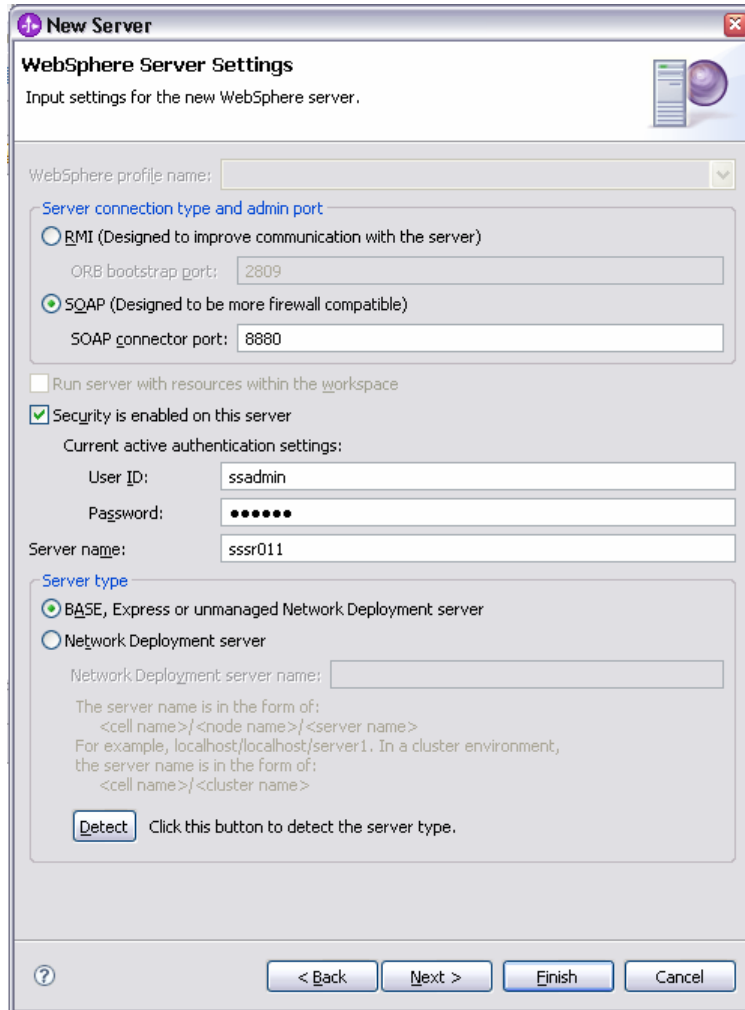


- \_\_\_ c. In the New Server dialog, specify the remote server's host name, **<HOSTNAME>**.
- \_\_\_ d. Ensure that the appropriate server type, **'WebSphere Process v6.1 Server'** or **'WebSphere ESB v6.1 Server'**, is highlighted in the server type list

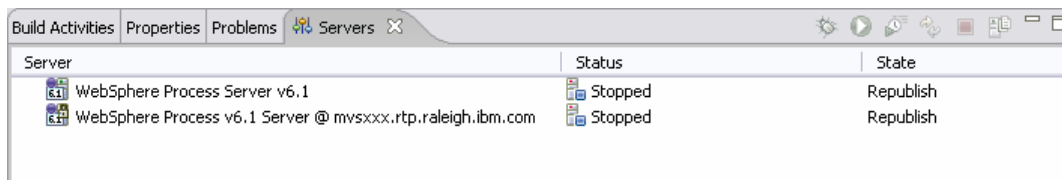


- \_\_\_ e. Click **Next**.

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



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



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

**'telnet <HOSTNAME> <TELNET\_PORT>'**

User name: **<USERID>**

Password: **<PASSWORD>**

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

**cd <WAS\_HOME>/profiles/<PROFILE\_NAME>/bin**

\_\_ c. Run the command file to start the server: **./startServer.sh <SERVER\_NAME>**

\_\_ d. Wait for status message indicating server has started:

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

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