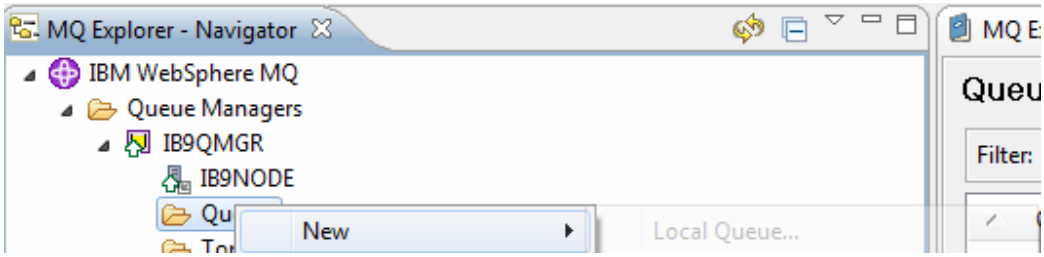
Lab 3 Content Based Routing and the Debugger

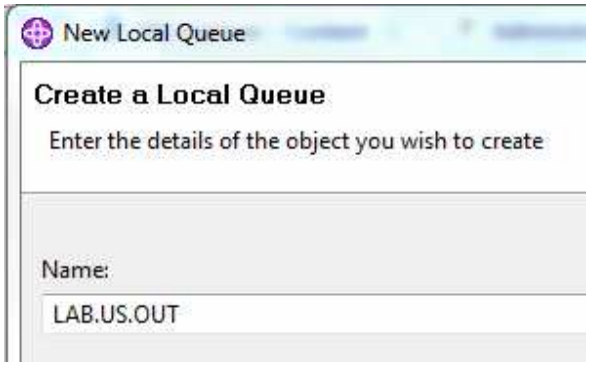
3.1 Overview

In this lab, you will perform simple routing. Input messages will be sent to one of three destinations depending on the country code. Addresses in the United States will be sent to a US shipping queue, while addresses in Canada will be sent to a Canadian shipping queue, and those that are not in the United States or Canada will be sent to a third queue.

3.2 Create the Additional MQ Queues in the WebSphere MQ Explorer

Right click on Queue, select New and Local Queue





Repeat for **LAB.CANADA.OUT**

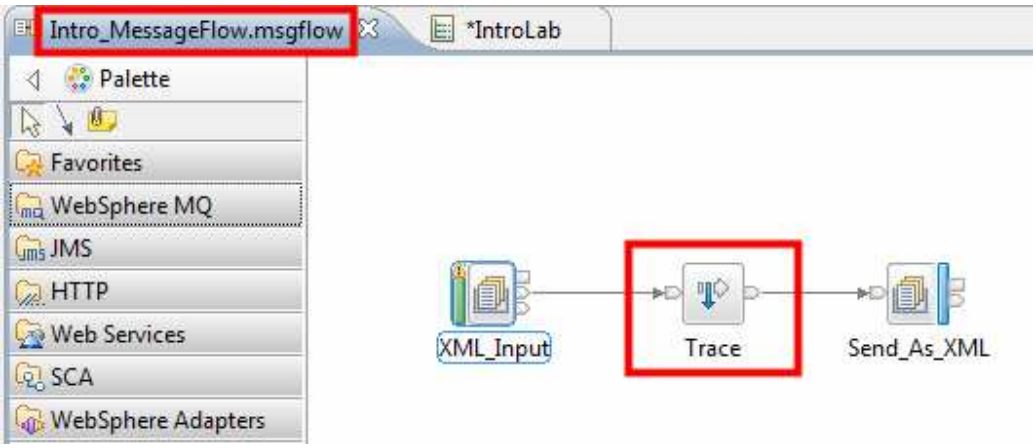
3.3 Add Routing Logic

__1. Return to the Integration Bus Toolkit.

__2. Select the **IntroMessageFlow** message flow in the **message flow editor**.

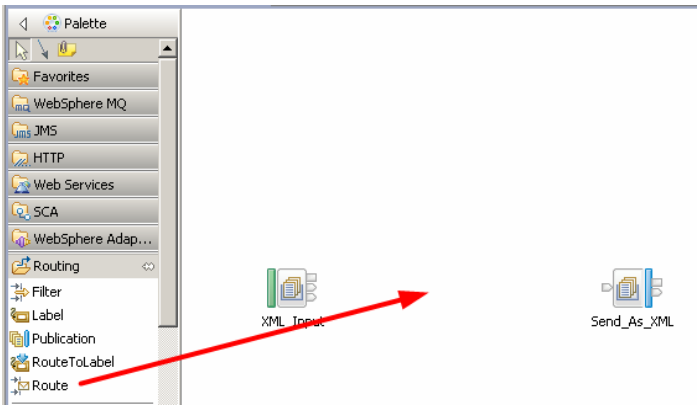
__3. Select the **Trace** node.

__4. Press the **Delete** key.



__5. Expand the **Routing** drawer in the **Palette**.

__6. Select a **Route** node and place it between the **XML Input** and **Send_As_XML** nodes.



__7. Change the name of the new routing node to **CheckCountry**.

__8. Press the **Enter** key to complete the rename operation.

__9. Wire the **Out** terminal of the **XML_Input** node to the **In** terminal of the **CheckCountry** node.

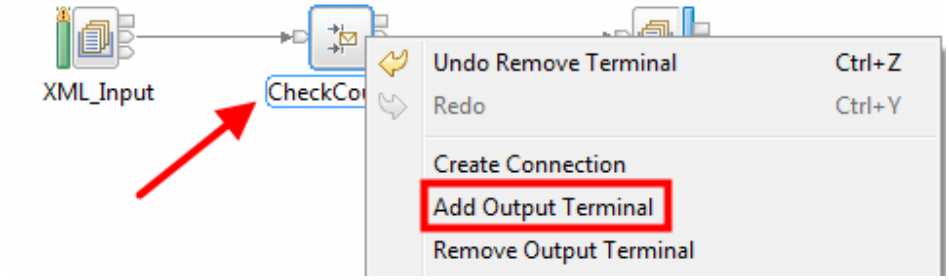
__10. Wire the **Default** terminal of the **CheckCounty** node to the **Send_As_XML** node.

Terminals will now be added to the **CheckCountry** routing node for **US** and **Canadian** addresses.

__11. Select the **CheckCountry** route node.

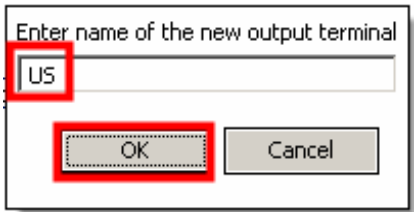
__12. Press the right mouse button.

__13. Select **Add Output Terminal** from the menu.



__14. Enter **US** as the name of the new output terminal.

__15. Press the **OK** button to continue.

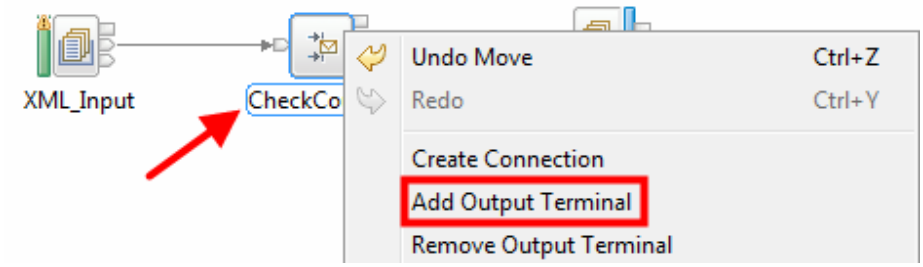


The steps will now be repeated to add a terminal called **Canada**.

__16. Select the **CheckCountry** node.

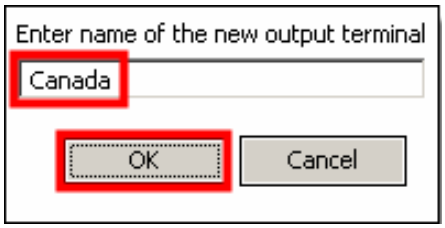
__17. Press the right mouse button.

__18. Select **Add Output Terminal** from the menu.



__19. Enter **Canada** as the name of the new output terminal.

__20. Press the **OK** button to continue.



Two additional output destinations will now be added. One will be for addresses within the United States and a second one will be for addresses within Canada. All other addresses will be sent to the existing output queue.

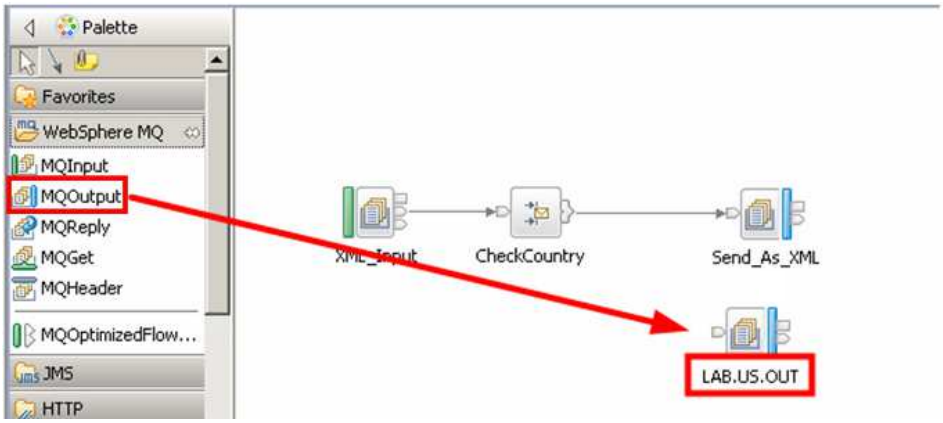
__21. On the **Palette** expand the **WebSphere MQ** drawer.

__22. Select an **MQOutput** node.

__23. Place the new node below the **Send_As_XML** node.

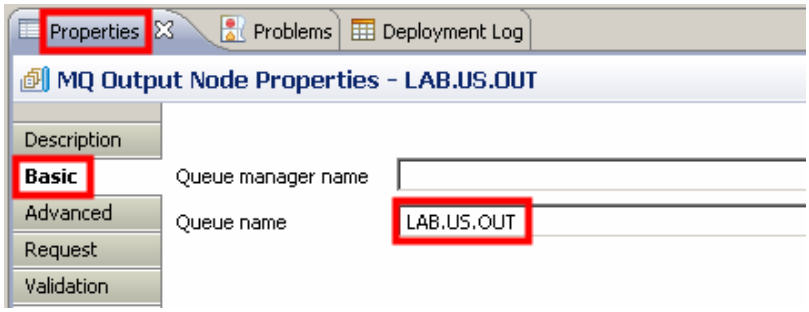
__24. Change the name of the node to **LAB.US.OUT**

__25. Press the **Enter** key to complete the rename operation.



__26. In the **Properties** pane select the **Basic** tab.

__27. Enter **LAB.US.OUT** as the **Queue name** parameter.



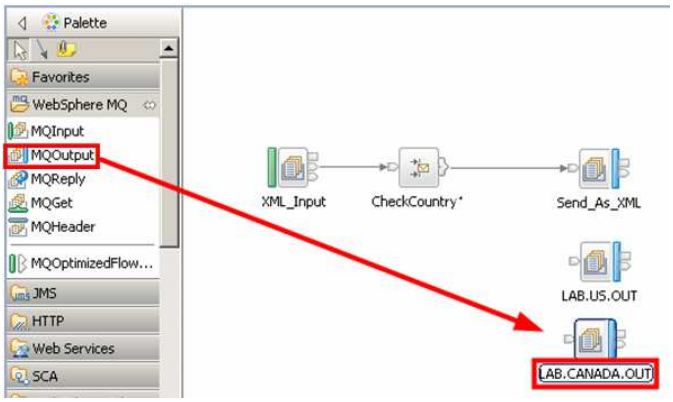
__28. Select the **MQOutput** node in the **WebSphere MQ** drawer again.

__29. Move the mouse pointer to a point below the **LAB.US.OUT** node.

__30. Press the left mouse button to place the new node below the **LAB.US.OUT** node.

__31. Change the name to **LAB.CANADA.OUT**.

__32. Press the **Enter** key to complete the rename operation.

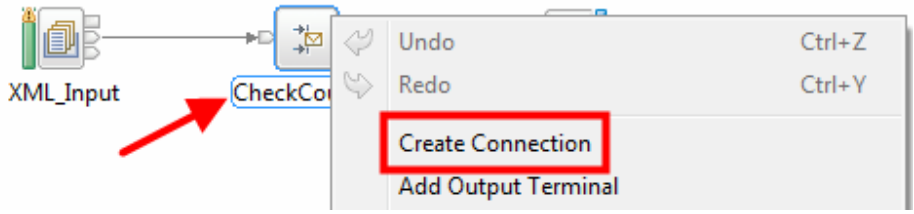


__33. In the **Properties** pane select the **Basic** tab.

__34. Enter **LAB.CANADA.OUT** as the **Queue name** parameter.

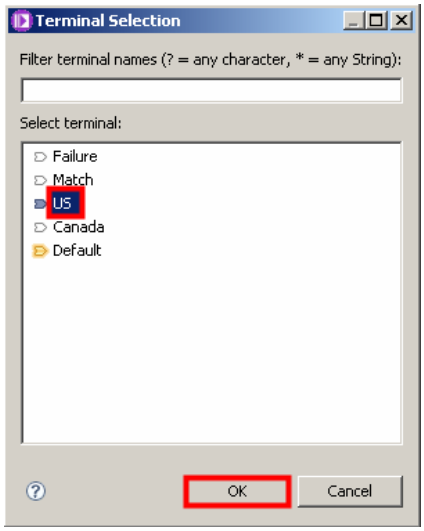
__35. Right click the **CheckCountry** route node.

__36. Select **Create Connection** from the menu.



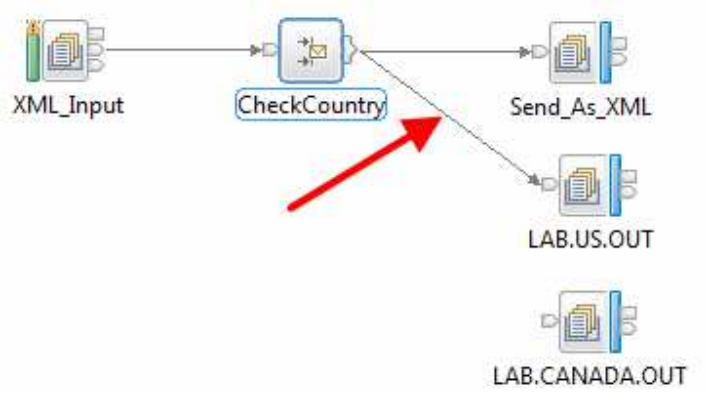
__37. In the Terminal Selection popup, select **US**.

