

WebSphere Lab Jam

Connectivity

WebSphere Message Broker

Lab Exercises



An IBM Proof of Technology

Catalog Number

Contents

LAB 1	GETTING STARTED WITH IBM WEBSHERE MESSAGE BROKER V7.0	5
	1.1 INTRODUCTION TO THE TOOLKIT AND BUILDING MESSAGE FLOWS.....	5
LAB 2	DEPLOYING AND EXECUTING A MESSAGE FLOW	23
	2.1 OVERVIEW	23
LAB 3	WORKING WITH XML MESSAGES	39
	3.1 OVERVIEW	39
LAB 4	WORKING WITH FIXED FORMAT (COBOL) MESSAGES	46
	4.1 OVERVIEW	46
LAB 5	BUILDING THE JKE SERVER – WEB SERVICES	70
	5.1 LAB OVERVIEW.....	70
LAB 6	BUILDING THE JKE SERVER – WEB SERVICES CLIENT	92
	6.1 OVERVIEW	92
	6.2 DEPLOYING THE JKE_CLIENT – WEB SERVICE CLIENT MESSAGE FLOW	93
	6.3 ADDING A WEB SERVICE REQUEST TO THE JKE_SERVER MESSAGE FLOW	96
LAB 7	BUILDING THE JKE SERVER – ROUTING	103
	7.1 LAB OVERVIEW.....	103
LAB 8	BUILDING THE JKE SERVER - MAPPING	141
	8.1 OVERVIEW	141
APPENDIX A.	NOTICES	129
APPENDIX B.	TRADEMARKS AND COPYRIGHTS	131

THIS PAGE INTENTIONALLY LEFT BLANK

Lab 1 Getting Started with IBM WebSphere Message Broker V7.0

1.1 Introduction to the Toolkit and Building Message Flows

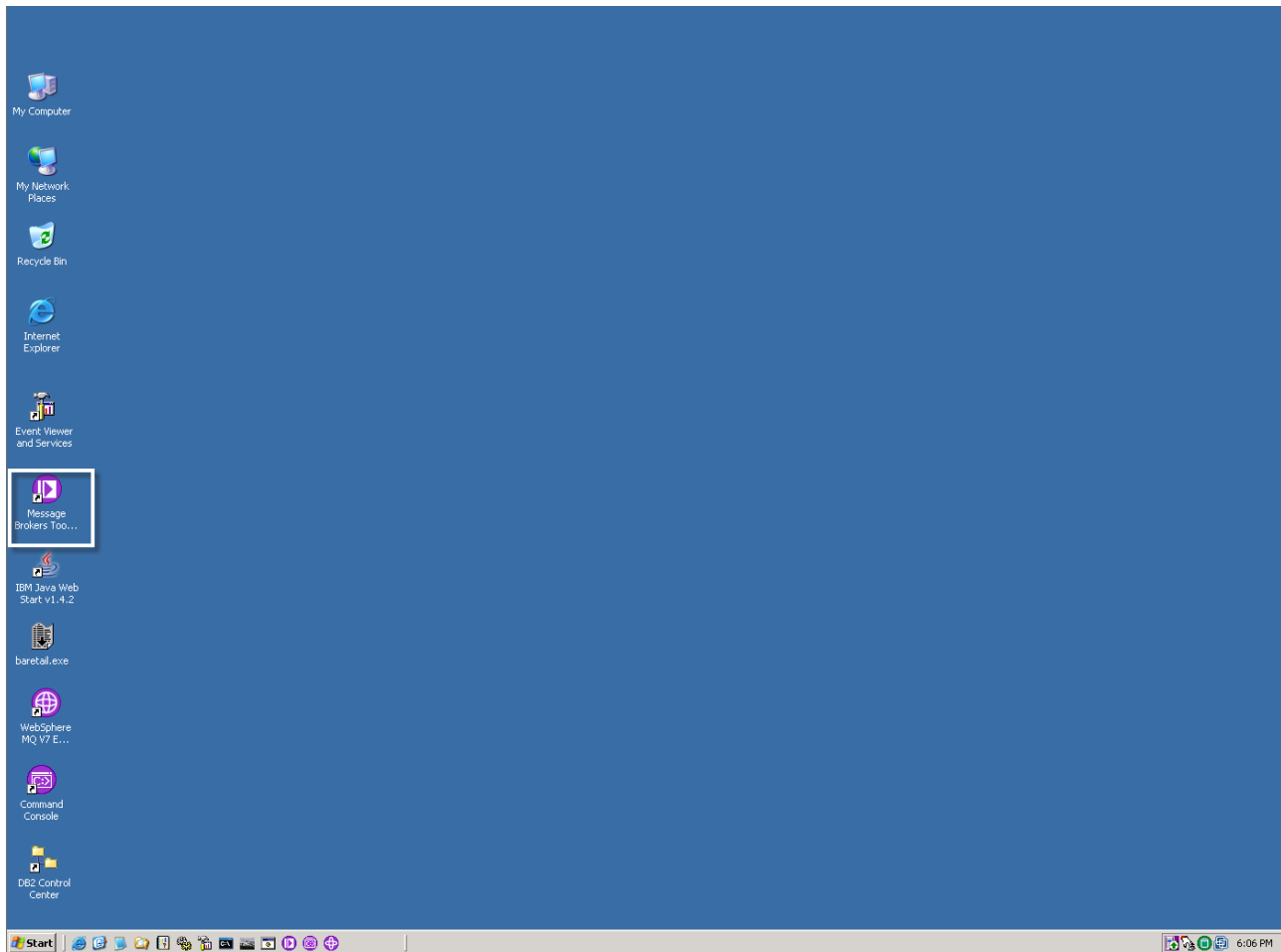
In this lab you will build a simple message flow.

As a convention for these labs, a red box will be used to identify a particular area, and when information is to be entered and/or an action is to be taken, it will be in **bold** characters. Red lines may be used to indicate where nodes are to be placed when building your message flow.

Additional icons are provided in the quick launch area. These are:

- Desktop
- Internet Explorer
- Notepad
- Windows® Explorer
- Display Event Logs and Services
- IBM® DB2® Command Center
- Windows command window
- RfhUtil (from SupportPac™ IH03)
- A link to NetTool (Open Source, also good for testing HTML and SOAP messages)

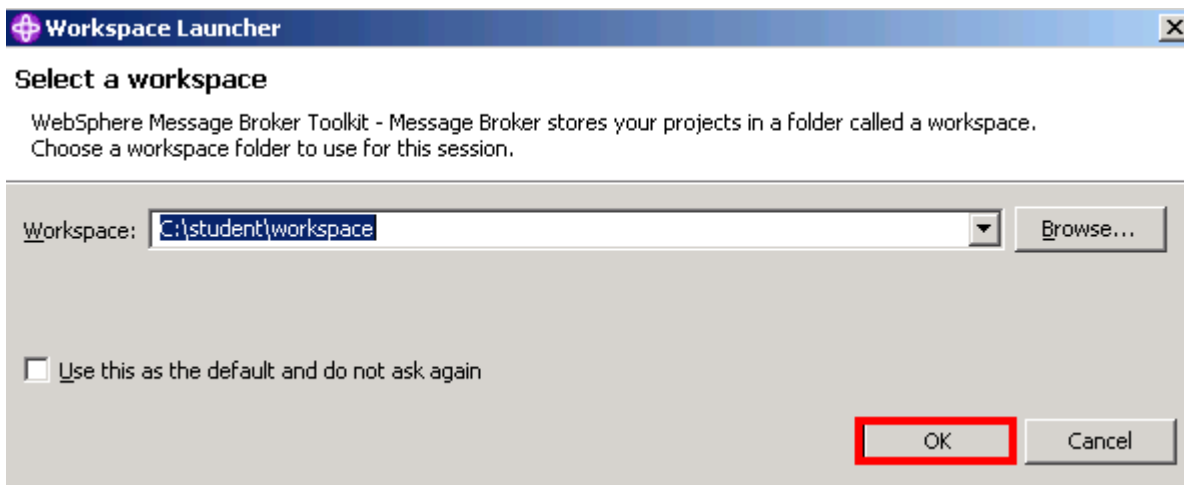
The system is Windows XP, Service Pack 2 (SP2). The IBM software includes IBM WebSphere® MQ V7.0.1, WebSphere Message Broker V7.0, DB2 V8.2 Fix Pack 14 (FP14), WebSphere Application Server V6.1 Fix Pack 19 (FP19) and WebSphere Service Registry and Repository V6.2. Other IBM software is also installed and will be described in the later labs.



The icon for the IBM WebSphere Message Broker Toolkit is located on the desktop. In later labs, you will also be using some of the icons that have been placed on the smart bar. **Double click on the icon** to start the WebSphere Message Broker Toolkit.

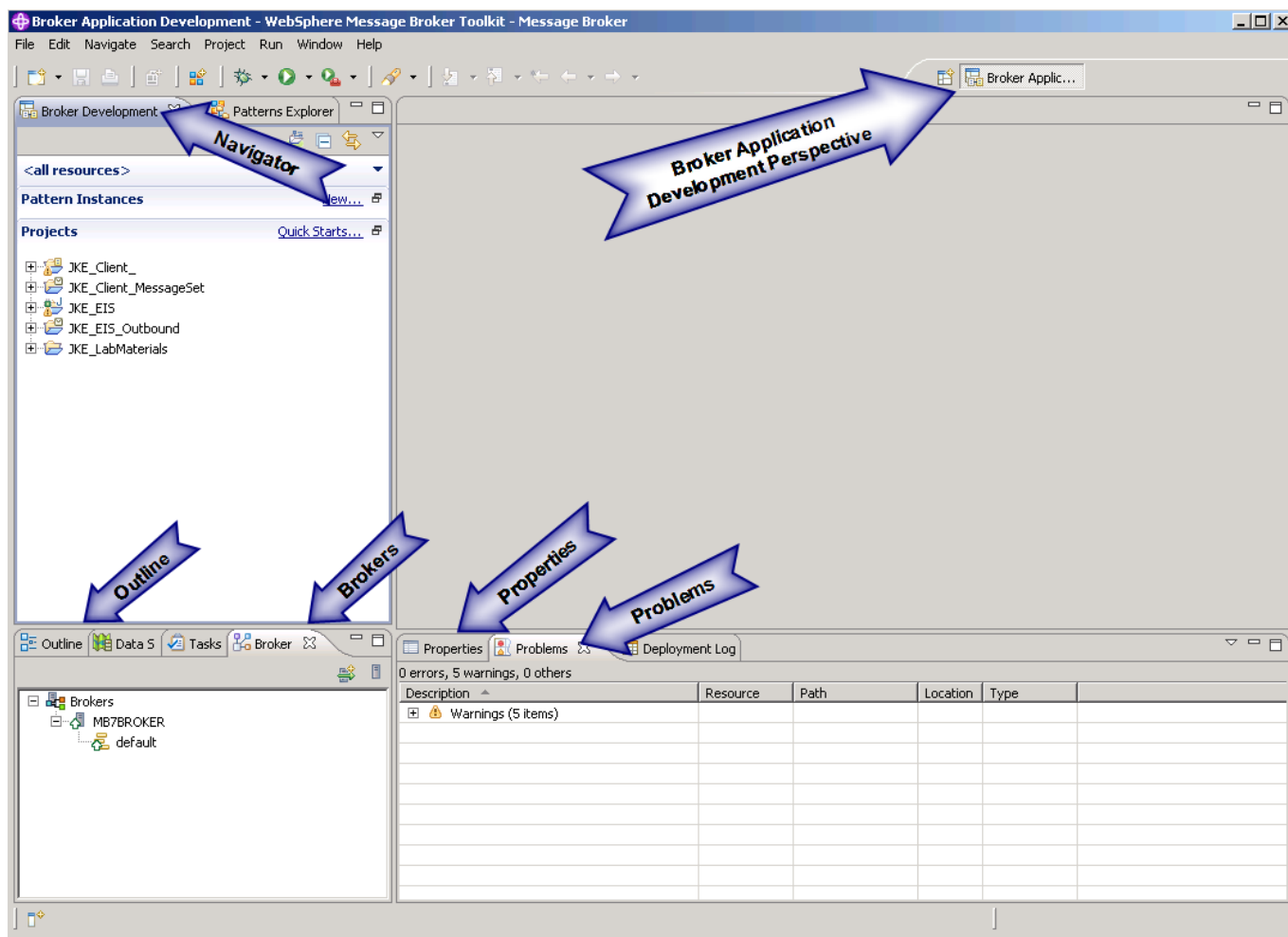


The above splash screen is displayed when starting the WebSphere Message Broker Toolkit



You are prompted to select an Eclipse Workspace – you will take the default here but this is a nice facility to allow you switch between workspaces when starting the WebSphere Message Broker Toolkit

__1. Click **OK**.



This is the WebSphere Message Broker V7.0 Toolkit. It is based on Eclipse and includes components from IBM Rational® Application Developer V7. It provides one Perspective specifically for WebSphere Message Broker as well as additional Perspectives from Rational Application Developer and Eclipse. This system is using the default installation. During the labs and lectures you will be learning more about the components in a typical development and runtime environment.

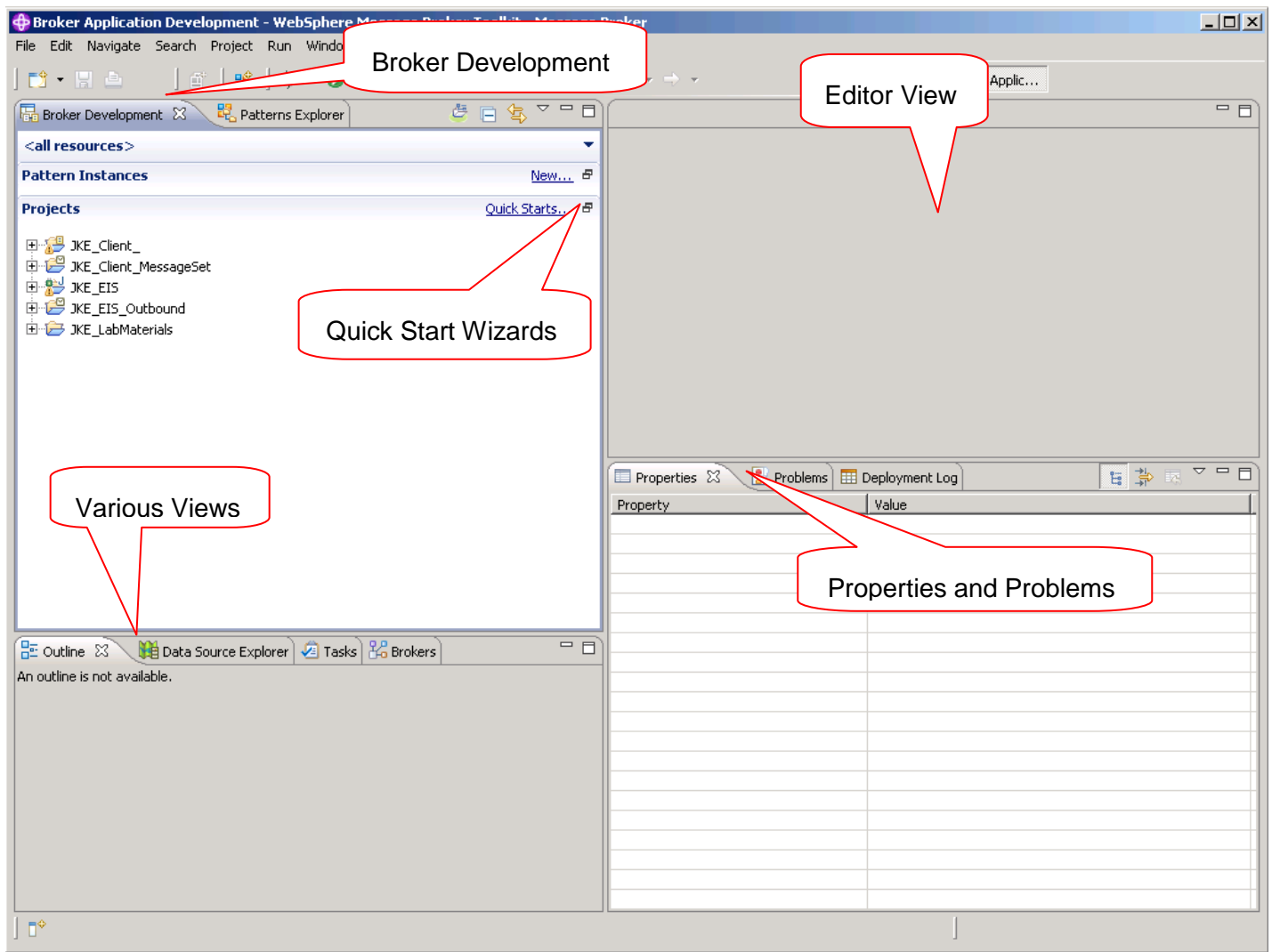
This is the **Broker Application Development** Perspective. It is divided into multiple views (or panes). Each view is identified by a tab at the top of the view. On the lower left is the Outline view

On the upper left is the Navigator view, which has tabs for projects (**Broker Development**) and patterns. It contains the projects that are available within the Eclipse workspace. There is a set of resources provided for your use during the labs.

The area below the navigator view is the summary area. The **Broker** tab will show all defined local brokers as well as connections to remote brokers that have been created.

The large area on the right is used by the resource editors. When an editor has opened a resource, it will also be represented by a tab. Below the editor view is a pair of views for Properties and Problems.

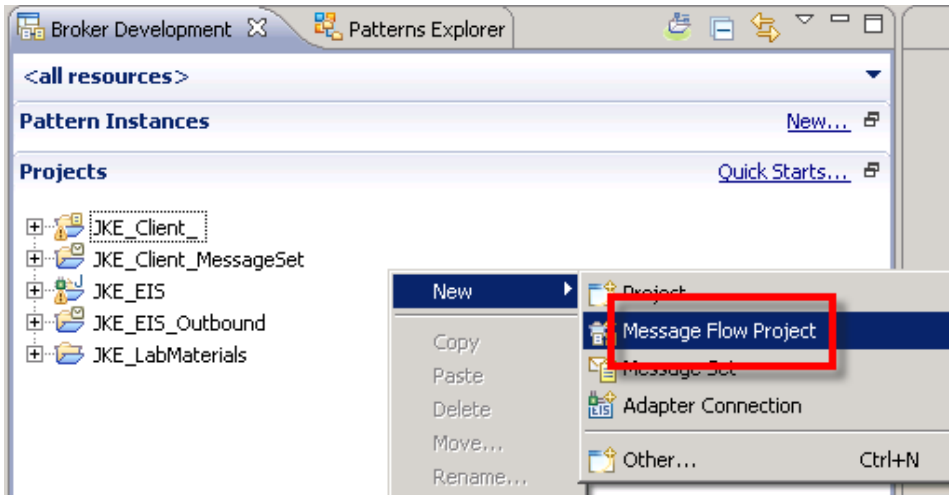
On the top right are tabs for the perspectives that have been opened. To change an open perspective, you can simply click on its tab.



Quick start wizards can be viewed using the drop down menu in the navigator pane on the **Broker Development** tab. The wizards do some of the initial work for creating various types of solutions.

The WebSphere Message Broker Toolkit provides seven quick start wizards plus a number of pre-defined patterns to assist in creating message flows and message sets. Two of these will be used in the later labs.

Eclipse is project oriented – artifacts are organized into projects. A project is typed. That is to say that a particular type of project can only contain a certain kinds of artifacts. For example, you now need to create a message flow. Before you can do that you must create a Message Flow Project to hold that artifact.

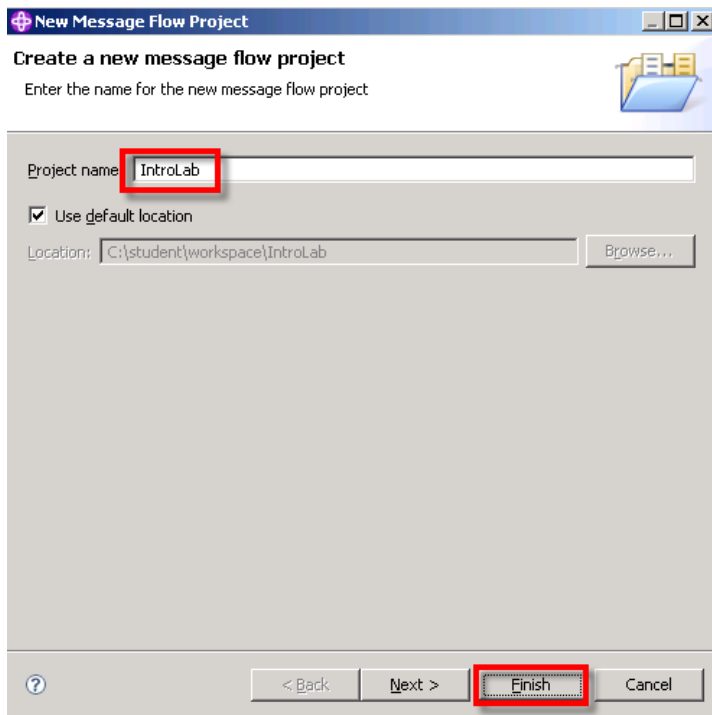


__2. Right click in the white space of the Broker Development pane.

__3. Select **New->Message Flow Project**.

As an alternative, you can select **File** from the menu bar, then **New**, then **Message Flow Project**.