__38. Click **OK**.



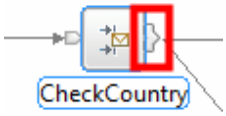
__39. Drag the wire that appears to the **LAB.US.OUT** node.

__40. Press the left mouse button to complete the connection.



__41. Another way to make a connection is just to click on the terminal itself. Try this by selecting the output terminal of the **CheckCountry** node.

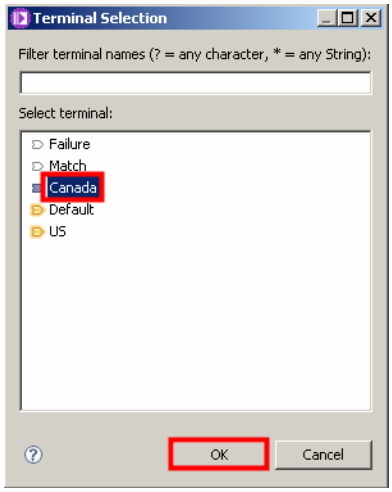
__42. Press the left mouse button.



Since there are too many terminals to individually depict graphically, a dialog box appears allowing you to select the proper terminal.

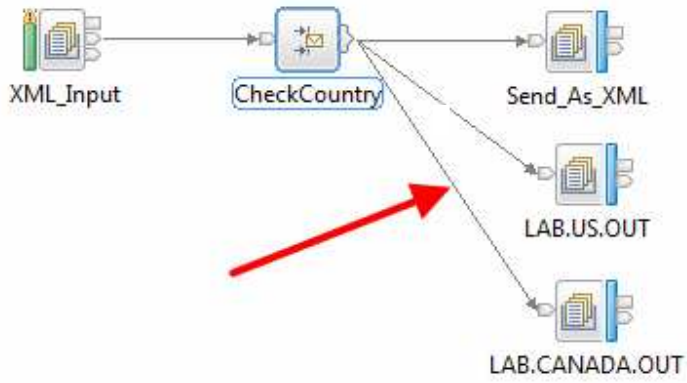
__43. Select the **Canada** terminal.

__44. Press the **OK** button.



__45. Drag the wire that appears to the **LAB.CANADA.OUT** node.

__46. Press the left mouse button to complete the connection.

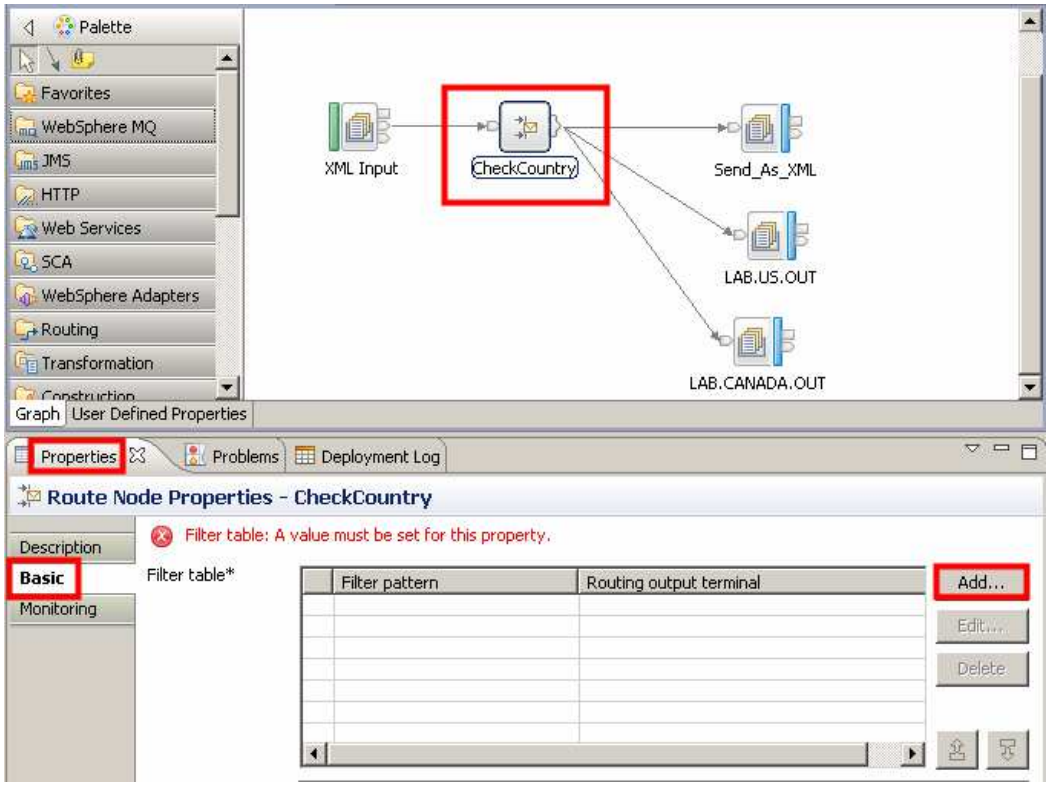


The criteria to be used by the **CheckCountry** routing node must now be specified.

__47. Select the **CheckCountry** node.

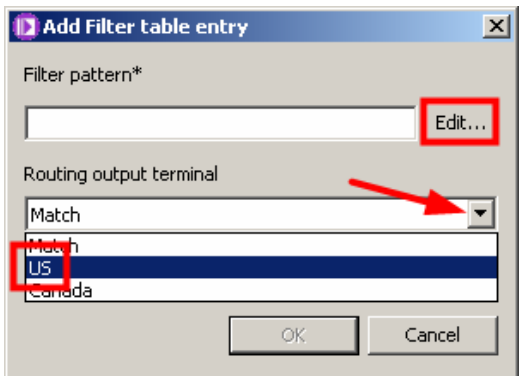
__48. In the **Properties** pane select the **Basic** tab.

__49. Press the **Add** button.



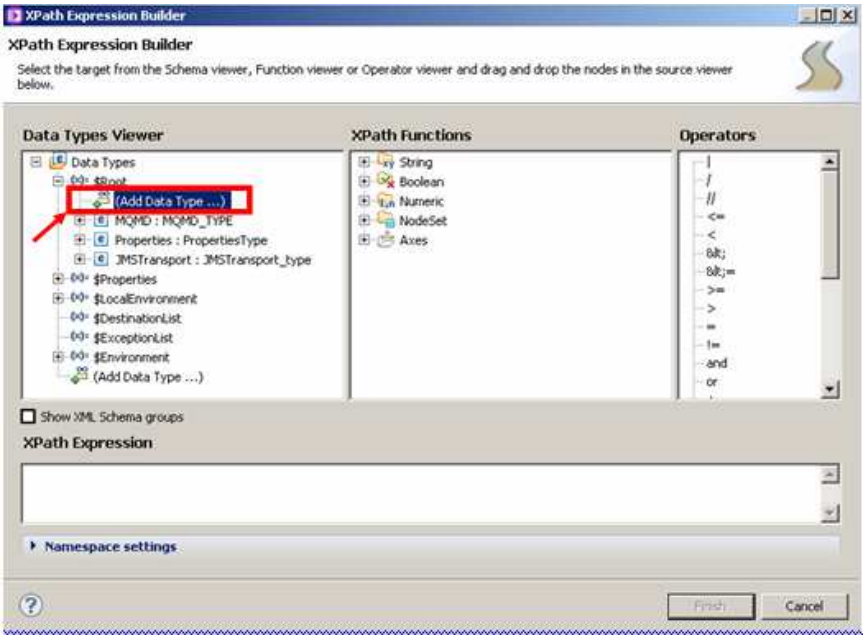
__50. Use the drop down menu to select the **US** terminal as the **Routing output terminal**.

__51. Press the **Edit…** button.



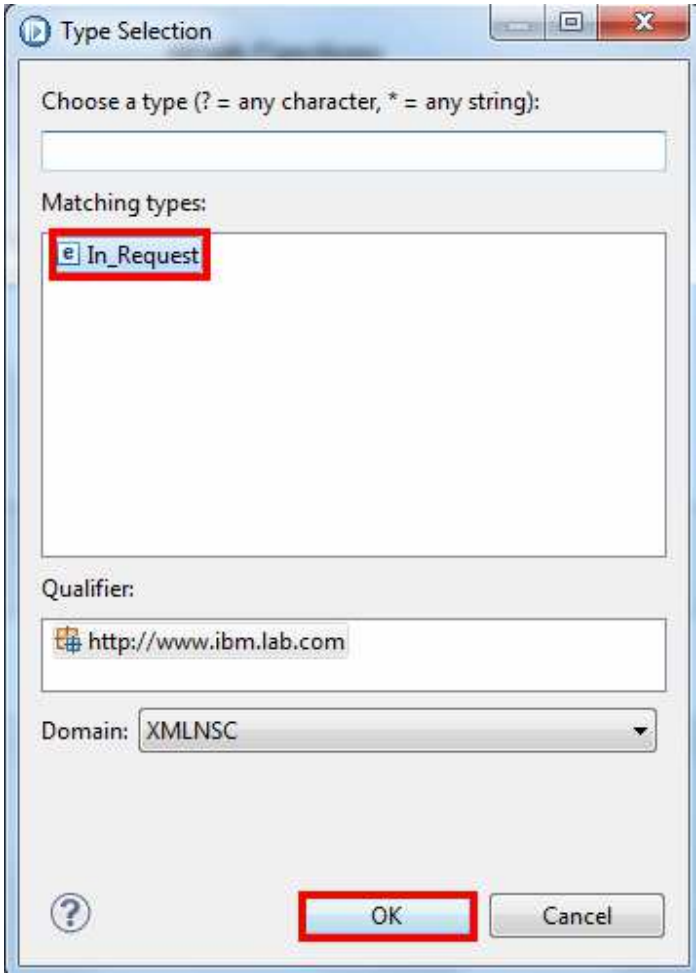
__52. Expand **$Root**.

__53. Select **(Add Data Type …)**.



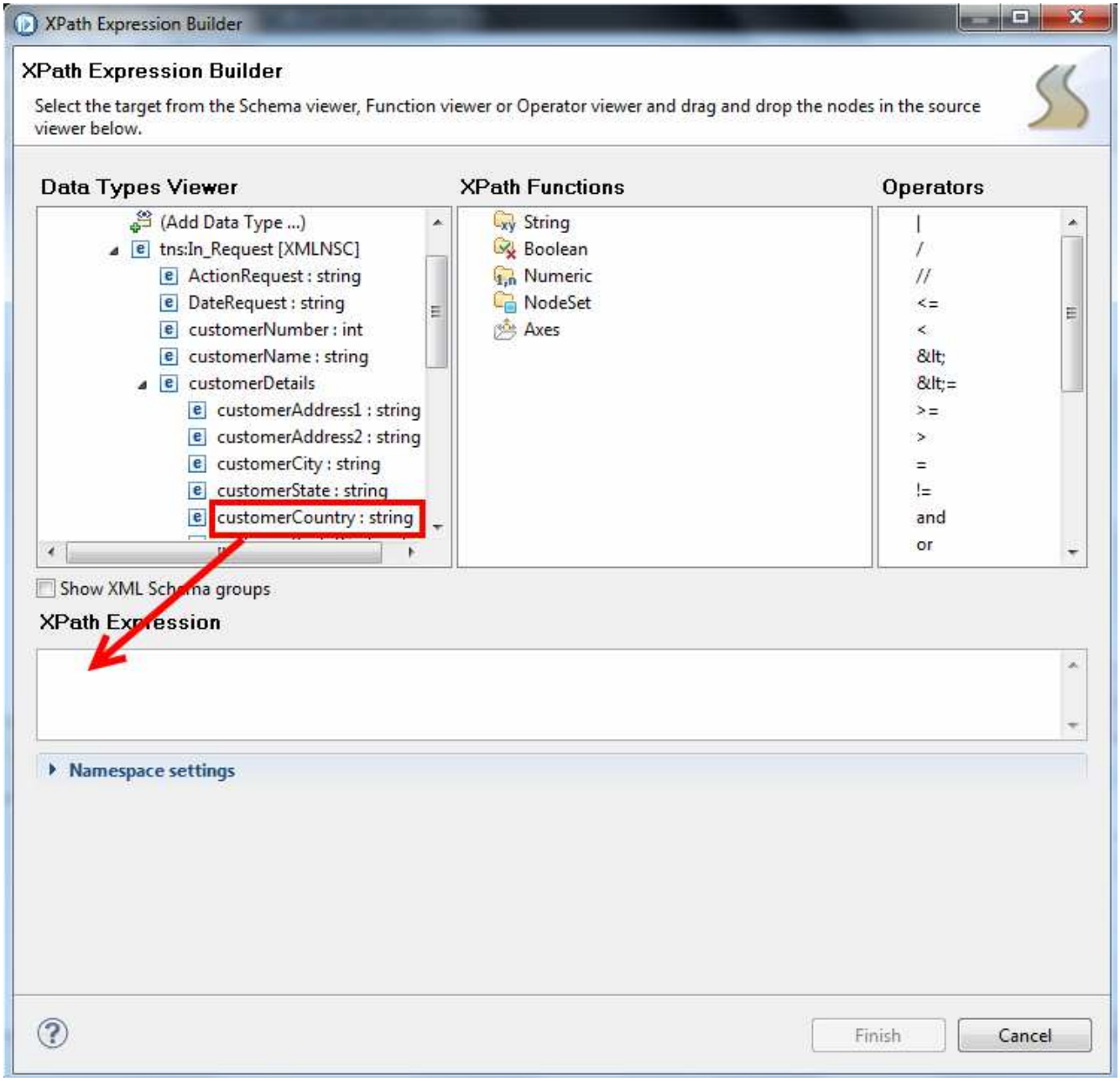
__54. Select **In_Request**.
Note: If you do not see In_Request in the list, then you must add a reference to the Library by right clicking the IntroLab application and selecting **Manage Library references**.