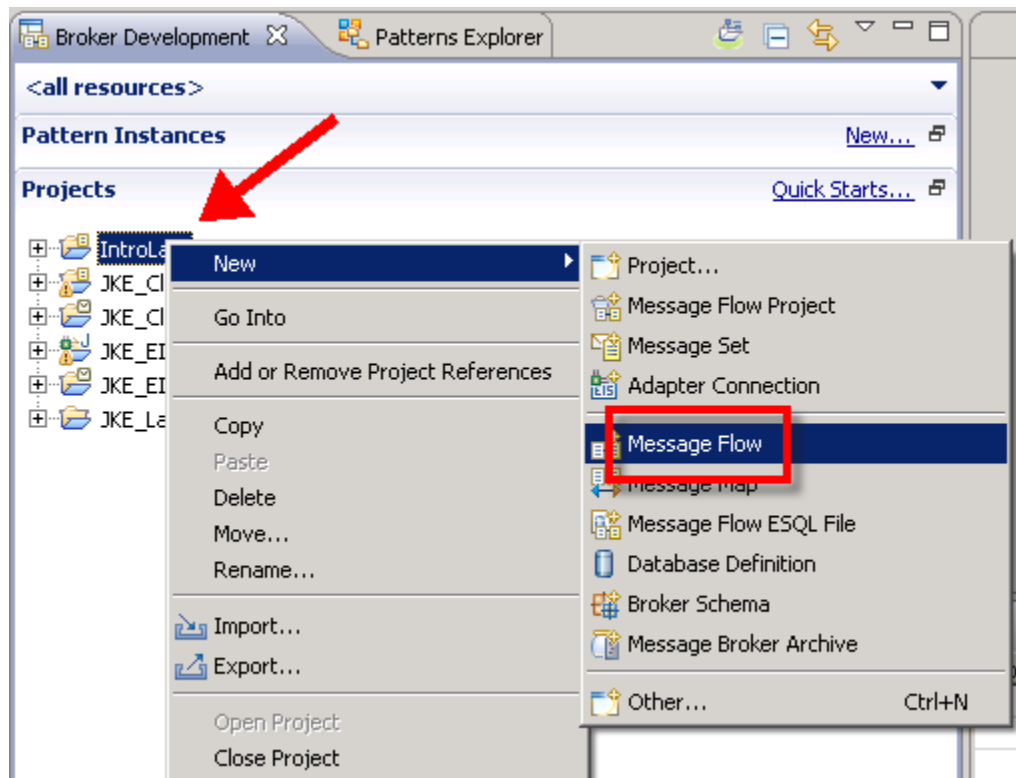
Note: These actions are also available as icons on the toolbar.



You are prompted to enter a name for your Project.

__4. Enter **IntroLab** for the Project name.

__5. Click **Finish**.

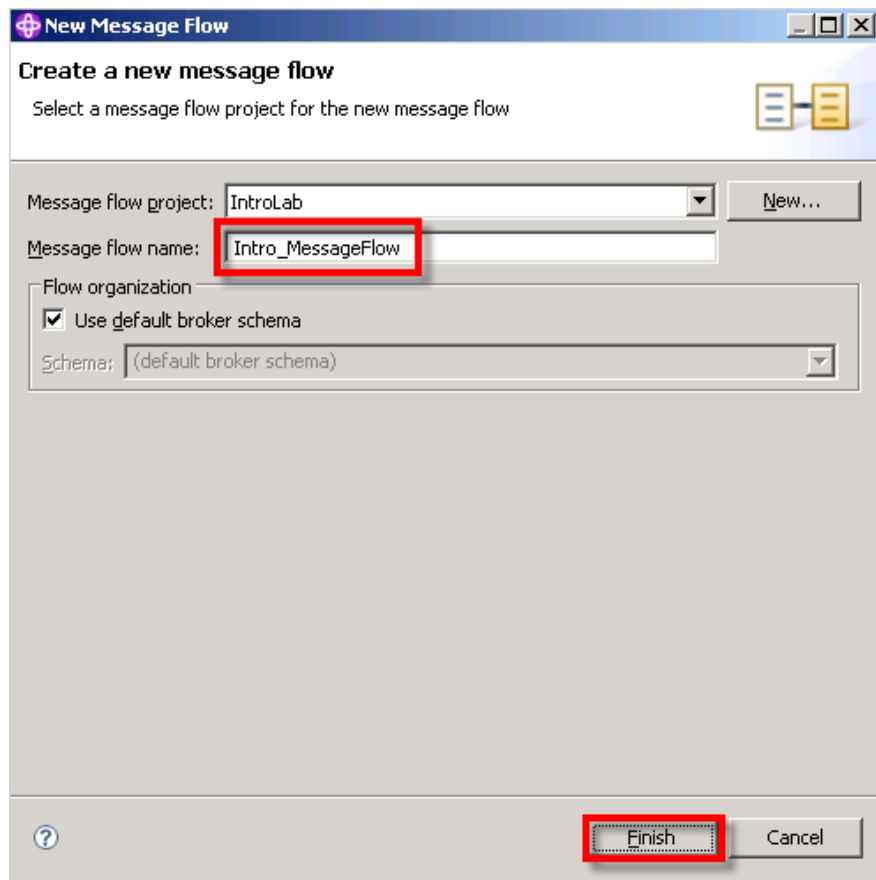


Now you will create a new message flow.

__6. Right click on **IntroLab**.

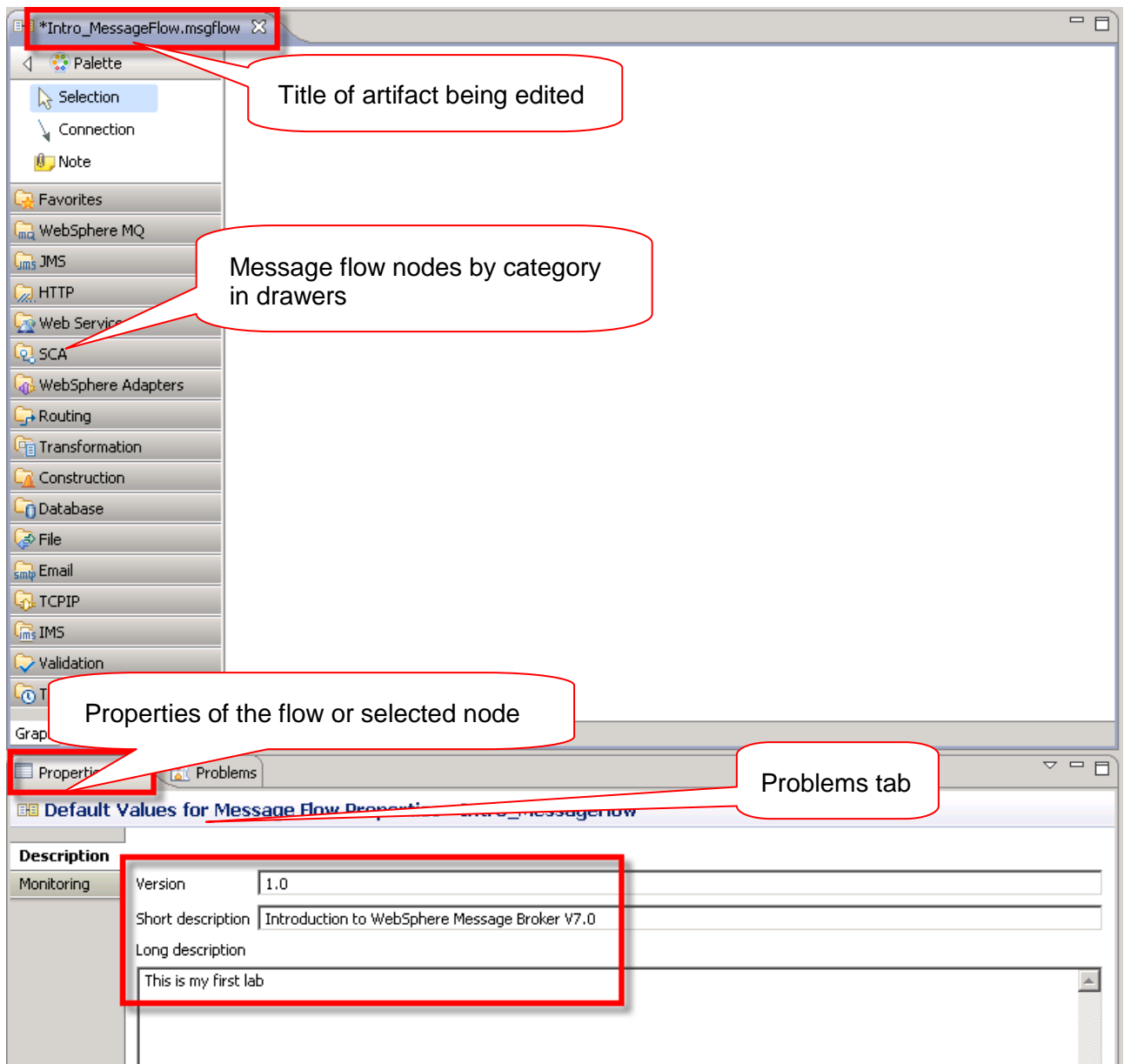
__7. Select **New->Message Flow**.

The options for the New action will be different depending on how the request is made. For example, when using the File → New from the Menu bar, all of the options will be listed. However, in this case, by starting from a Message Flow Project, the only options shown are those that are related to the selected project.



Here you are asked to name the message flow. A Message Flow Project may contain multiple message flows.

- __8. Enter **Intro_MessageFlow** in the Message flow name box.
- __9. Click **Finish**.



You are placed into the Message Flow Editor where you can compose the message flow. When you click on the Message Flow Editor, information about the message flow appears in the Properties pane.

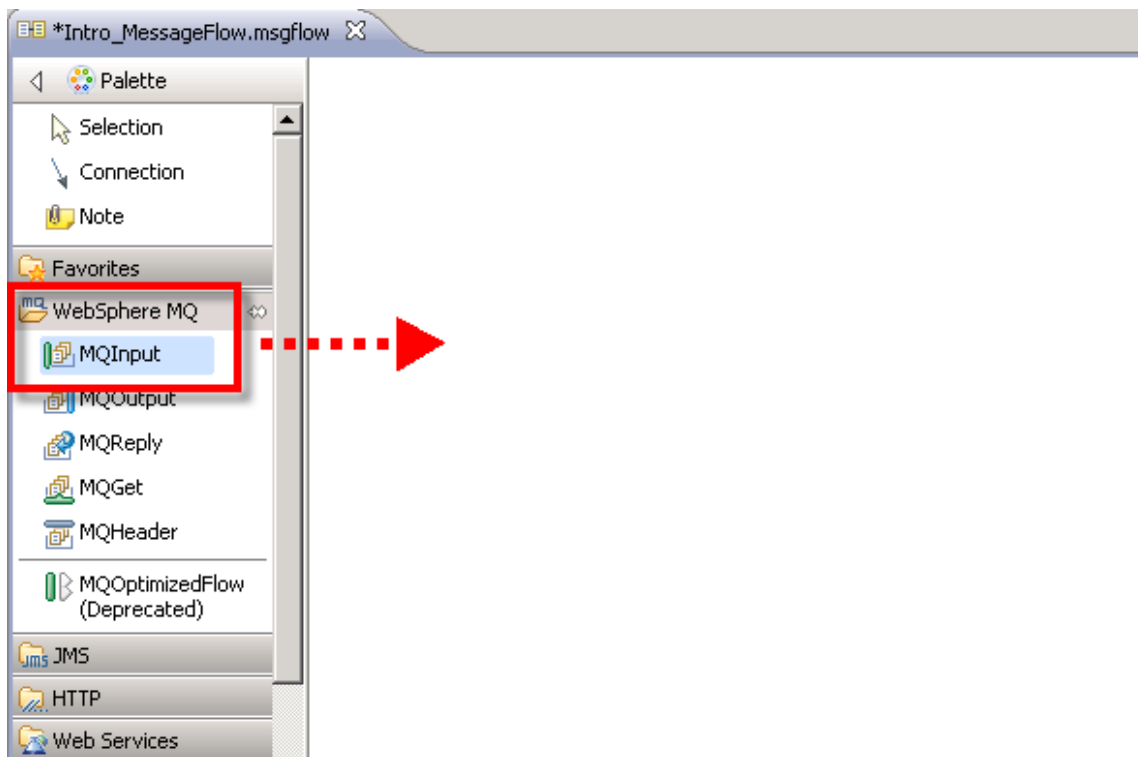
__10. Select the **Properties** tab.

__11. Enter the following:

Enter **1.0** for the **Version** field;

Enter Introduction to WebSphere Message Broker V7.0 for the Short description field;

Your choice of information in the **Long description** field.



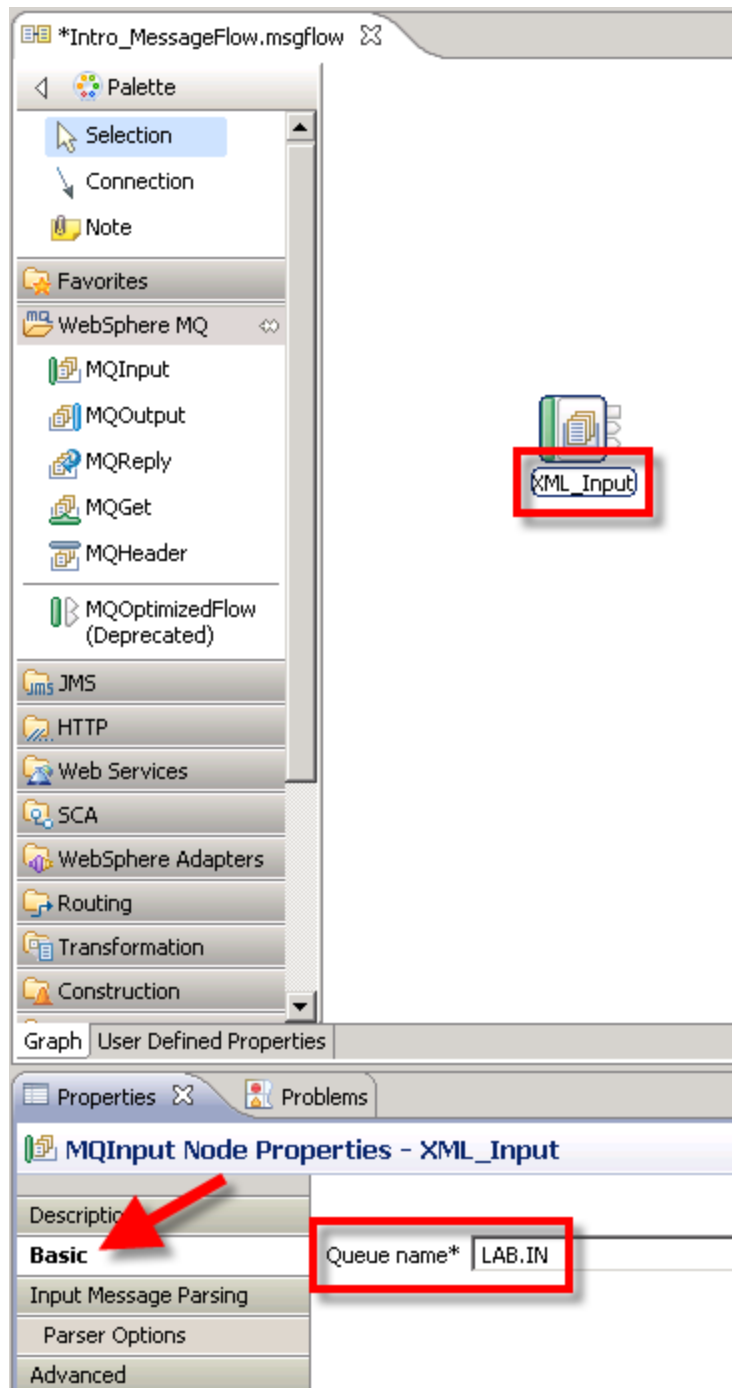
A message flow must begin with an Input node as this particular class of node establishes the environment for the flow. There is an Input node for each of the various protocols that Message Broker supports as well as a matching Output node. We will process a WebSphere MQ message with this flow so we need an MQInput node.

The **MQInput** node is in the WebSphere MQ drawer.

- __12. Open the **WebSphere** MQ drawer by clicking on it. If a drawer is open it will close when clicked.
- __13. Highlight the desired node (**MQInput**).
- __14. Either drag it to the canvas or move the mouse to the canvas and click again.

When a node is initially added, its name can be changed immediately by over-keying the default name – or – by entering a new value in the Node name field in the Description tab of the Properties.

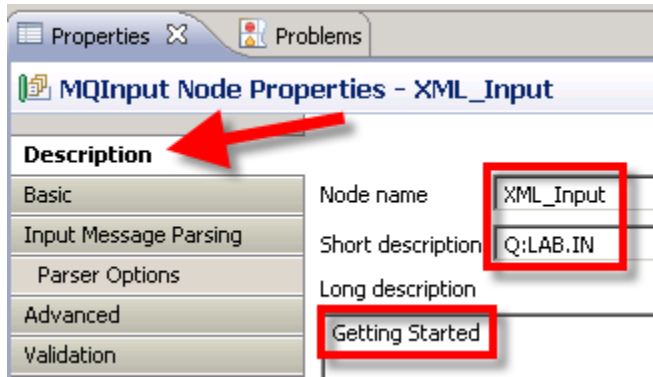
A “best practice” is to provide a new name for each node that is descriptive of the function that it provides. For most of the labs, you will be renaming the nodes. If you use the names as suggested it will make it easier to follow the lab guide. Another “naming convention” for MQInput and MQOutput nodes is to use the queue name that the node is accessing.



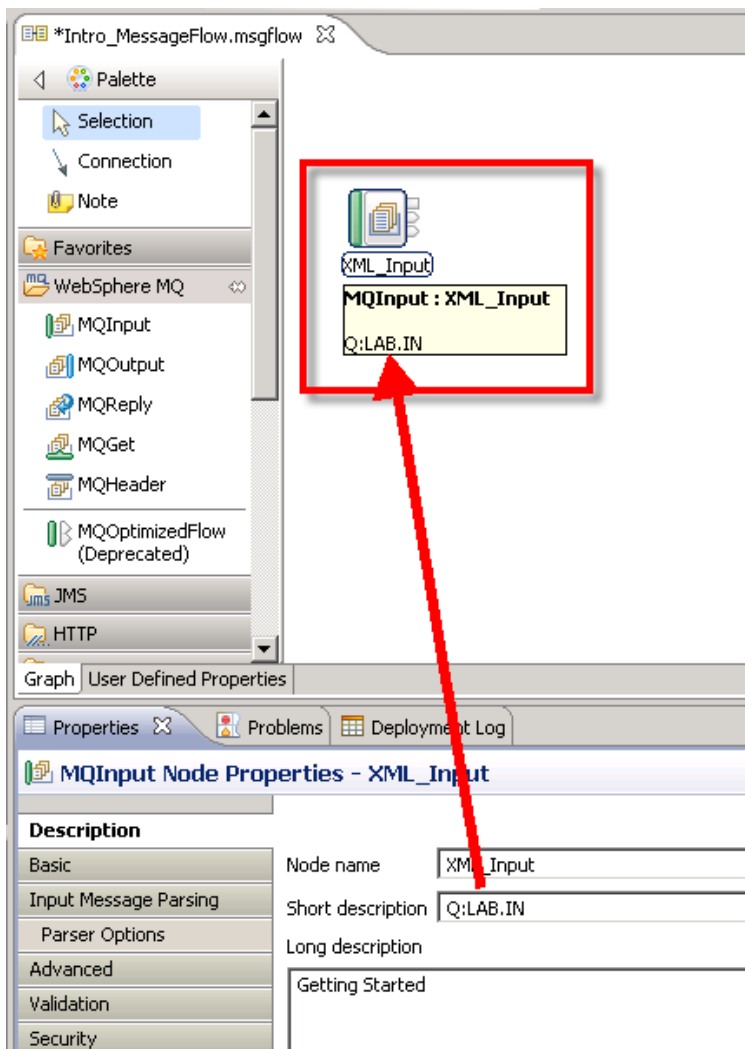
__15. Change the name of the node to **XML_Input**.