__55. Press the **OK** button to continue.



__56. Expand the **tns:In_Request→customerDetails** elements.

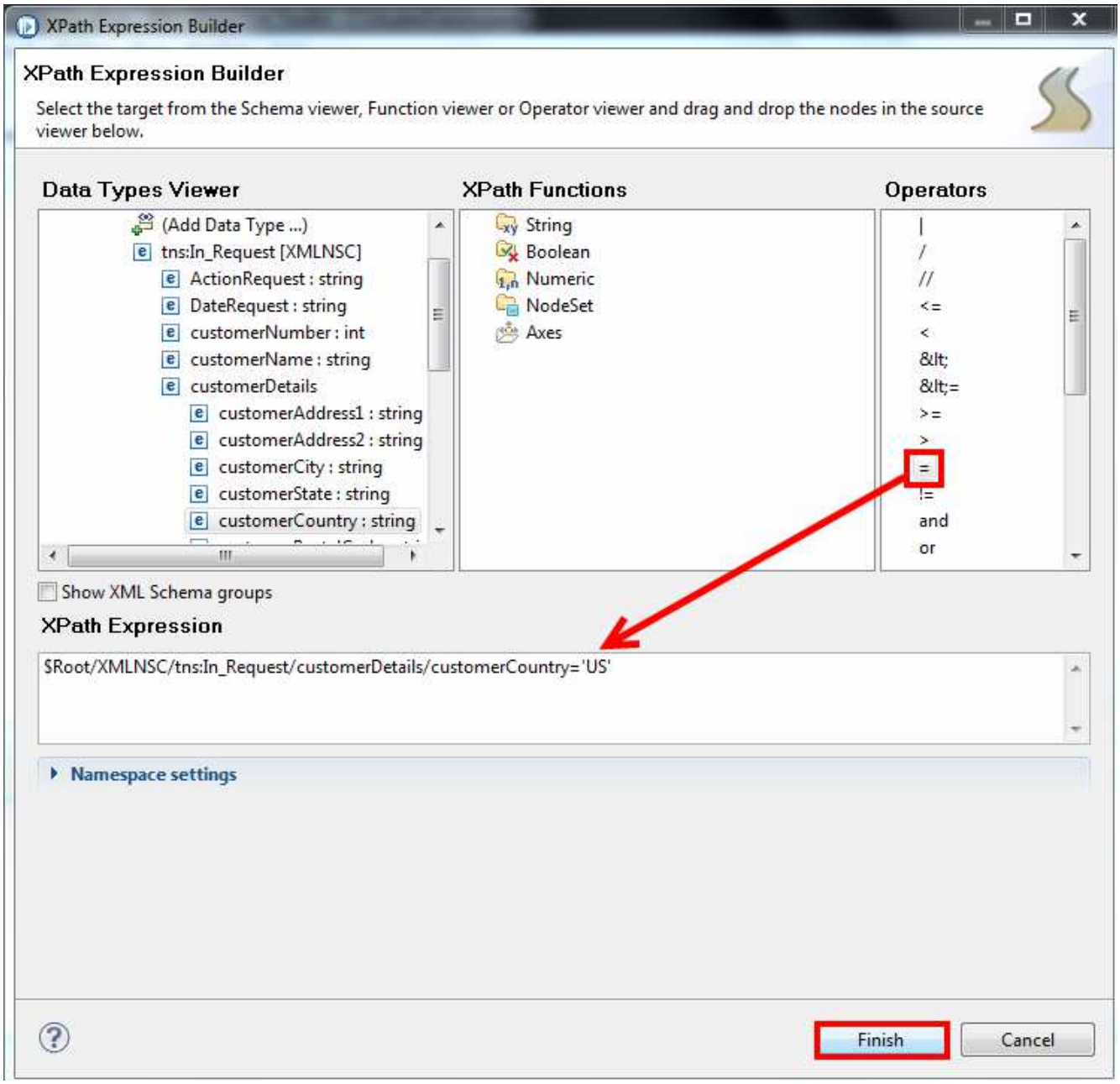
__57. Select the **customerCountry** element and **drag** it into the **XPath Expression** dialog box.



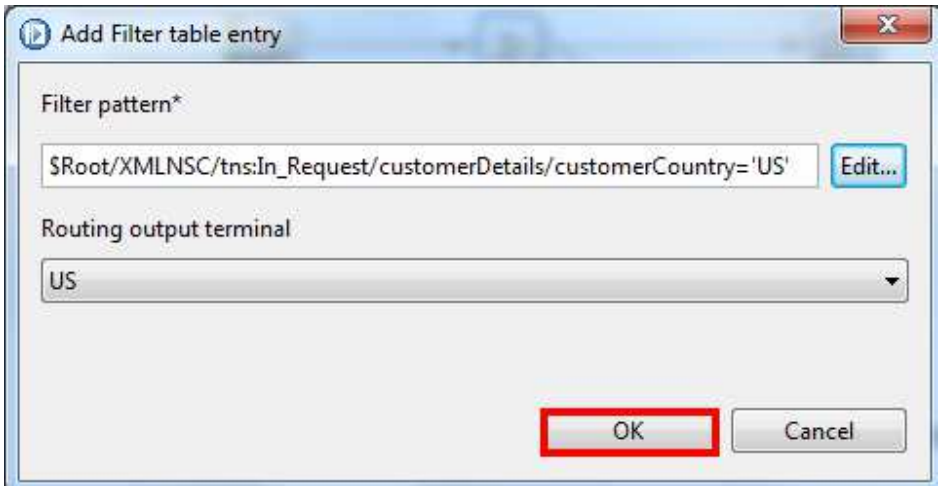
__58. Drag an equal sign from the **Operators** pane to the end of the expression.

__59. Append the letters **'US'** (including the single quotes) after the equal sign.

__60. Press the **Finish** button to complete the XPath expression.

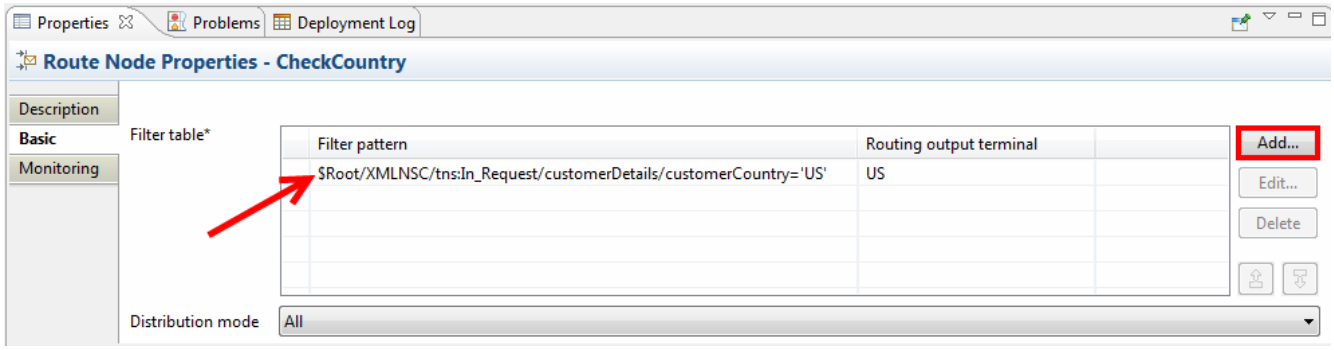


__61. Press the **OK** button to complete the **Filter table entry**.



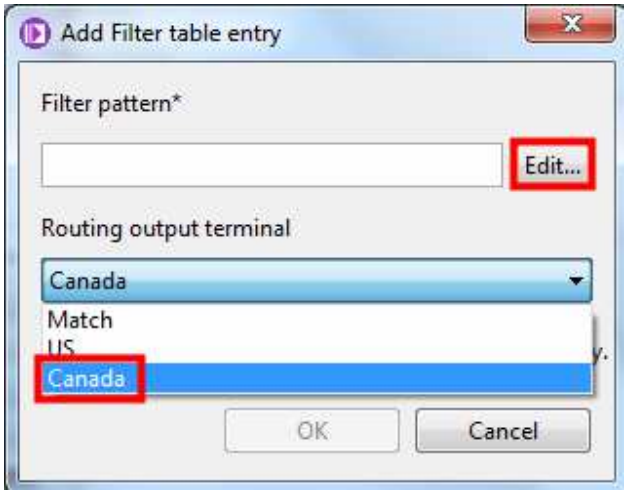
The filter pattern for the **US** terminal should now be visible. The process will now be repeated to create a Filter table entry for the **Canada** terminal.

__62. Press the **Add** button to enter a filter pattern for the **Canada** terminal.



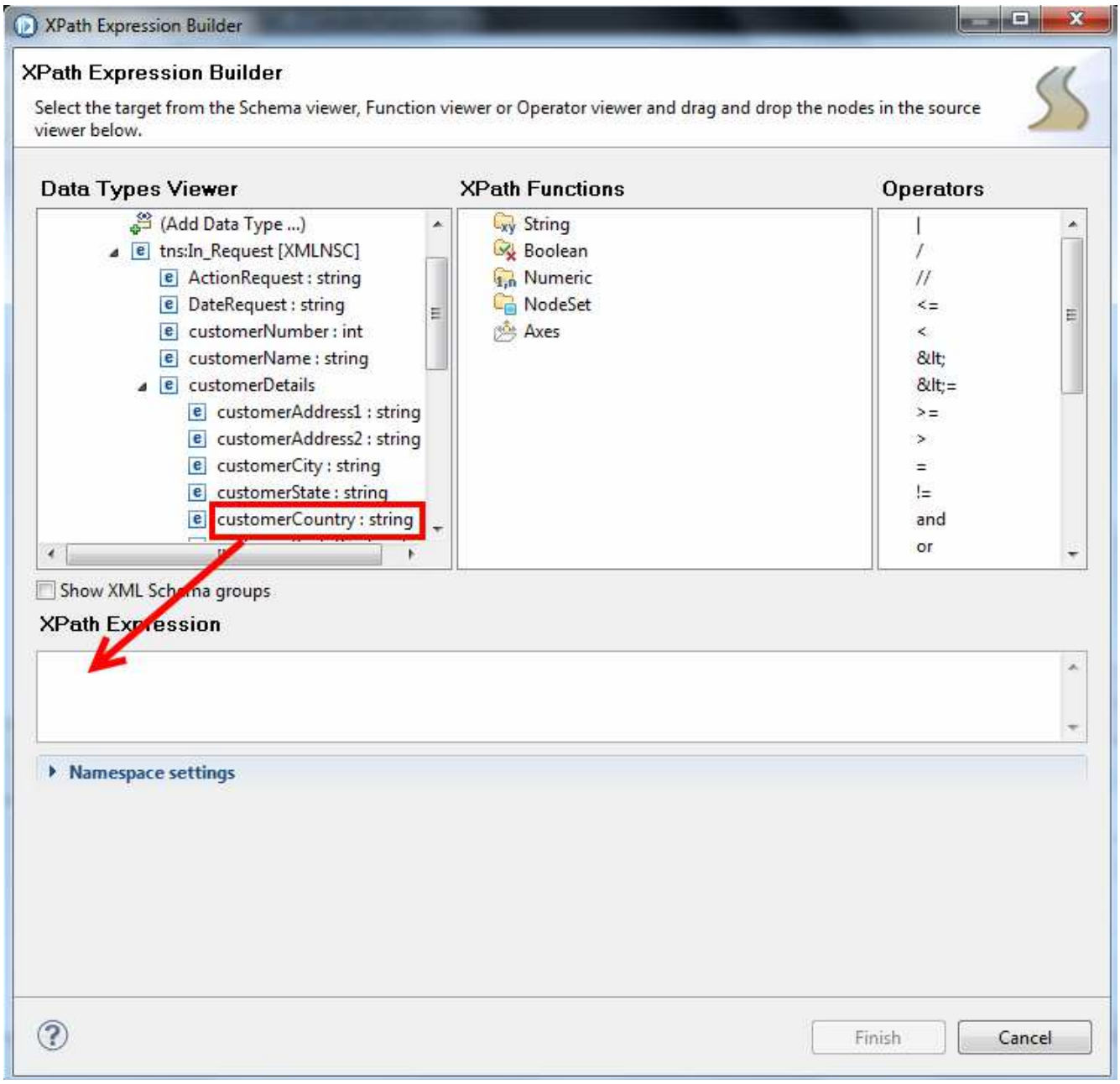
__63. Use the drop down menu to select the **Canada** terminal as the **Routing output terminal**.

__64. Press the **Edit…** button.



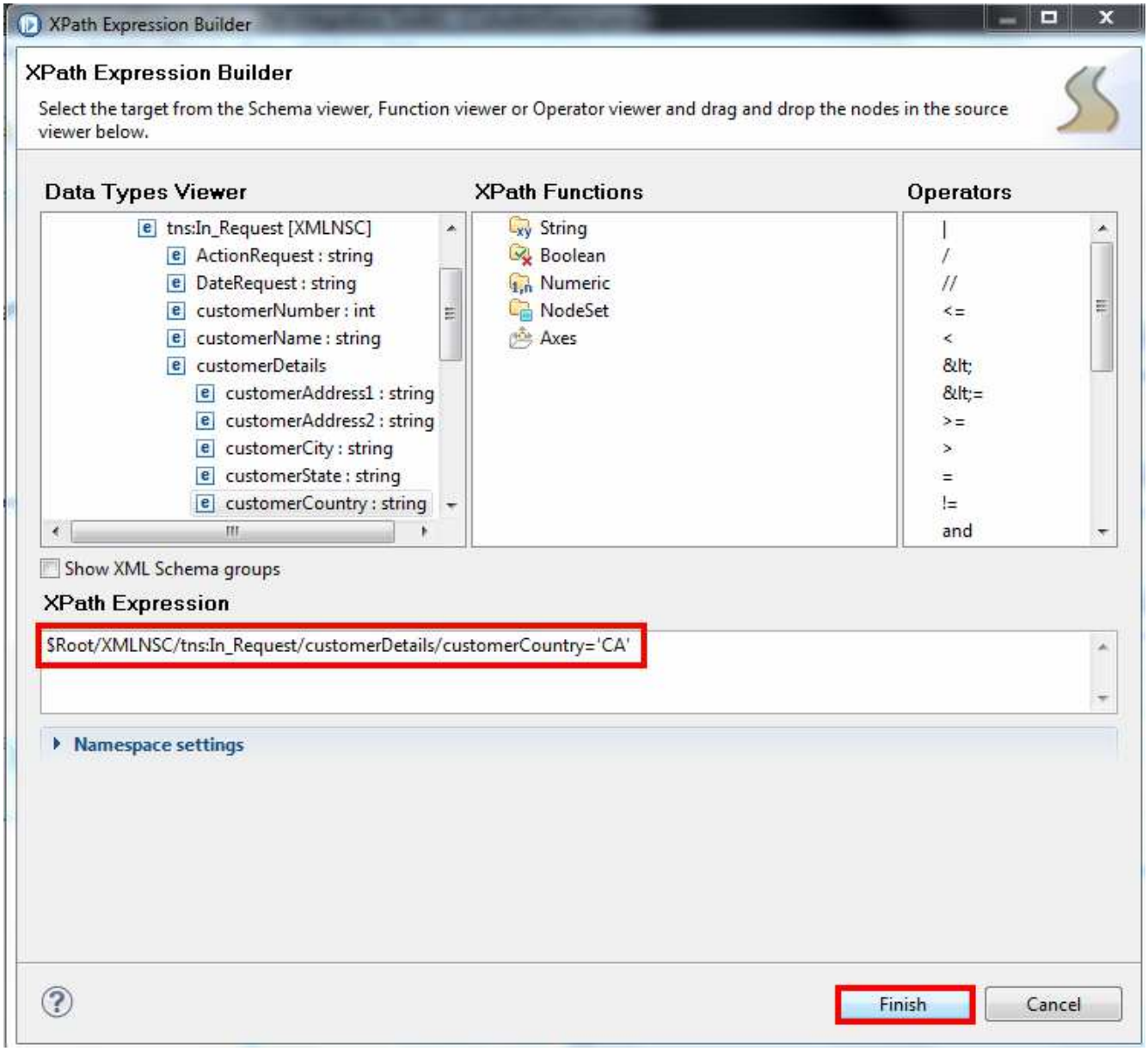
__65. Expand **Root→tns:In_Request→customerDetails**. Tip: If you do not see In_Request in the Viewer, re-add it as done in steps 53-56.