You must also define the Queue name in the node properties. The Basic tab should be selected. A Queue name is required and this is indicated by a message in red.

__16. Enter **LAB.IN** as the Queue name. Note that queue names are case sensitive. All queue names in the lab are upper case

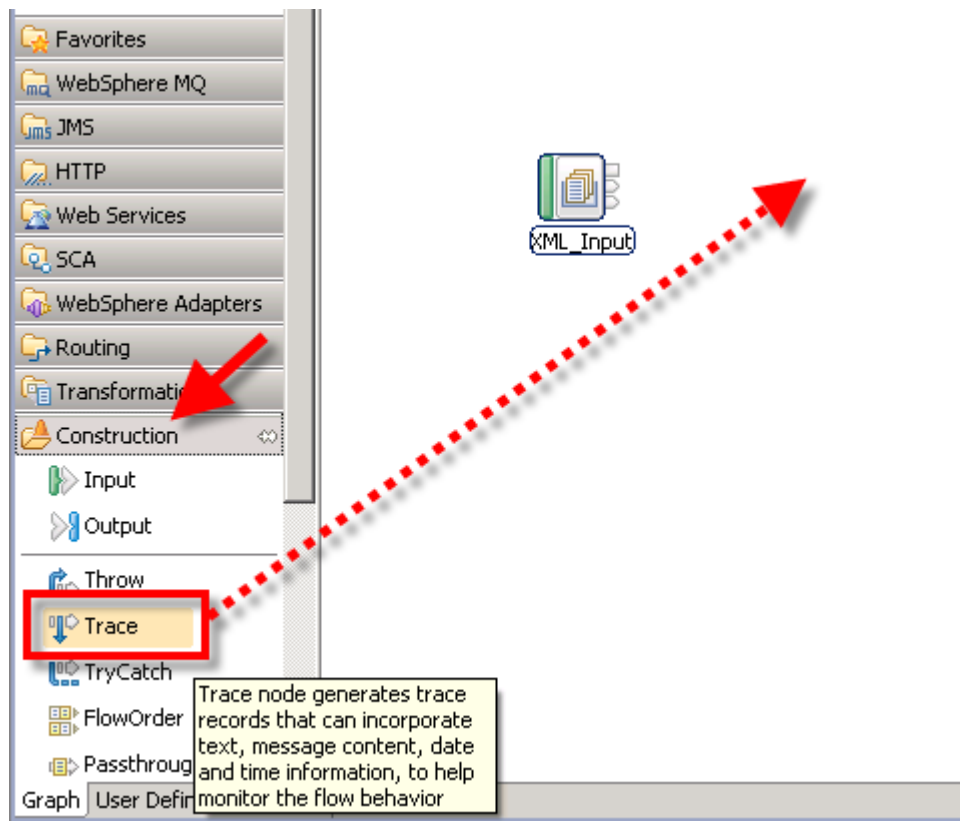


- __17. Select the **Description** tab. The purpose of this section is to encourage the developer to document the message flows and its nodes. In addition the name of the node may be changed. Note that XML_Input is the node name.
- __18. Enter **Q: LAB.IN** in the **Short description** field.
- __19. Enter your choice of text for the **Long description** (Getting Started is shown in the screen shot).



__20. Hover the mouse over the node name.

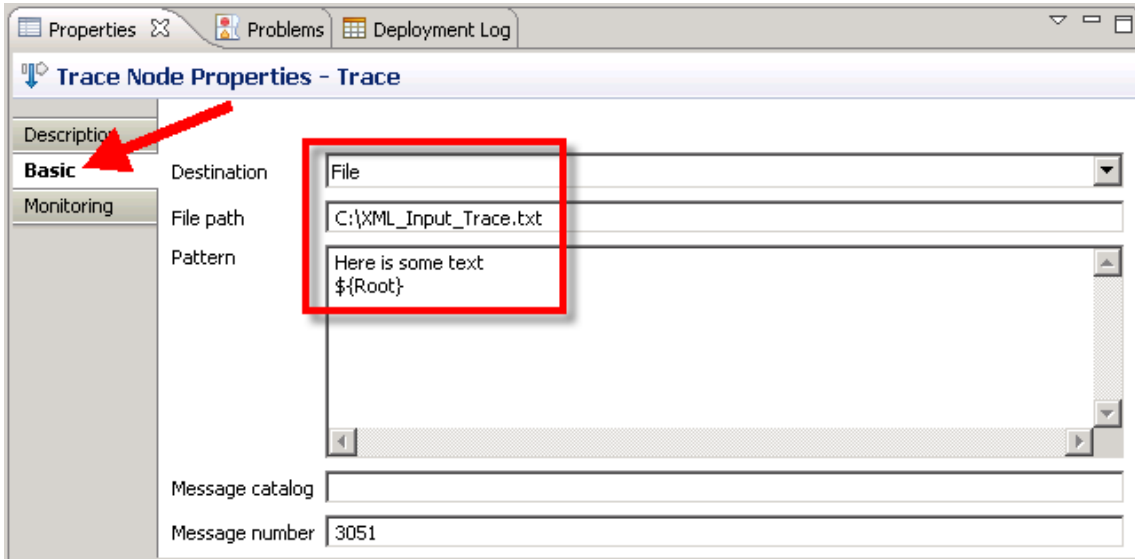
The information in the Short description field is displayed. When there are multiple nodes on the canvas, if you move from node to node with the mouse, the same tab in the Properties will be displayed.



The **Trace** node is in the **Construction** drawer.

__21. Open the **Construction** drawer by clicking it with the mouse.

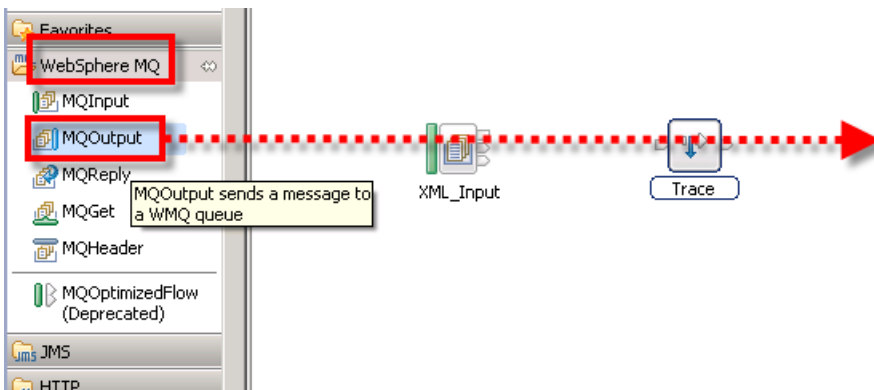
__22. Select the **Trace** node and place it on the canvas to the right of the XML_Input node. As shown in the example, when you place the cursor on a node name, a description is shown. You do not need to rename the Trace node.



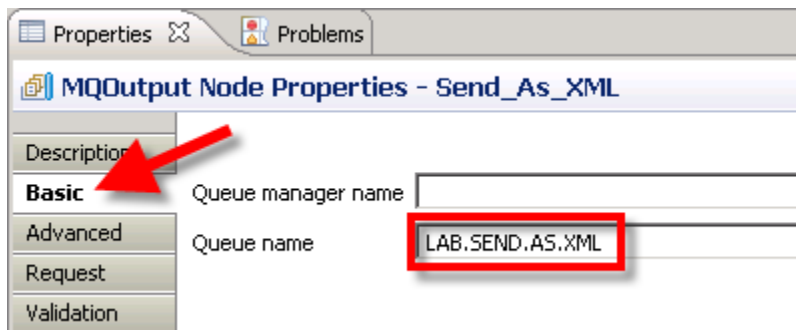
- __23. Use the pull down list on the Destination field and select **File**.
- __24. In the File Path field, enter **C:\XML_Input_Trace.txt**.

The information in the Pattern box tells the node what information to produce in the trace output. If you type a line of raw text it is echoed to the output. Enter a line of your choice – no quotes are needed.

- __25. In the Pattern box, enter the string **`\${Root}`** exactly as indicated – this tells the trace node to dump out the entire contents of the message that enters the node. Important – the pattern uses **curly braces**, not parenthesis.



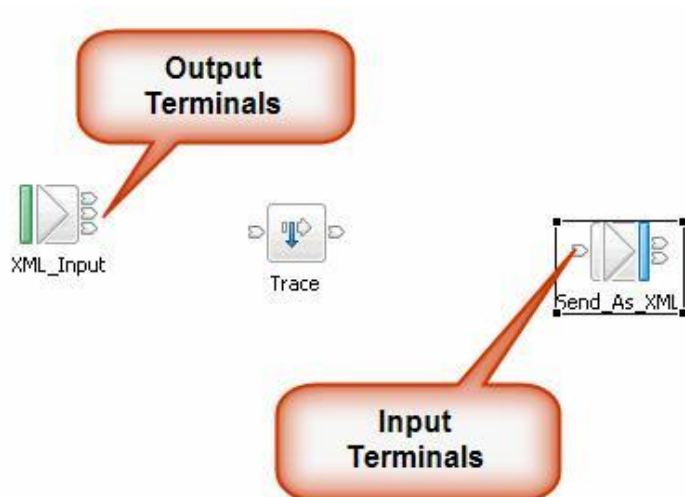
- __26. Open the **WebSphere MQ** drawer.
- __27. Select an **MQOutput** node from the drawer.
- __28. Place it to the right of the **Trace** node.
- __29. While the node name is highlighted, enter **Send_As_XML** as the new name.



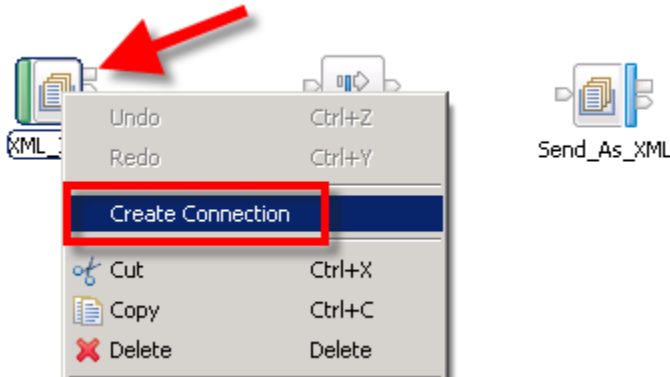
__30. If necessary, click the **Basic** tab.

__31. Set the **Queue name** to **LAB.SEND.AS.XML**. Queue names are case sensitive. It is a Best Practice to separate words in the queue name with a dot. Make sure you do not enter this information in the Queue manager name field!!