__66. Select the **customerCountry** field and drag it to the **XPath Expression** window.

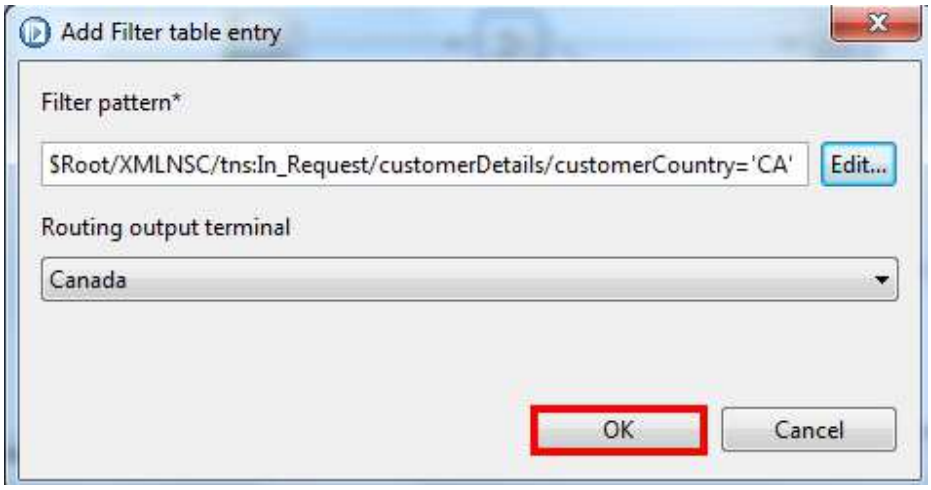


__67. Complete the XPath Expression by typing **=’CA’**.

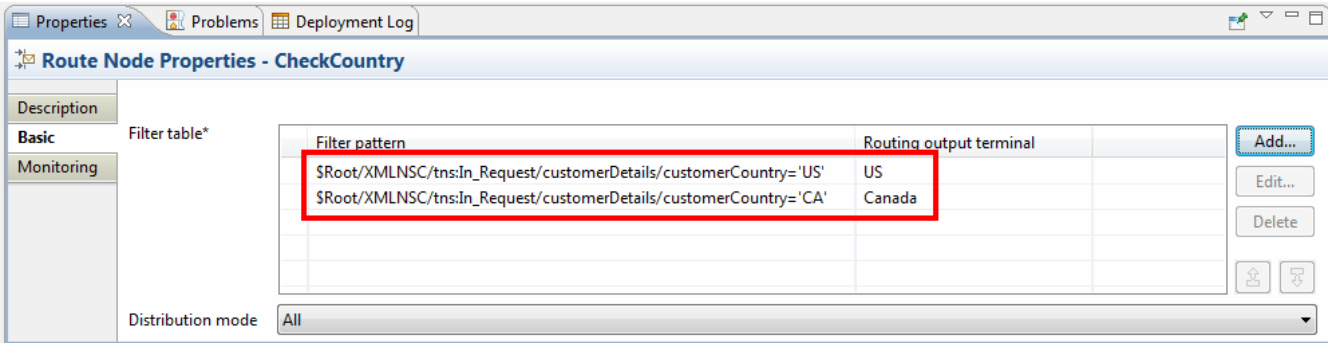
__68. Press the **Finish** button to complete the XPath expression.



__69. Press the **OK** button to complete the Filter table entry.



The filter pattern for the **Canada** terminal should now be visible.



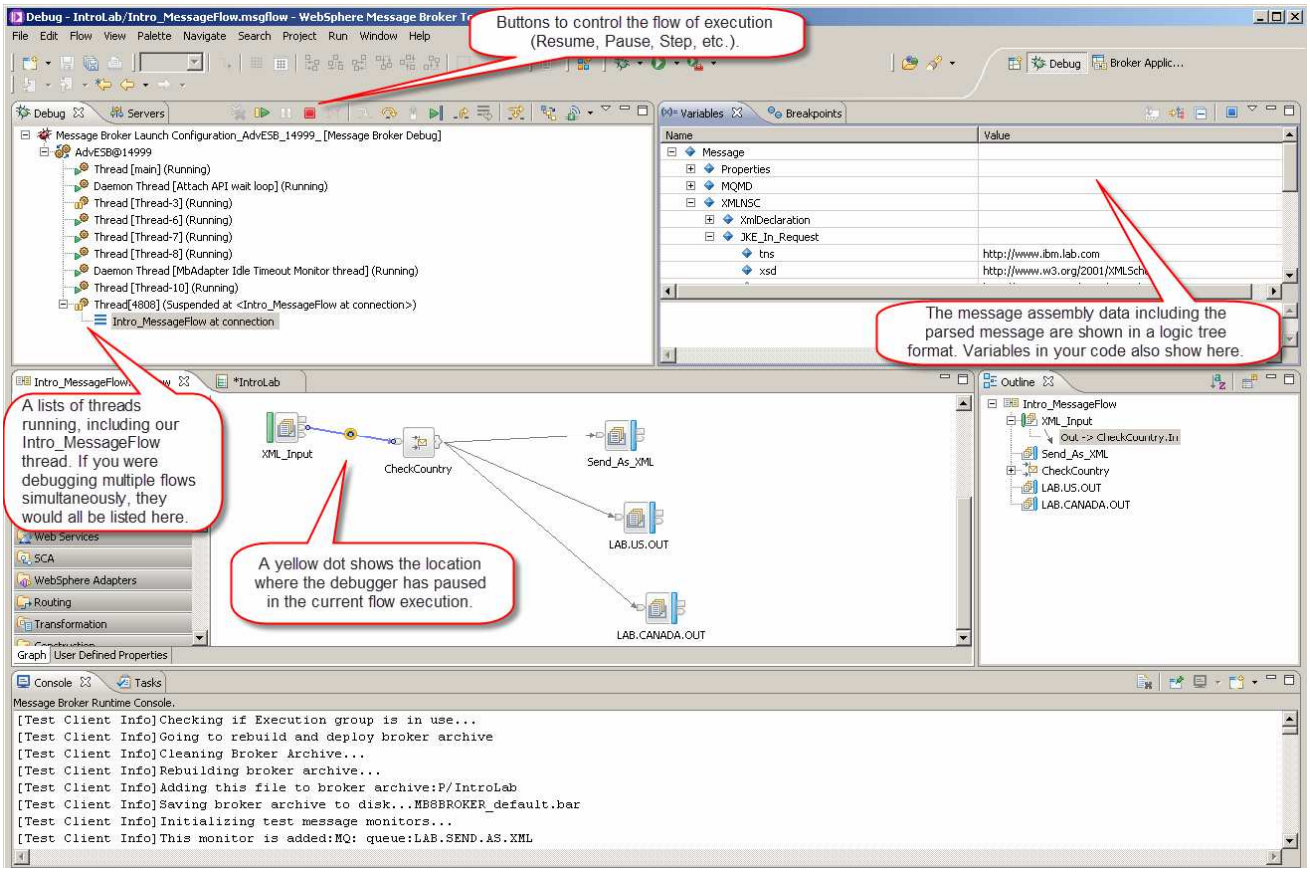
The updates to the message flow are now complete.

__70. 

Save the message flow.

3.4 Test with the Debugger

Next, we are going to test the application with the graphical debugger.



Key Idea: The Graphical Debugger

Use the flow debugger in the IBM® Integration Bus Toolkit to track messages through your message flows.

Use the Debug perspective in the IBM Integration Toolkit to use the flow debugger. The diagram above introduces the Debug perspective and the views it presents.

You can set breakpoints in a message flow, and then step through the flow. While you are stepping through, you can examine and change the message variables and the variables used by ESQL code and Java code. You can debug a wide variety of error conditions in flows, including the following:

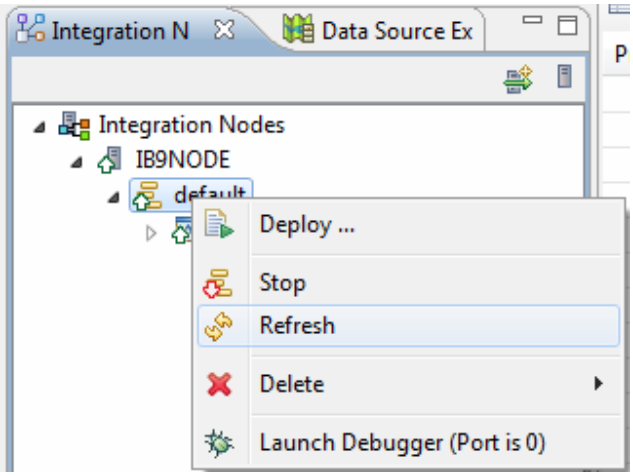
- Nodes that are wired incorrectly (such as outputs that are connected to the wrong inputs)
- Errors in transformation or logic within your code or maps
- Incorrect conditional branching in transition conditions
- Unintended infinite loops in flow

From a single IBM Integration Toolkit, you can attach the debugger to one or more integration servers, and debug multiple flows in different integration servers (and therefore multiple messages) at the same time. However, an integration server can be debugged by only one user at a time. Therefore, if you attach your debugger to an integration server, another user cannot attach a debugger to that same integration server until you have ended your debugging session.

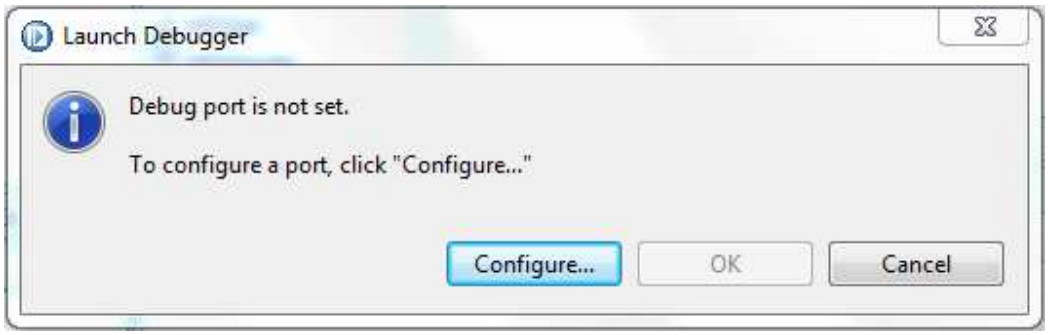
__1. In the **Integration Node** view in the bottom left, right click on the Integration Server called **default.**

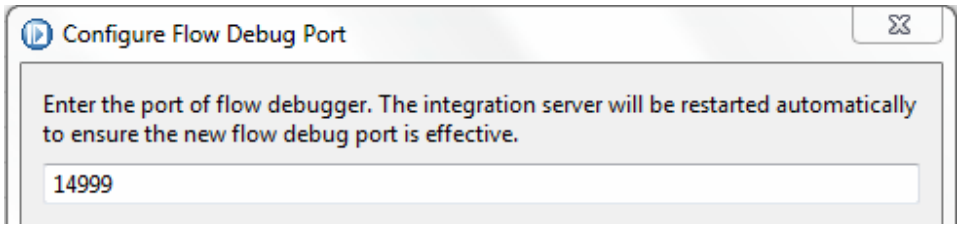
__2. Select **Launch Debugger** from the menu.

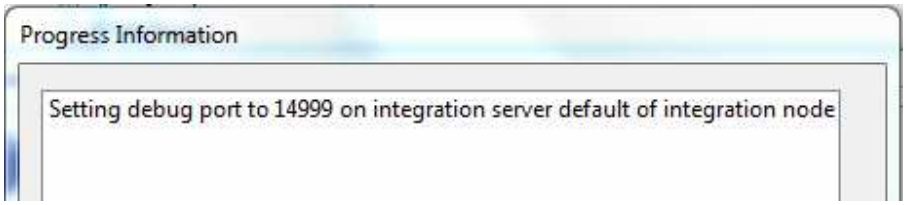
If this is the first time you have used the debugger you will need to configure its port number



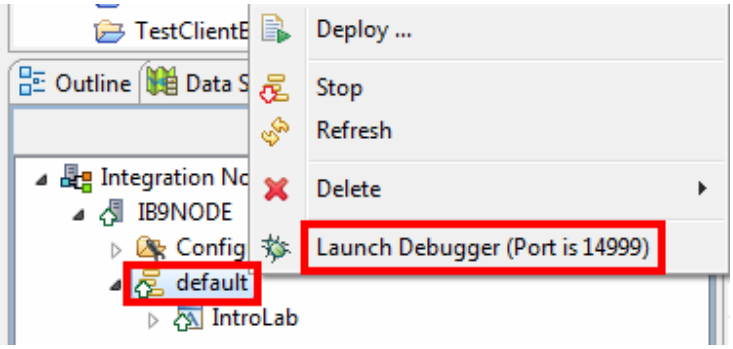
Select Launch Debugger (Port is 0), then hit configure



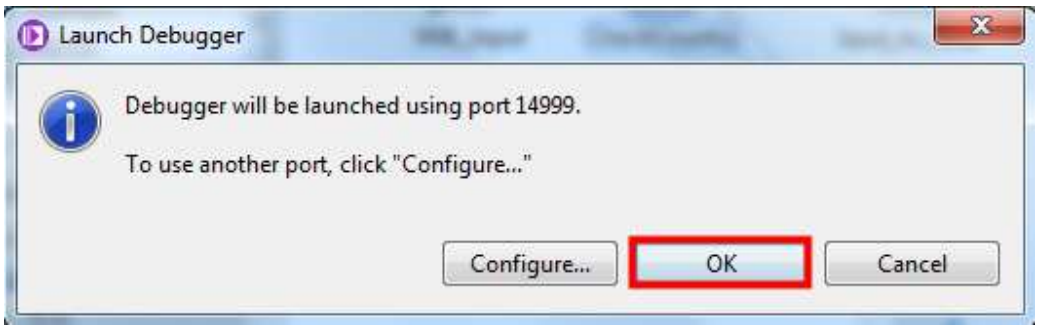




If the debugger was already configured it will launch directly

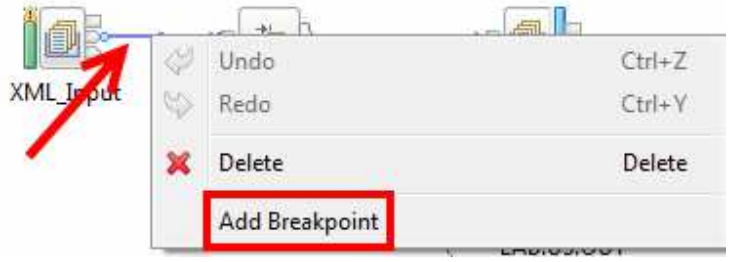


__3. In the subsequent Launch Debugger dialog, select **OK**.

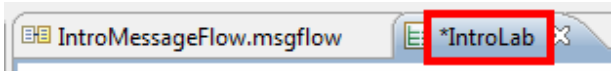


__4. In the **IntroMessageFlow**, right click the wire between the **XML_Input** node and the **Check Country** route node.

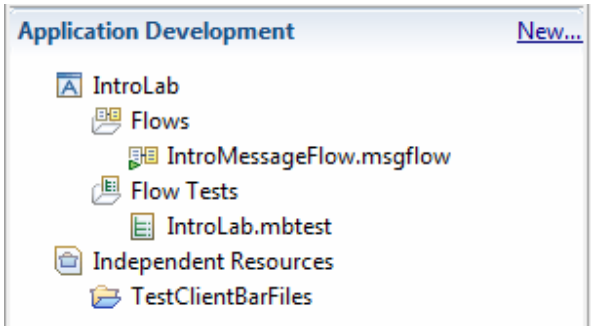
__5. Select **Add Breakpoint** from the menu.



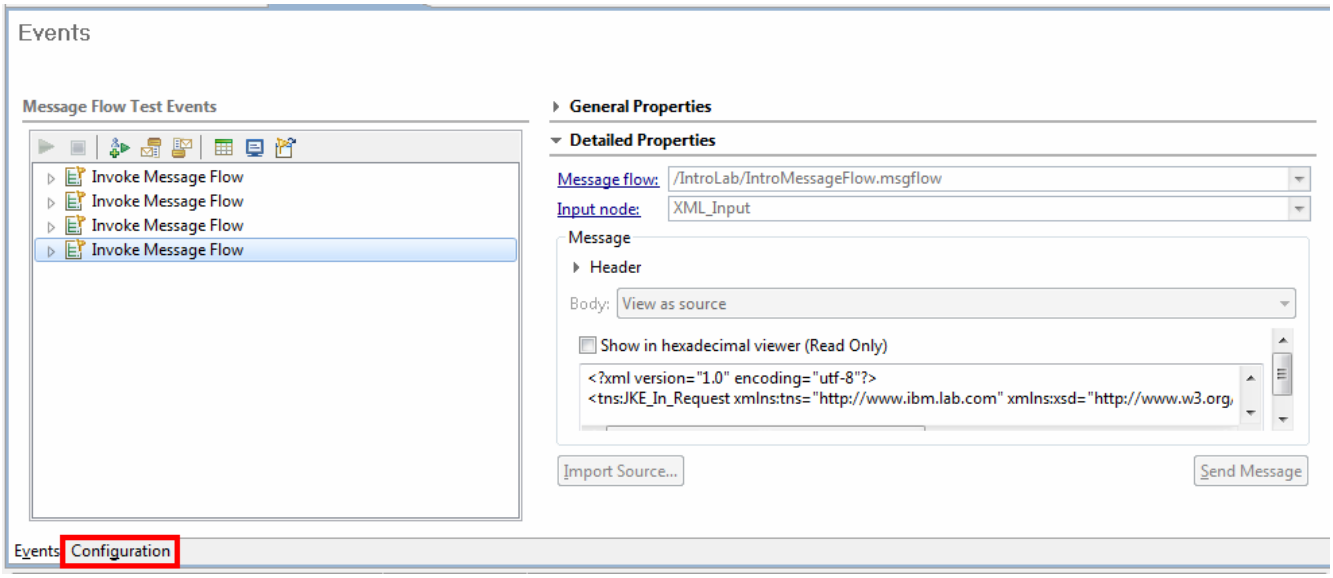
__6. In the main editor view, bring into focus the **IntroLab** Test Client.



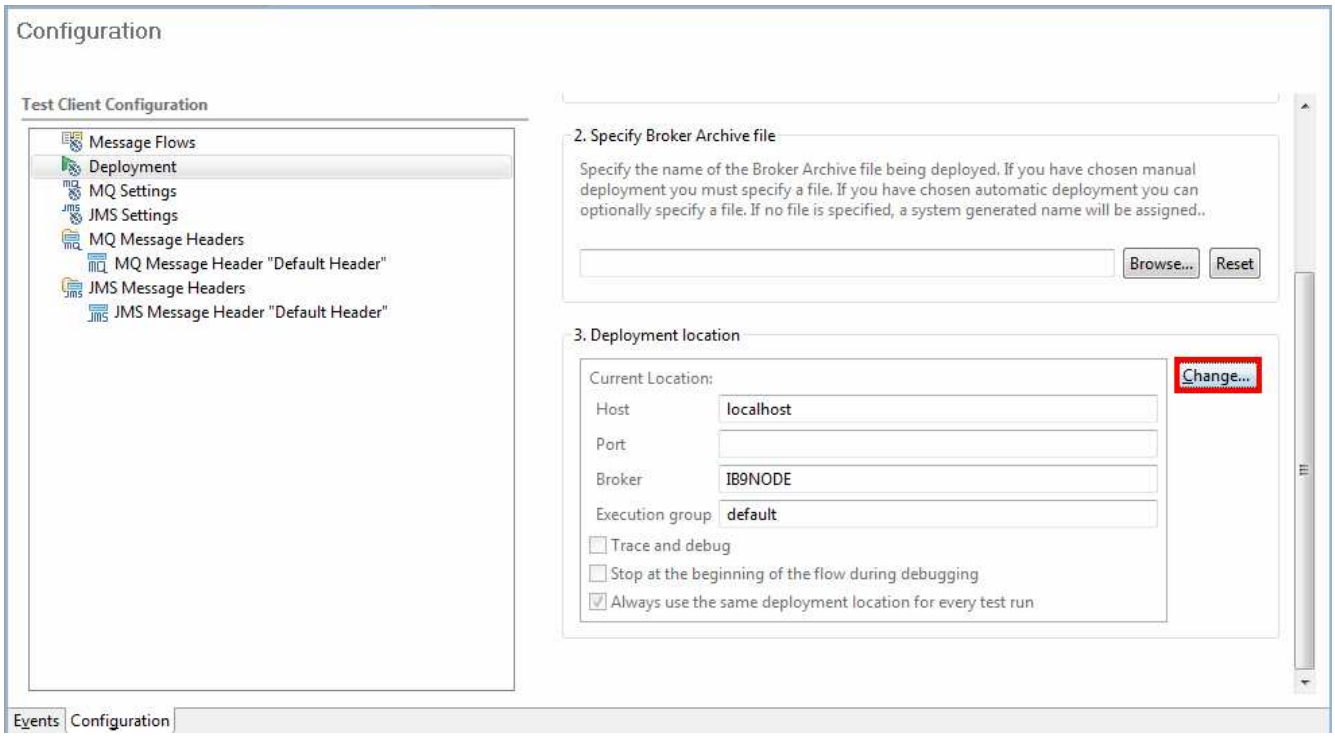
If you have closed IntroLab you will find the file on the left hand side.



__7. Along the bottom of the Test Client select the **Configuration** tab.

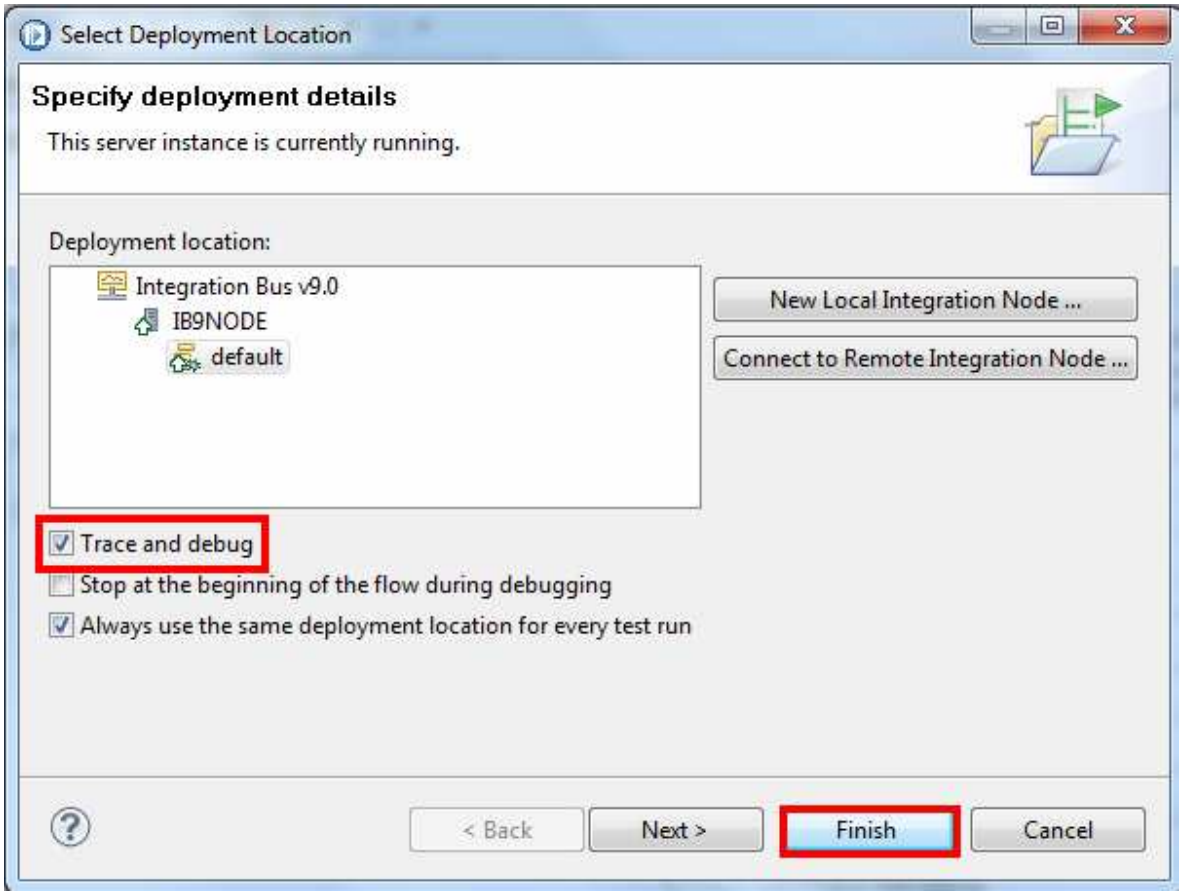


__8. Along the right within the Deployment Location section, click the box called **Change**.



__9. In the popup dialog, select the checkbox labeled **Trace and Debug**.

__10. Press the **Finish** button.

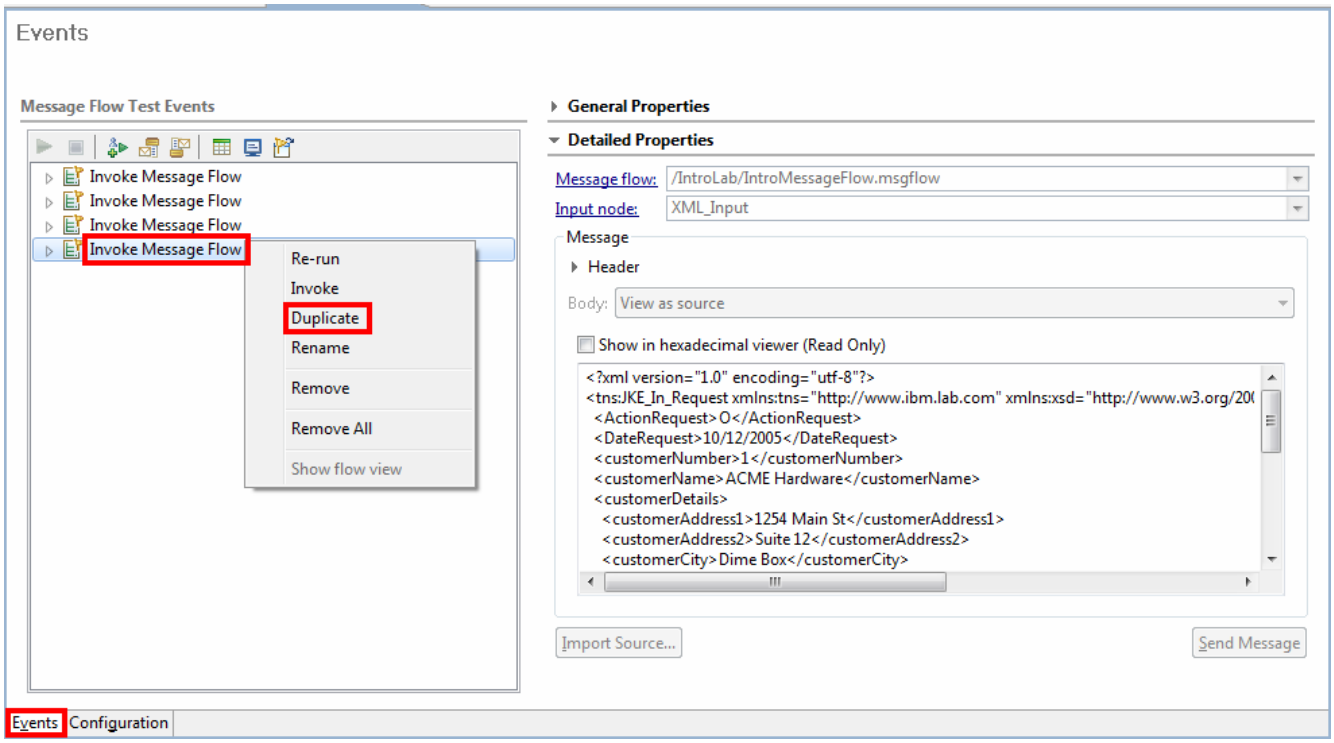


The debugger will now run next time the Test Client is run.