Background on Node Terminals:



As you work with the various nodes, you will also be working with their Input and Output terminals. Input terminals are typically named **In**. Most nodes have an Output terminal named **Out**. They may have several others. Some of these will have common names such as Failure or Catch and others will be unique to that particular node. Some nodes allow you to define the terminals. The terminals are given a name when they are defined. The lab instructions will identify the Output terminal to be used when connecting nodes together. If you hover the mouse pointer over a terminal, a small popup will appear that identifies the name of the terminal.



You will now wire the nodes together to create a logical path for the message to follow through the message flow. You want to wire the **Out** terminal of the XML_Input node to the **In** terminal of the Trace node. There are two techniques:

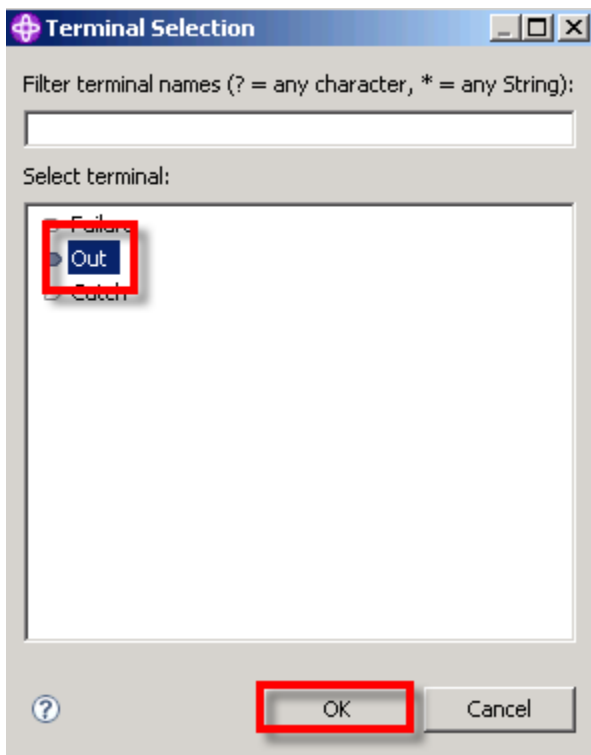
One – position the mouse over the Out terminal (in the middle), click and drag to the target and click again.

Two – right click on a node and select **Create Connection**. This is an example of a Terminal Selection presented as a result of the Create Connection.

The rest of the Lab instructions show the first method for wiring.

__32. Right-click on the XML_Input.

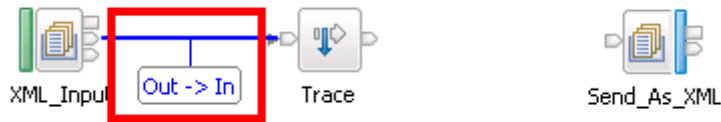
__33. Select **Create Connection**.



A list of the available Output terminals for this particular node type is shown

__34. Select **Out**.

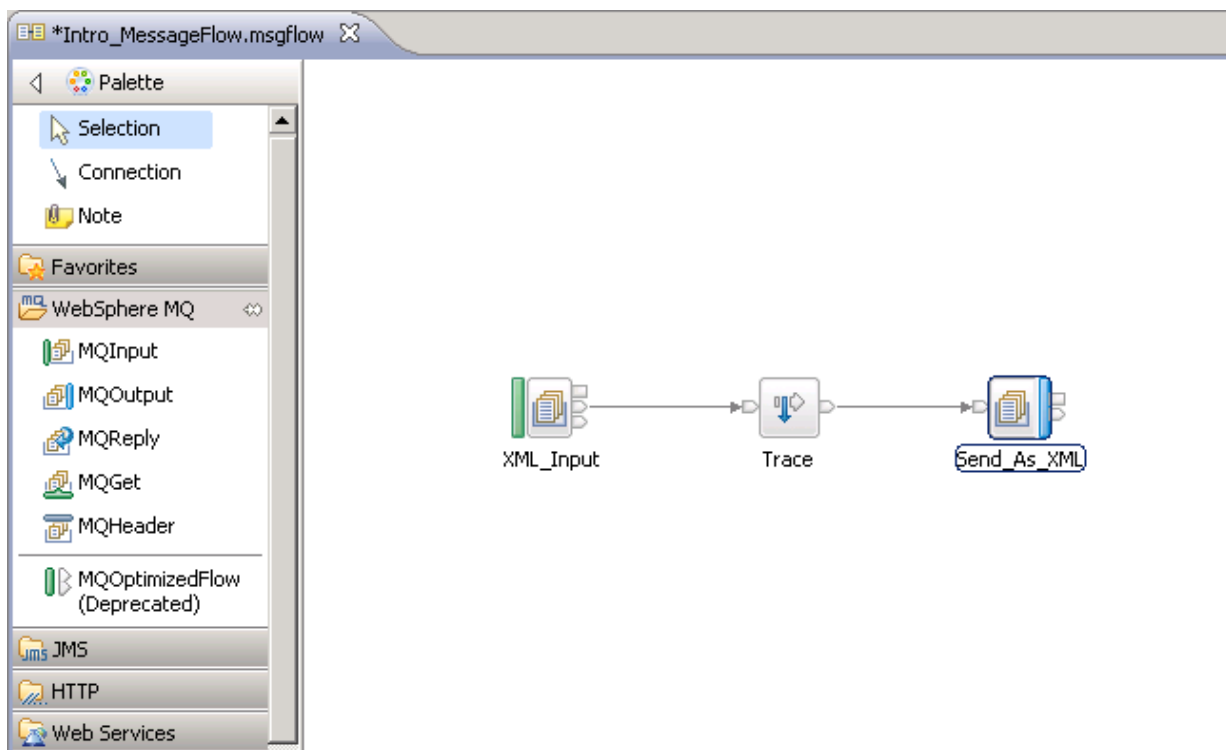
__35. Click **OK**.



You now have a rubber-banded connector.

__36. Position the connector on the In terminal of the Trace node.

__37. Click to anchor it giving a connection between the two nodes. If you put your mouse pointer on the connection it will pop up a summary of “from-to” information.



The same steps will be used to make a connection from the **Trace** node to the **Send_As_XML** node.

__38. Right click on the **Trace** node.

__39. Select **Create Connection**. This time you immediately get a rubber-banded connector. No selection list of terminals is presented because there is only an **Out** terminal on the Trace node.

__40. Position the mouse pointer on the **In** terminal of the **Send_As_XML** node.

__41. Click to anchor the connection.

Your message flow should now look like the above diagram.

__42. It is time to save your work – hold down the **Ctrl** key and press the **S** key to save the message flow. You can also click on the “diskette” icon to do a save or use **File → Save**.

The following graphic will be used as a reminder when it is time to save your work.



This is the end of lab 1.

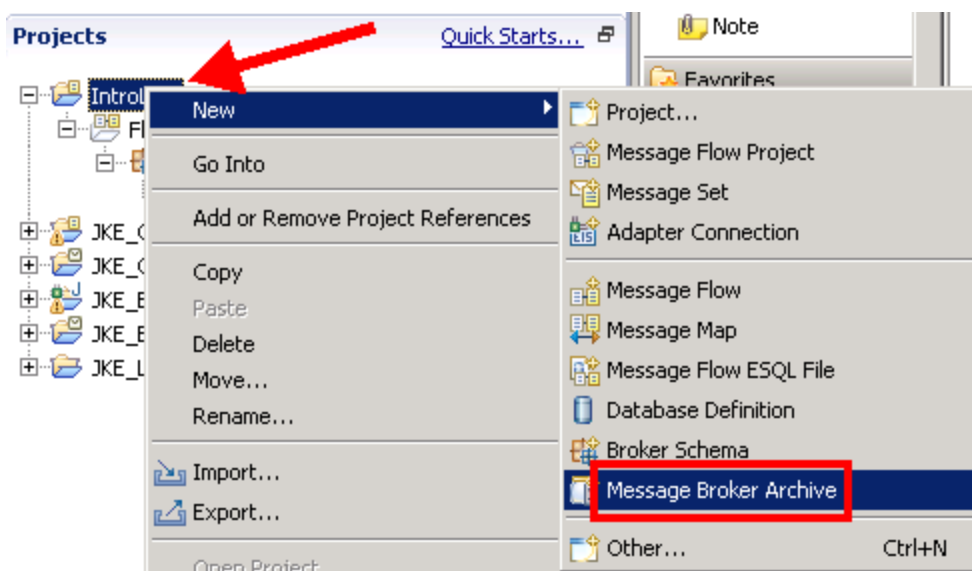
Lab 2 Deploying and Executing a Message Flow

2.1 Overview

The message flow that was built in the previous lab will now be tested. The flow will be deployed to an Execution Group in a Broker where it will execute. It will then be available to begin processing messages. There is no need to restart the Broker or the Execution Group for the deployment of a new or changed message flow.

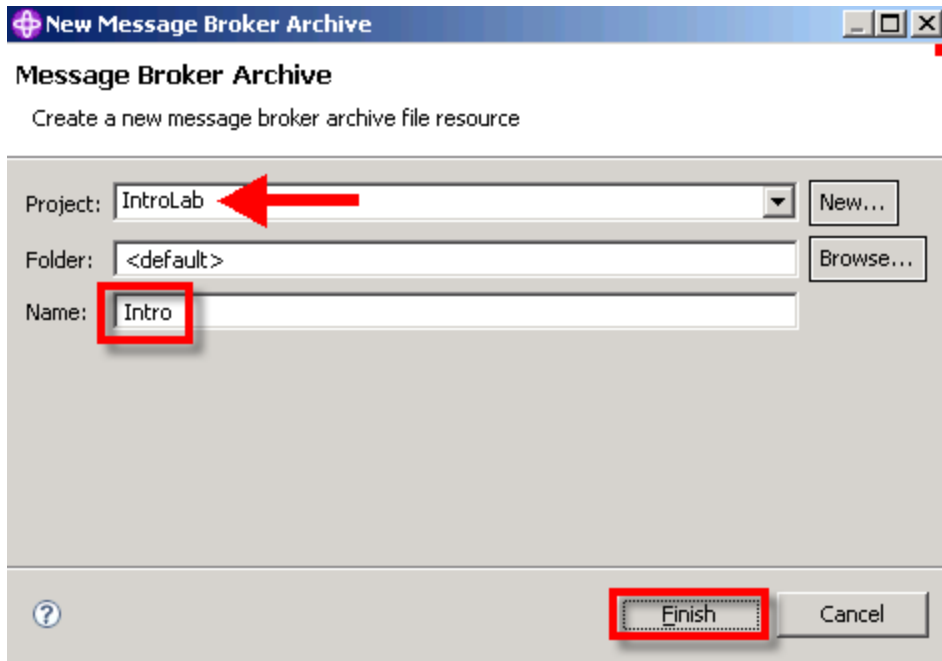
The unit of deployment is a Broker Archive file (BAR). Broker archive files have a file extension of “bar”.