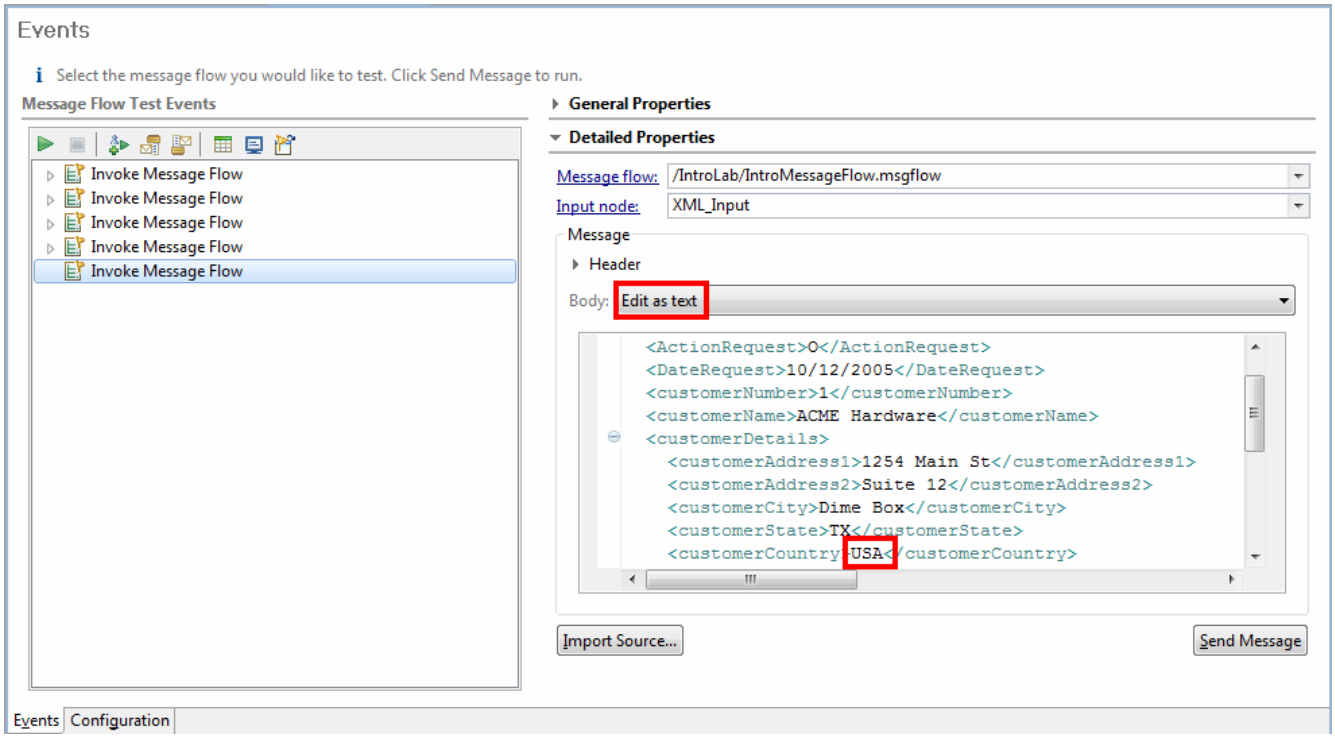
__11. Switch back to the **Events** tab.

__12. Right click on the last **Invoke Message Flow** in the Message Flow Test Events.

__13. Select **Duplicate** from the menu.



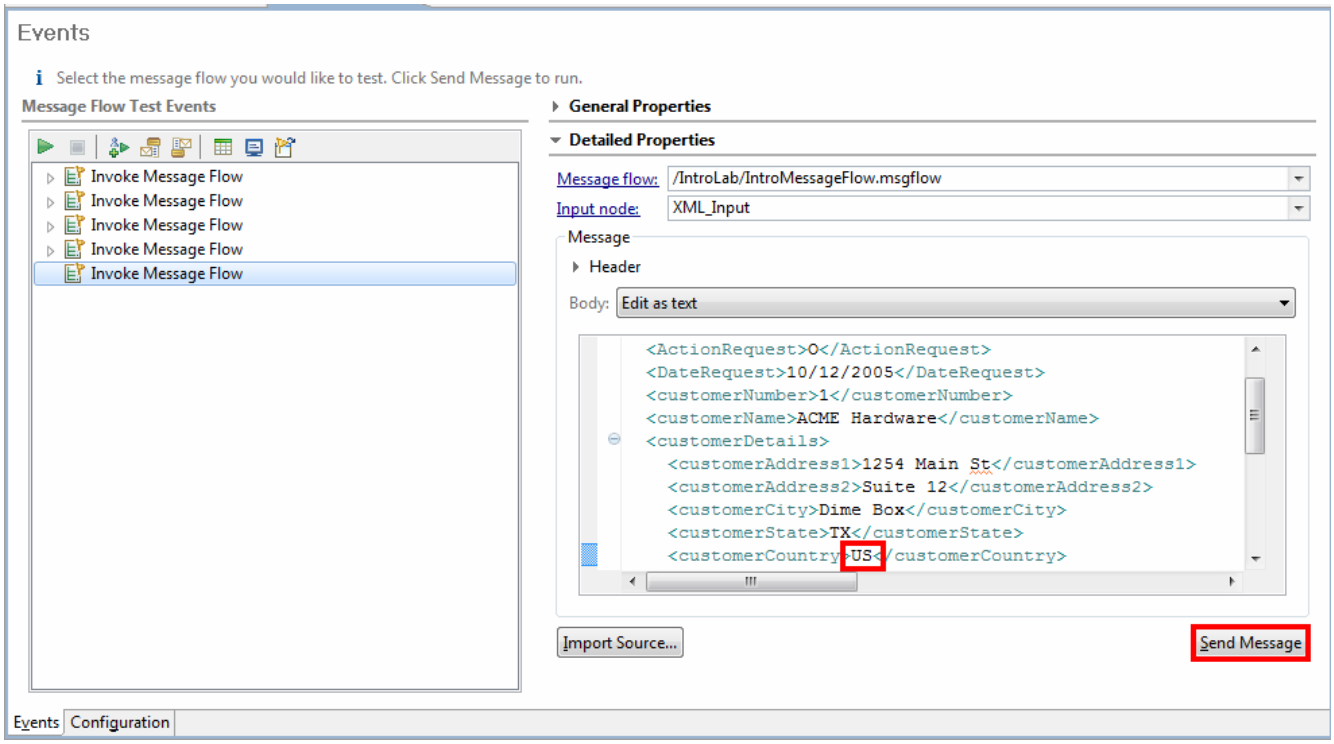
__14. In Detailed Properties, the Body should now show Edit as text. If not, use the drop down to select it.



__15. Edit the input message. Locate the customerCountry element.

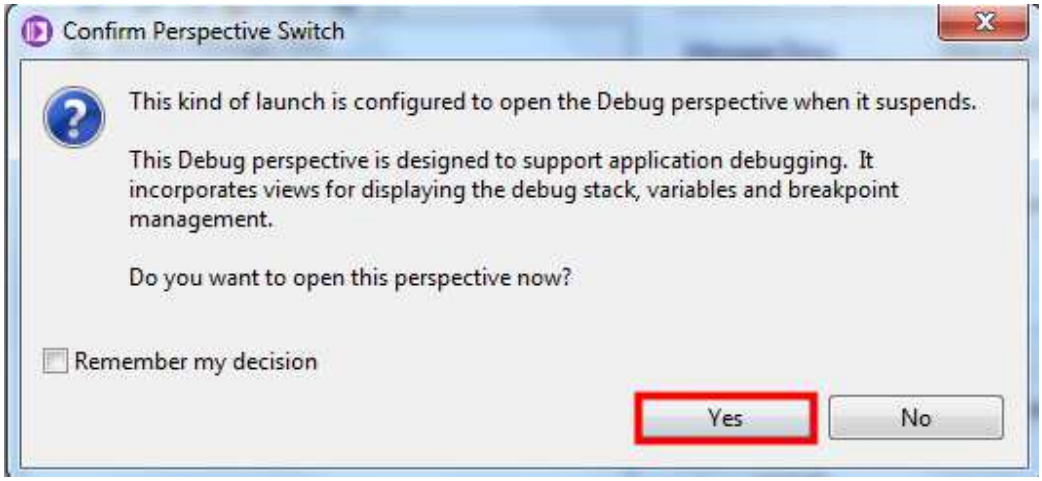
__16. Change the data value to **US**.

__17. Run the test client by clicking **Send Message**.

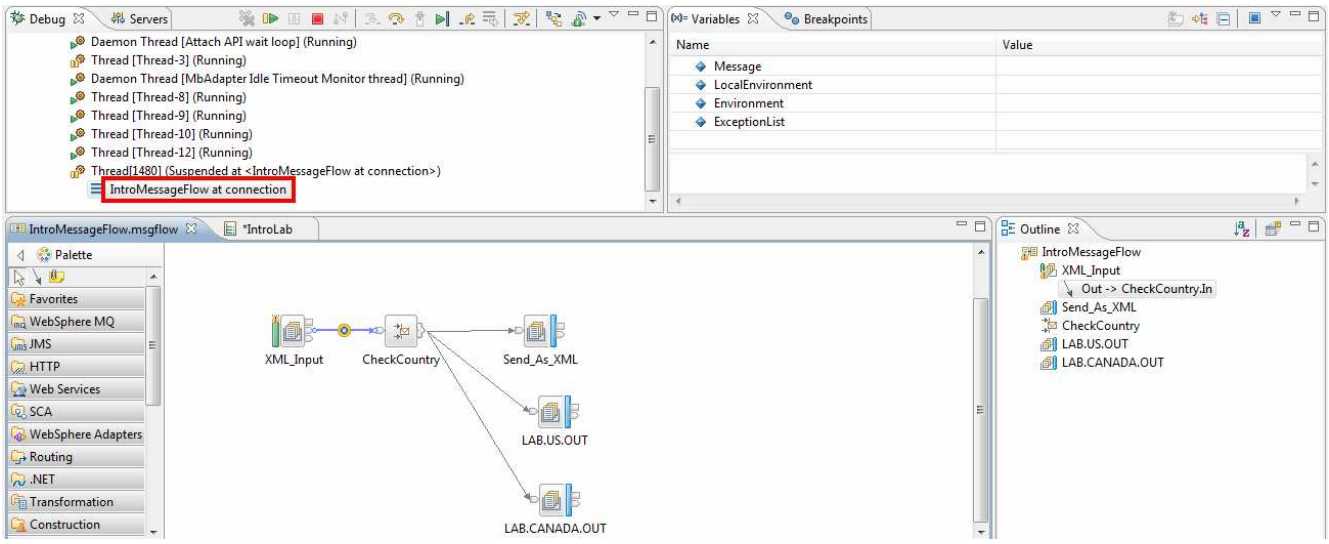


A popup will confirm switching the eclipse perspective to the Debug perspective.

__18. Press the **Yes**.

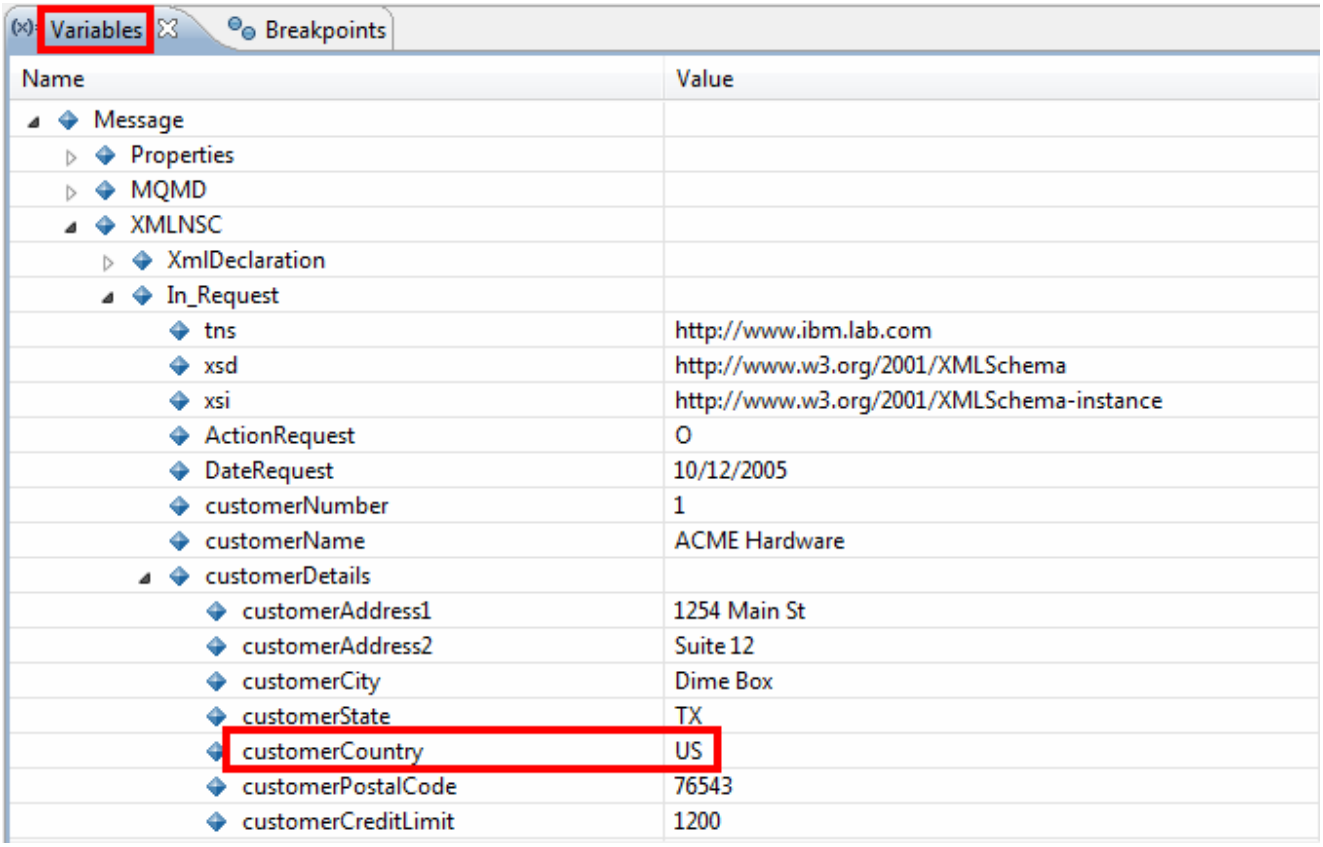


__19. Ensure that the **IntroMessageFlow** thread is selected in the top left view. You should see a yellow halo around the breakpoint. The yellow halo indicates that this is where execution of the flow has been suspended.