The Broker Archive file will hold the artifacts to be deployed to a specific Execution Group in a specific Broker. It may contain message flows, message sets, XSL Transformations (XSLT) Style Sheets, Java™ Archive (JAR) and XSDZIP (schema) files. In addition, the related source files may also be included. When you add a message flow to the BAR file, additional validation of the message flow is performed. The BAR file is then deployed to the Broker. The final validation of the artifacts is done by the Broker. If errors are found by the Broker they will be reported back in the Event Log.

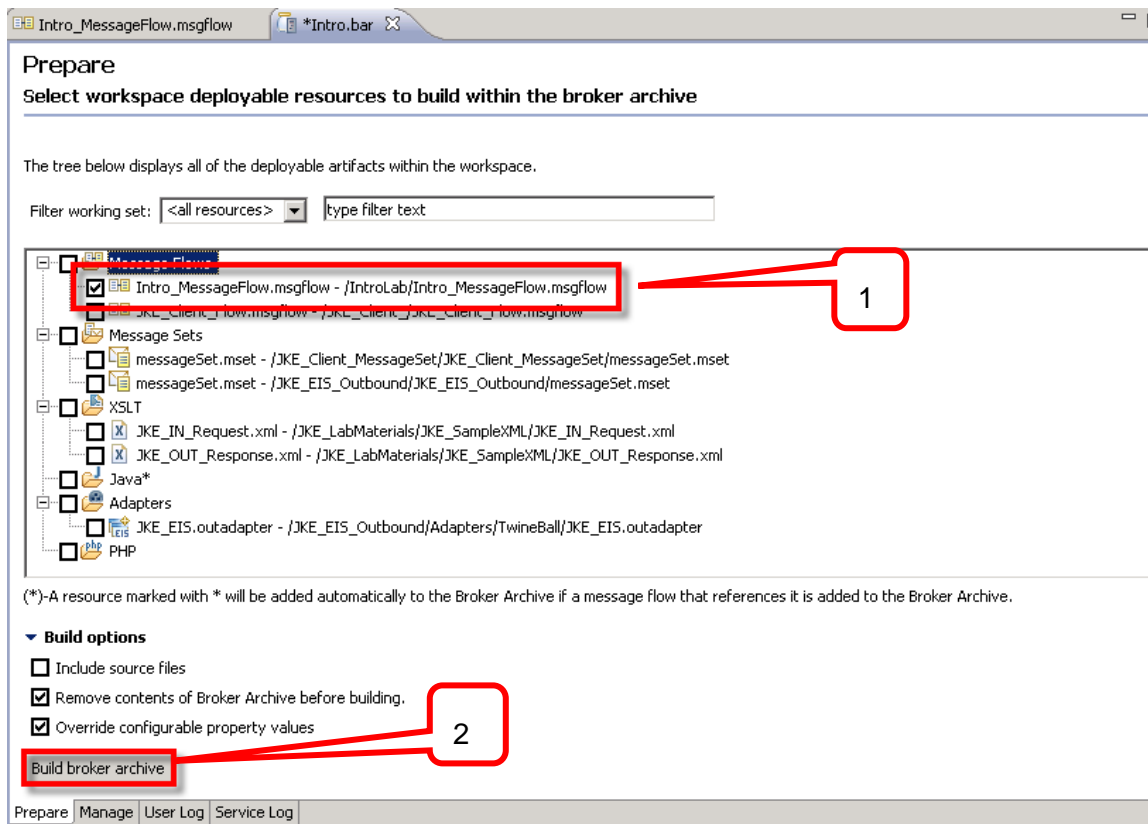


A Broker Archive file must be created to hold the resources to be deployed.

- ___1. Select the **IntroLab** project in the project navigator pane.
- ___2. Press the right mouse button.
- ___3. Select **New->Message Broker Archive** from the menu.

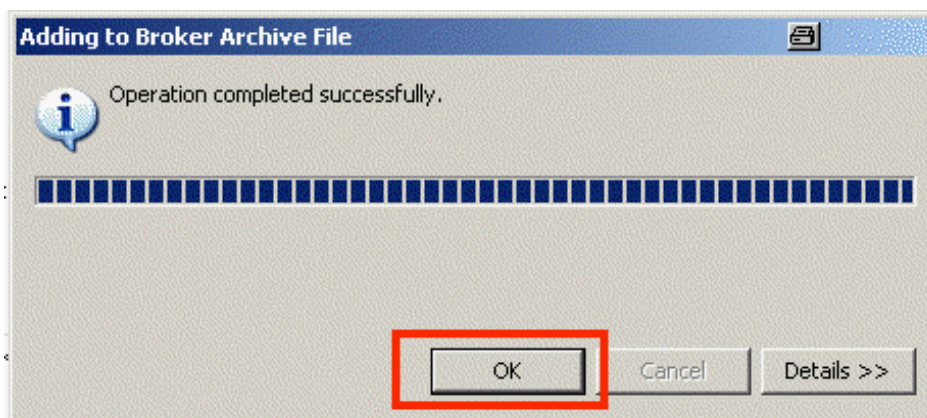


- __4. The **IntroLab** project should already be selected as the project to store the archive file in.
- __5. Enter **Intro** as the name for the bar file.
- __6. Press the **Finish** button.

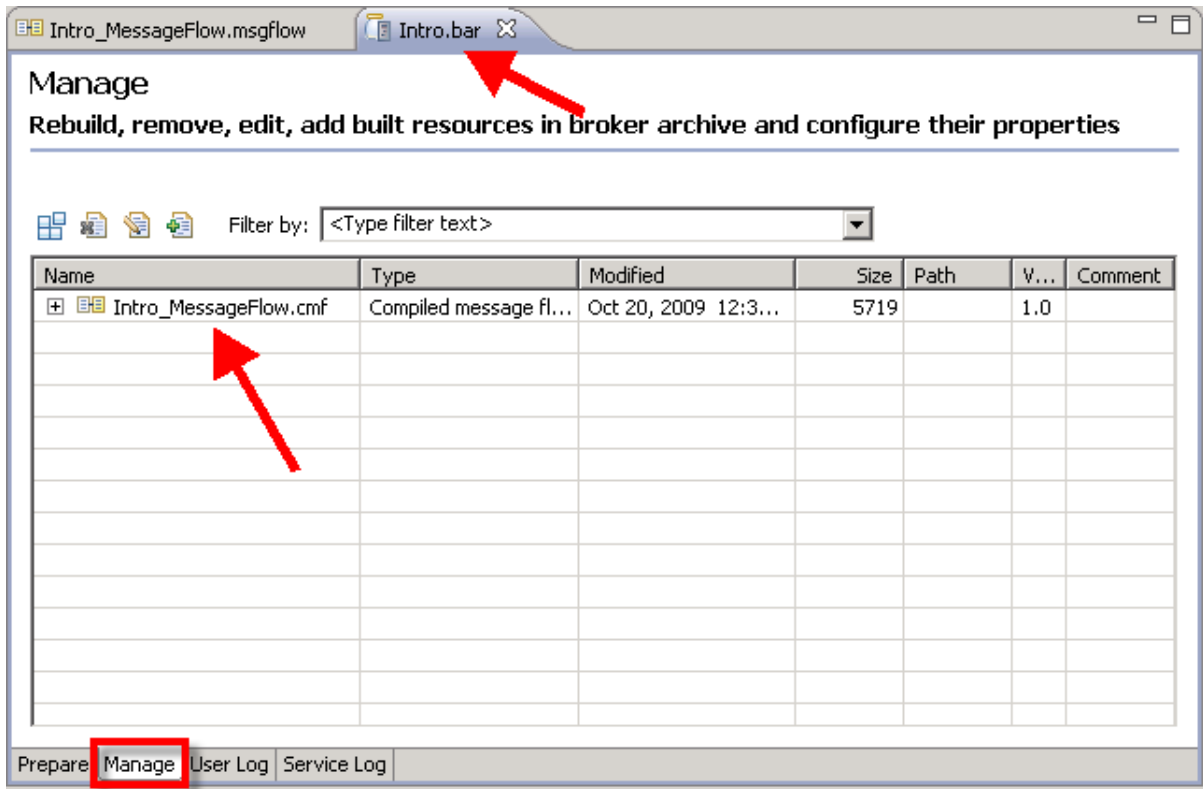


The BAR file editor will be launched. This is the **Prepare** tab. It lists the deployable assets that are available in your workspace. You are only interested in the **Intro_MessageFlow** message flow at this time.

- __7. Click the check box for the **Intro_MessageFlow**.
- __8. Click **Build broker archive**.