__20. In the top right Variables view, expand **Message→XMLNSC→In_Request→customerDetails**.

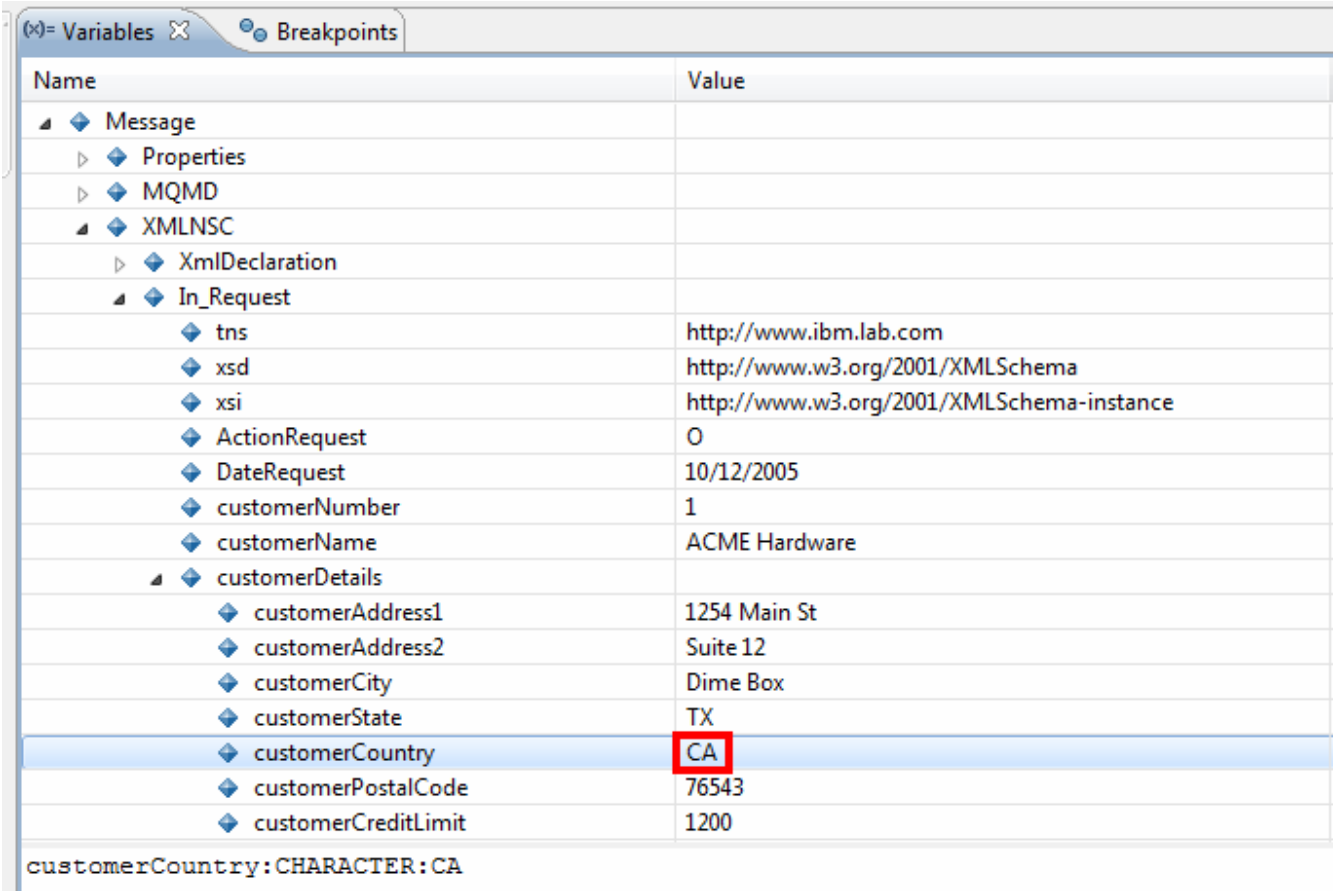
__21. You will see the value is set to **US**.



__22. To interactively change this, single click the **Value** column of the **customerCountry** element.

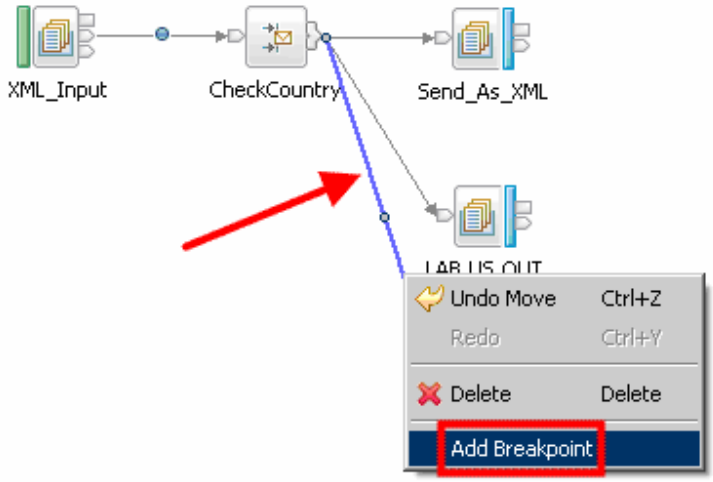
__23. Change the value in the editor box to **CA**.

__24. Press **Enter** to update the value.

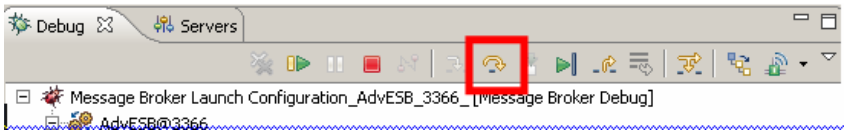


__25. Right click on the wire between the **CheckCountry** node and the **LAB.CANADA.OUT** node.

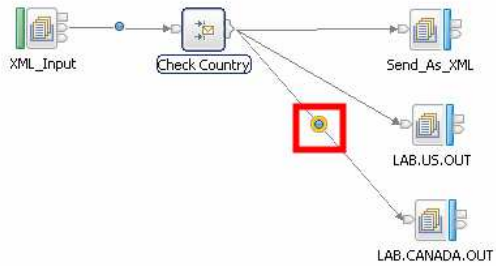
__26. Select **Add Breakpoint** from the menu.



__27. Select the **Step Over** button along the actions menu panel to run the Route node logic. Alternatively, you can press F6 to step over.



__28. Verify that the Route node has sent the message down the Canada terminal.

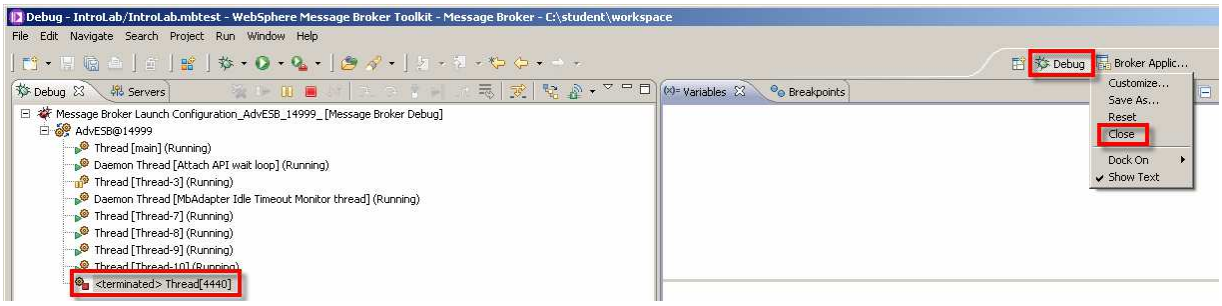


__29. Select Resume or Step Over to finish the flow (or press F8 or F6, respectively).

You can now re-run the test without overriding the customerCountry element in order to validate the path to the **LAB.US.OUT** queue. Return to the Integration Development perspective and the Test Client, and "Re-run" the last Invoke Message Flow.

__30. When finished testing the debugger (the debugger should be in a terminated state), right click on the **Debug** perspective.

__31. Select **Close** from the menu.



3.5 A Closer Look at the Deployment Process

Up to this point, we have been utilizing the Test Client to initiate our unit testing and it has been handling the deployment process for us “under the covers”. To finish this lab, we will briefly examine the deployment process and manually do our own deploy.

Key Idea: The Deployment Process

When you create application resources such as message flows in the IBM® Integration Toolkit, you must distribute them to the nodes on which you want them to run. Data for message flows and associated resources is packaged in a broker archive (BAR) file before being sent to the node.

You can initiate a deployment in the following ways:

- From the IBM Integration Toolkit
- From the Integration Explorer
- By using the **mqsideploy** command
- By using functions defined by the Integration Bus CMP API

Depending on your work patterns, you might use all these methods at different times.

The Integration Toolkit provides a Nodes view in the lower left hand corner of the Integration Development perspective. If you expand a integration node, all the integration servers in that node are displayed, as well as deployed message flows and their associated resources. You can drag an Application or Library, message flow, or a broker archive (BAR) file from the Application Development view onto an execution group to deploy it. Alternatively, you can right click on an execution group to select an Application or Library, message flow, or BAR file to deploy to the selected execution group.

If you are working with an application and want to deploy and test it quickly, you can deploy just that resource. Drag the resource onto the execution group to which you want to deploy it. A BAR file is generated automatically, and deployed to the node. If libraries are referenced, they are added automatically to the BAR file and deployed. If a message flow contains a subflow that is defined in a “.subflow” file, the subflow is automatically included in the BAR file, and deployed with the message flow. If you drag a flow that is contained in an Application or Library, you will see a message saying that the whole application or library will be deployed, because you cannot deploy a message flow on its own if it belongs to an Application or Library.

Key Idea: The BAR file

The unit of deployment to a Broker is a BAR file. It is a .zip file which contains the flows, models, .jar files, maps, and any other resources in the workspace needed to run your Applications. The BAR file also contains a deployment descriptor .xml file, which exposes flow and node properties for override at build or deploy time. The following sequence of events illustrates how to deploy with a BAR file:

1. Create a broker archive.
2. Add files to the broker archive.
3. If necessary, you edit the configurable properties of the message flows or applications in the broker archive.
4. Deploy the BAR file by sending it to the broker, from where its contents are distributed to the integration servers.

A BAR file can be deployed in two ways:

- Incremental BAR file deployment. Deployed files are added to the execution group. Files that exist in the execution group are replaced by the new version.
- Complete BAR file deployment. Files that are already deployed to the execution group are removed before the entire contents of the BAR file are deployed. Therefore, nothing is left in the execution group from previous deployments.

__1. Return to the Integration Development perspective in the toolkit.

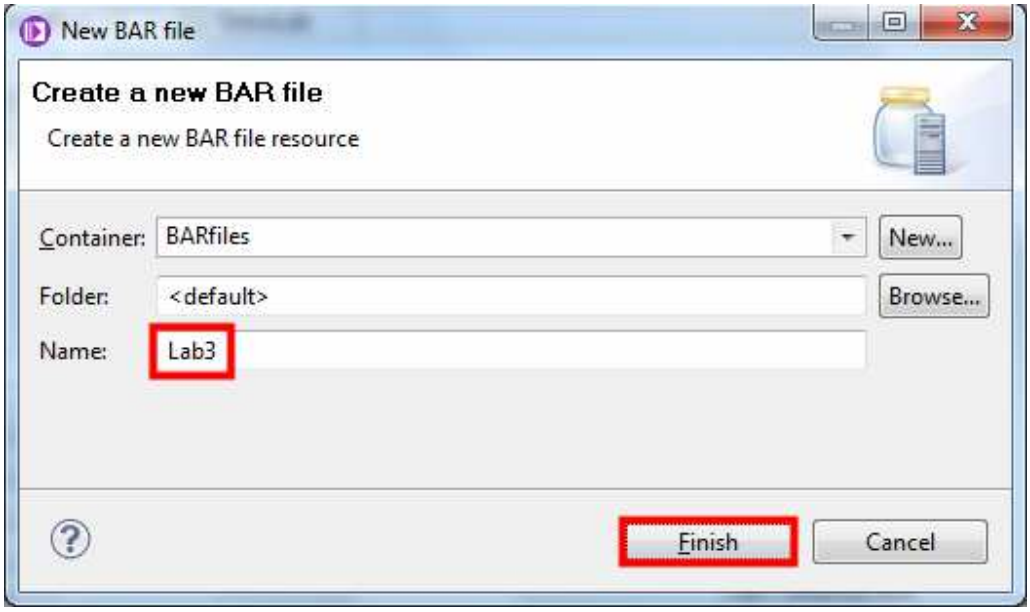
__2. Select a blank area in the **Navigation** view on the left.

__3. Press the right mouse button.

__4. Select **New→BAR** file from the menu.

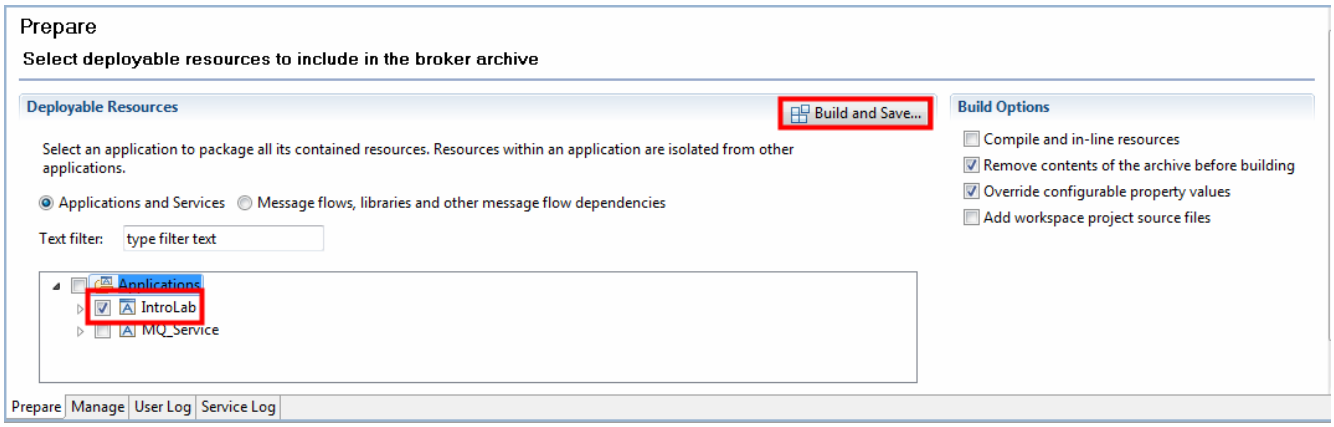
__5. Enter **Lab3** as the name of the new broker archive file.

__6. Select **Finish**.



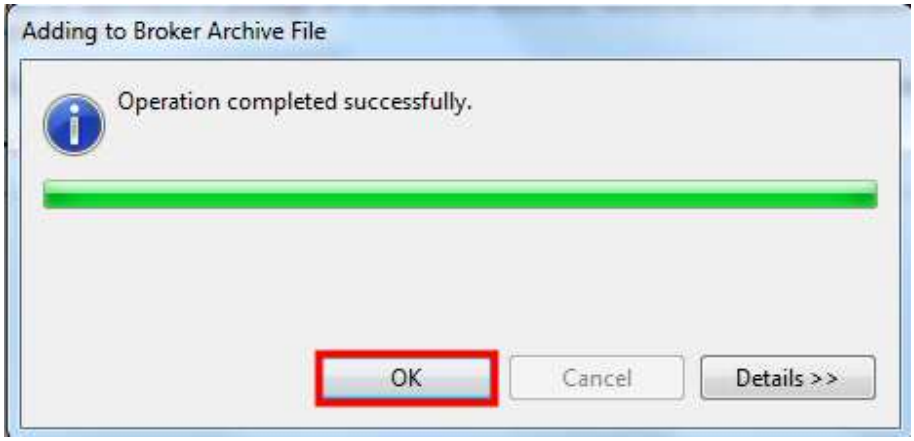
__7. In the BAR editor, select the **IntroLab** Application.

__8. Click the **Build and Save**… button.



__9. If prompted to Save, click **Yes**.

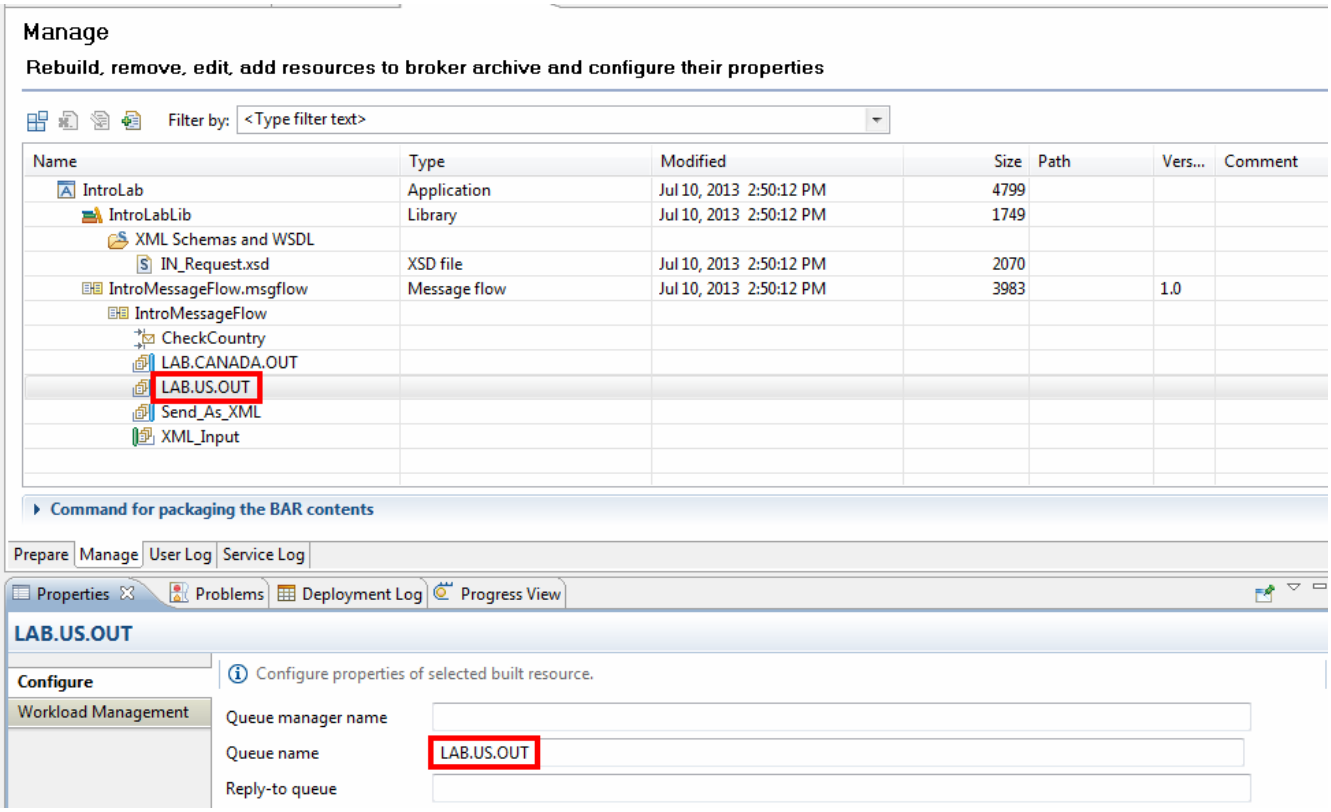
__10. Press the **OK** button.



__11. In the BAR editor, select the **Manage** tab.

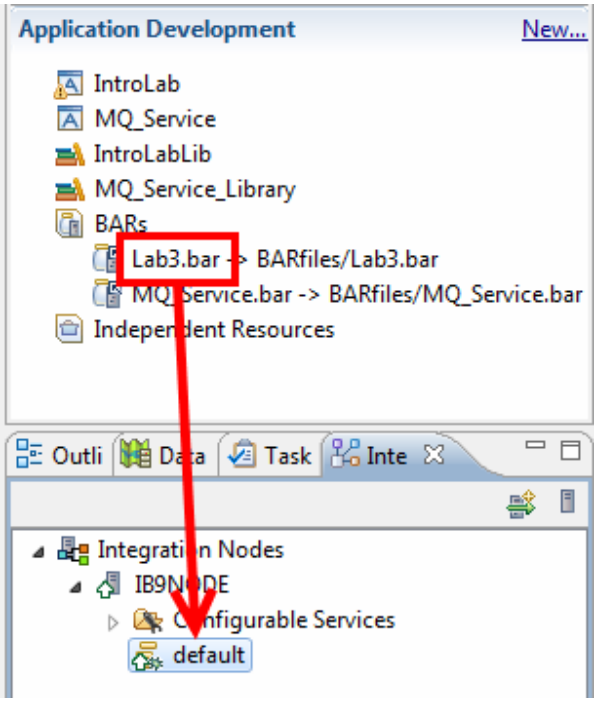
__12. Expand the **IntroLab** app and select the various resources.

__13. Look at the **Properties view** below to see what properties are exposed in order to be overridden within the BAR file. For example, select the **LAB.US.OUT** node. You can see that the Queue Name property can be overridden at deployment time.

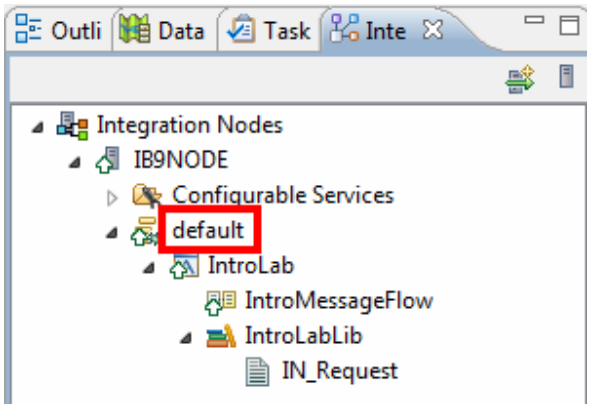


__14. To deploy, find the **Lab3.bar** file in the navigator in the **BARs** container.

__15. Drag and drop it onto the **default** integration server of the **IB9NODE**.



__16. Once deployment is complete, click the **default** integration server to open it to see the assets deployed to it. The IntroLab application has been deployed, which included the Intro_MessageFlow message flow and the library was included, which contains the IN_Request XML Schema.



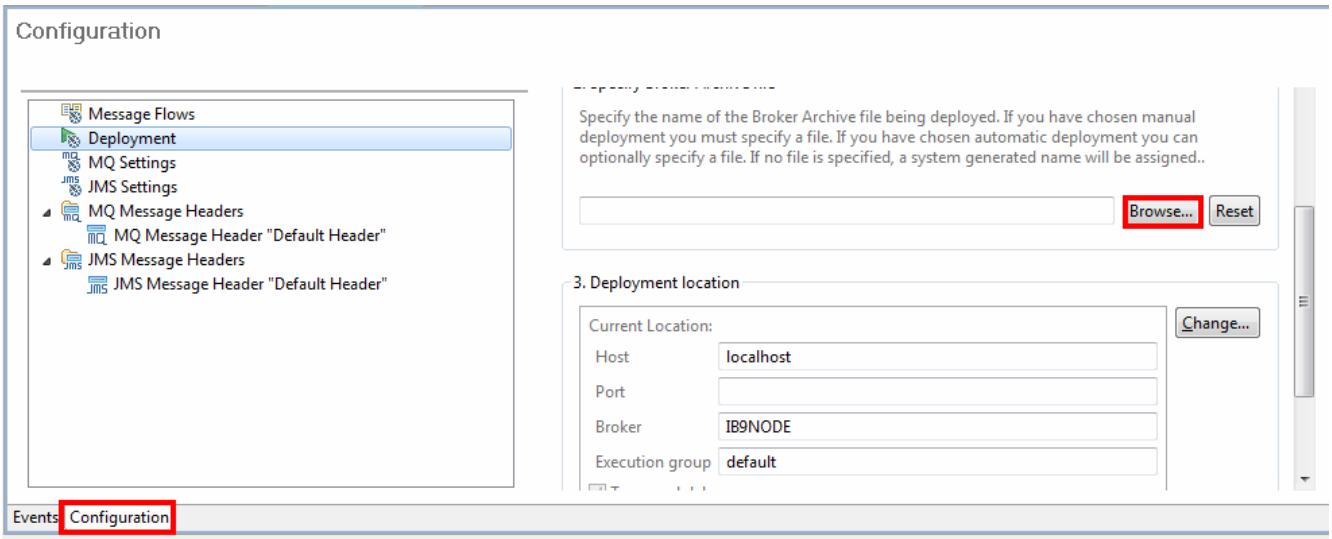
Finally, the Test Client will be updated to use this BAR rather than generating its own.

__17. In the editor, select the **Test Client**.

__18. Select the **Configuration** tab.

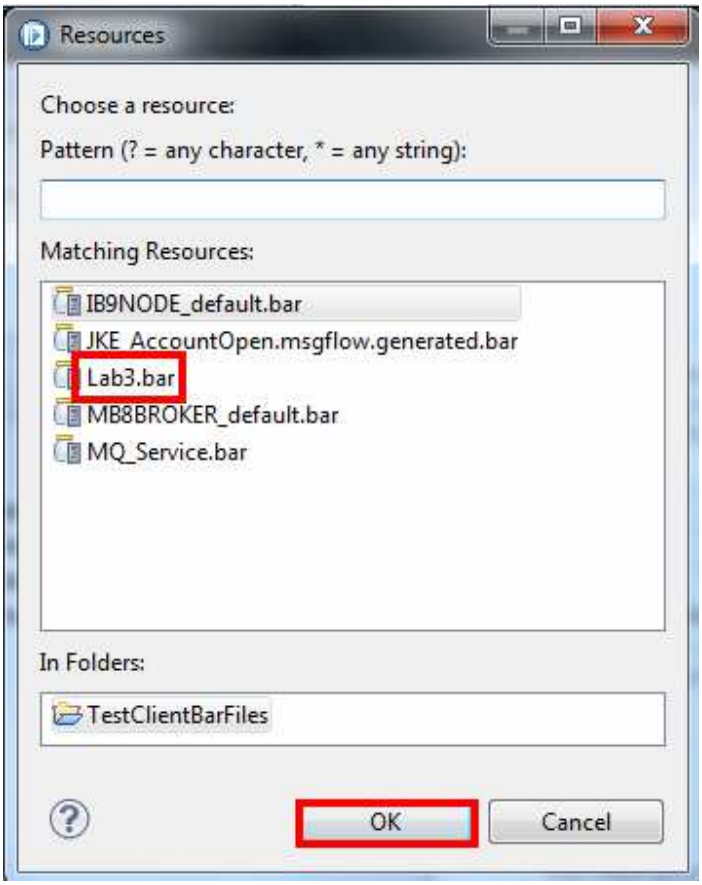
__19. Select the **Deployment** tab.

__20. On the right inside the **Specify Broker Archive** box, select **Browse**.



__21. In the dialog box, select **Lab3.bar.**

__22. Click **OK**.



__23. 

Save the Test Client.

__24. Close all the open editor tabs but leave the toolkit running.

This is the end of Lab 3.