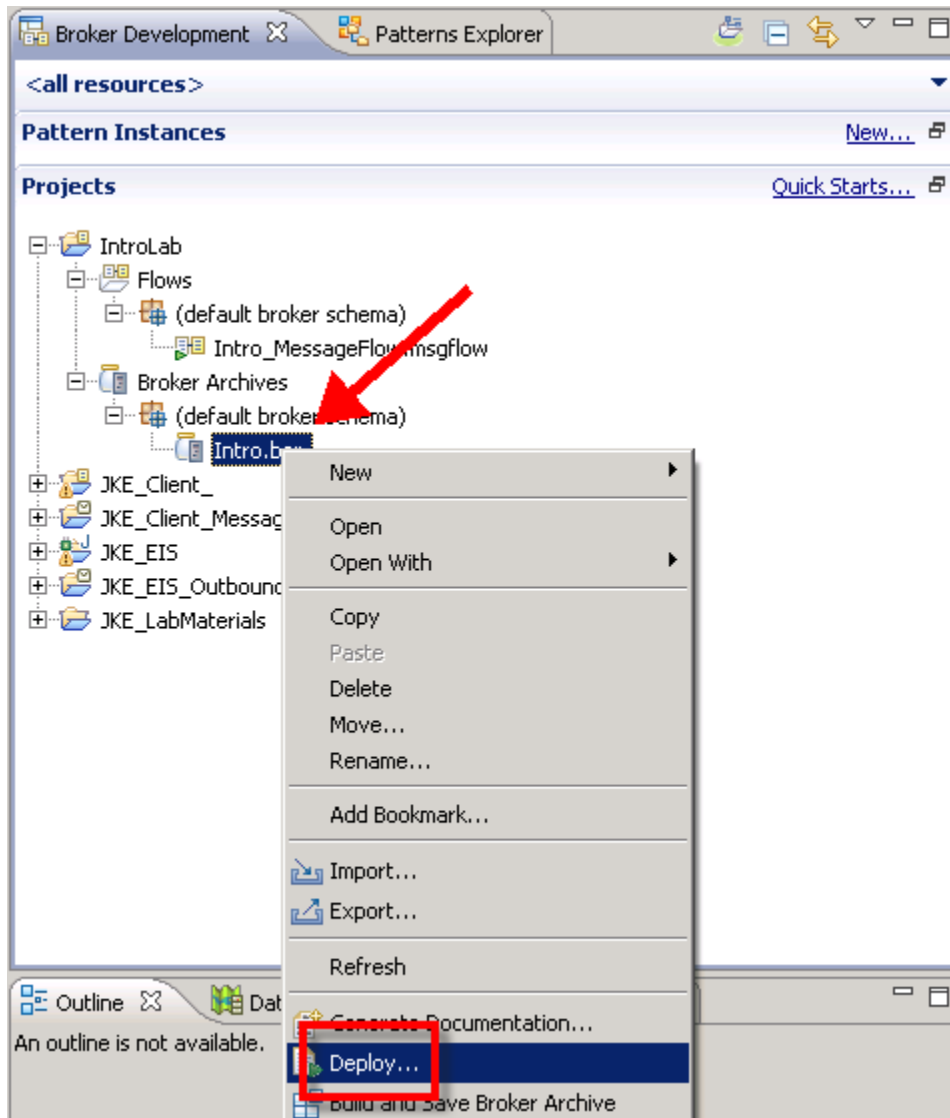
- ___9. You will receive a confirmation panel. Click **OK** to dismiss it. If any errors are found at this point, the status message will indicate that “some flows could not be added” and the Details tab will have specific information.



- ___10. Click the **Manage** tab at the bottom of the Intro.bar editor. This view will be shown.

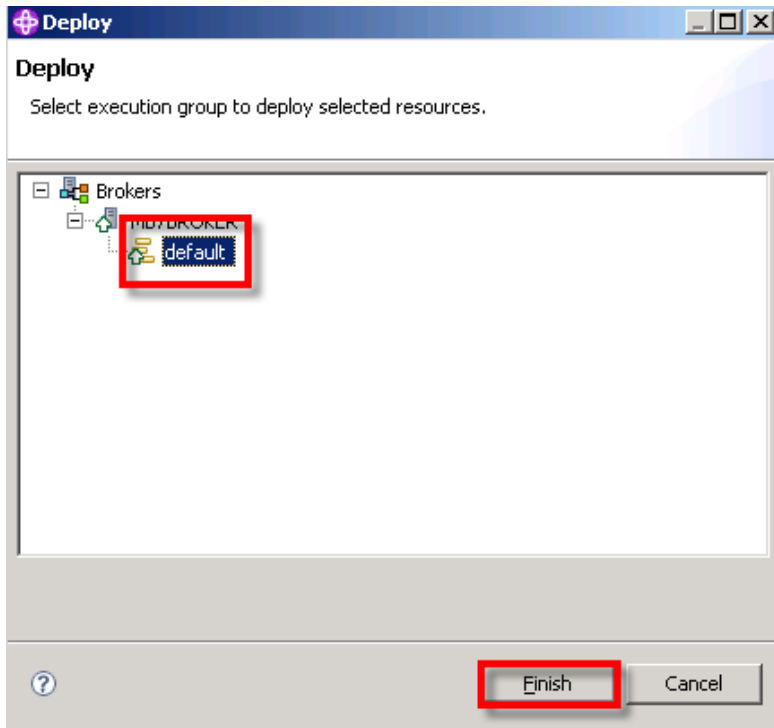
There is now content in the BAR file, a compiled message flow (file type .cmf). The time stamp that shows when this particular item was modified can be a handy piece of reference information. Make this field larger so that the entire value is visible.

- ___11.  Save the BAR file with a **Ctrl S**.

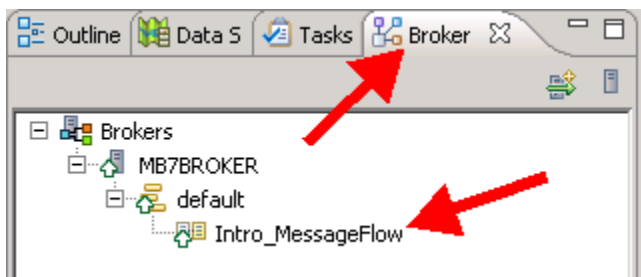


The next step is to “deploy” the Bar file using one of two methods. Both will be used in the exercises.

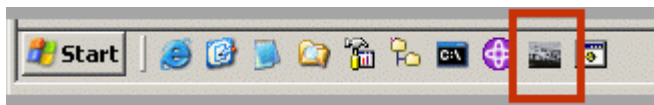
- __12. Expand **Broker Archives->(default broker schema)**.
- __13. Select the **Intro.bar** broker archive file
- __14. Press the right mouse button.
- __15. Select **Deploy** from the menu.



- __16. Select the **default** execution group.
- __17. Press the **Finish** button to initiate the deployment operation.



When the deployment operation finishes the message flow is now running. It is waiting for a message to appear in the **LAB.IN** queue. The message flow should be visible under the **default** execution group on the **Broker** tab in the lower left pane.

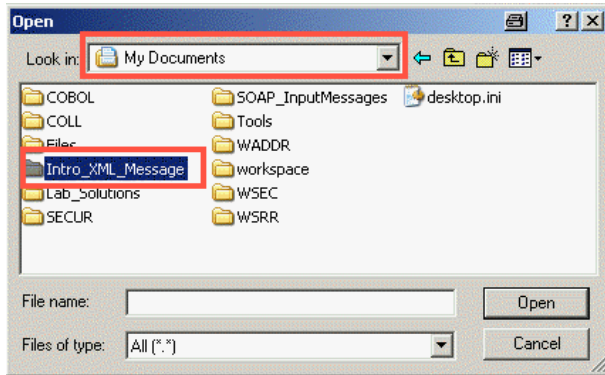


The message flow is now ready to be tested. You need a tool that will allow you to place a message on the input queue that your message flow has done a “GET with Wait” on. RFHUtil is a very convenient utility that is provided via a SupportPac (IH03). The RFHUtil tool will be used during the first four labs, so it is not necessary to close it. There are some exercises when you will have multiple copies activated. You will discover that when the tool has a queue open, the tab in the action bar will show the name of the open queues.

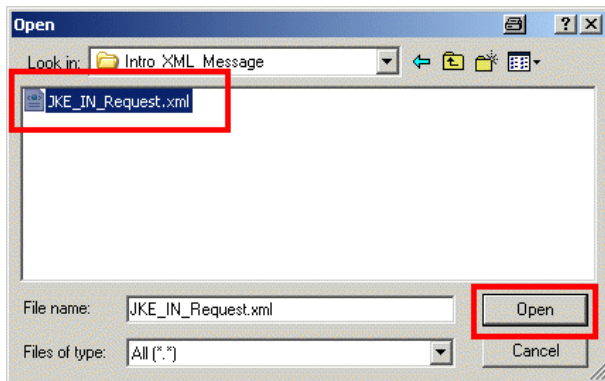
- __18. Launch RFHUtil by clicking on the indicated icon in the quick launch tray.

The screenshot shows the RfUtil V7.0.1 application window. The 'Queue Manager Name (to connect to)' dropdown is set to 'MB7QMGR' and the 'Queue Name' dropdown is set to 'LAB.IN'. Both dropdowns are highlighted with red boxes. The 'Open File' button is also highlighted with a red box. Other visible elements include the 'File Code Page' set to 437, the 'File Name' field, and various control buttons like 'Read Q', 'Write Q', 'Browse Q', 'Start Browse', 'Browse Next', 'Browse Prev', 'End Browse', 'Close Q', 'Save Q', 'Purge Q', 'Load Q', 'Display Q', 'Cluster Open', 'User Props', and 'Put/Get Options'.

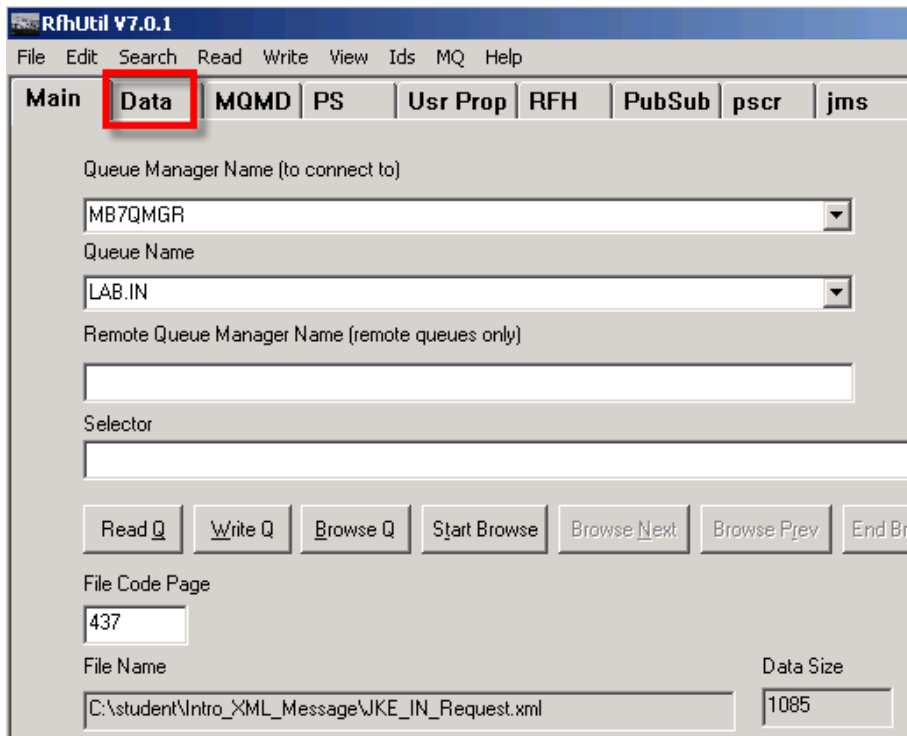
- __19. You must specify a Queue Manger and Queue name....**use the provided pull-downs** to select the **MB7QMGR** queue manager and the **LAB.IN** queue name.
- __20. Click **Open File**.



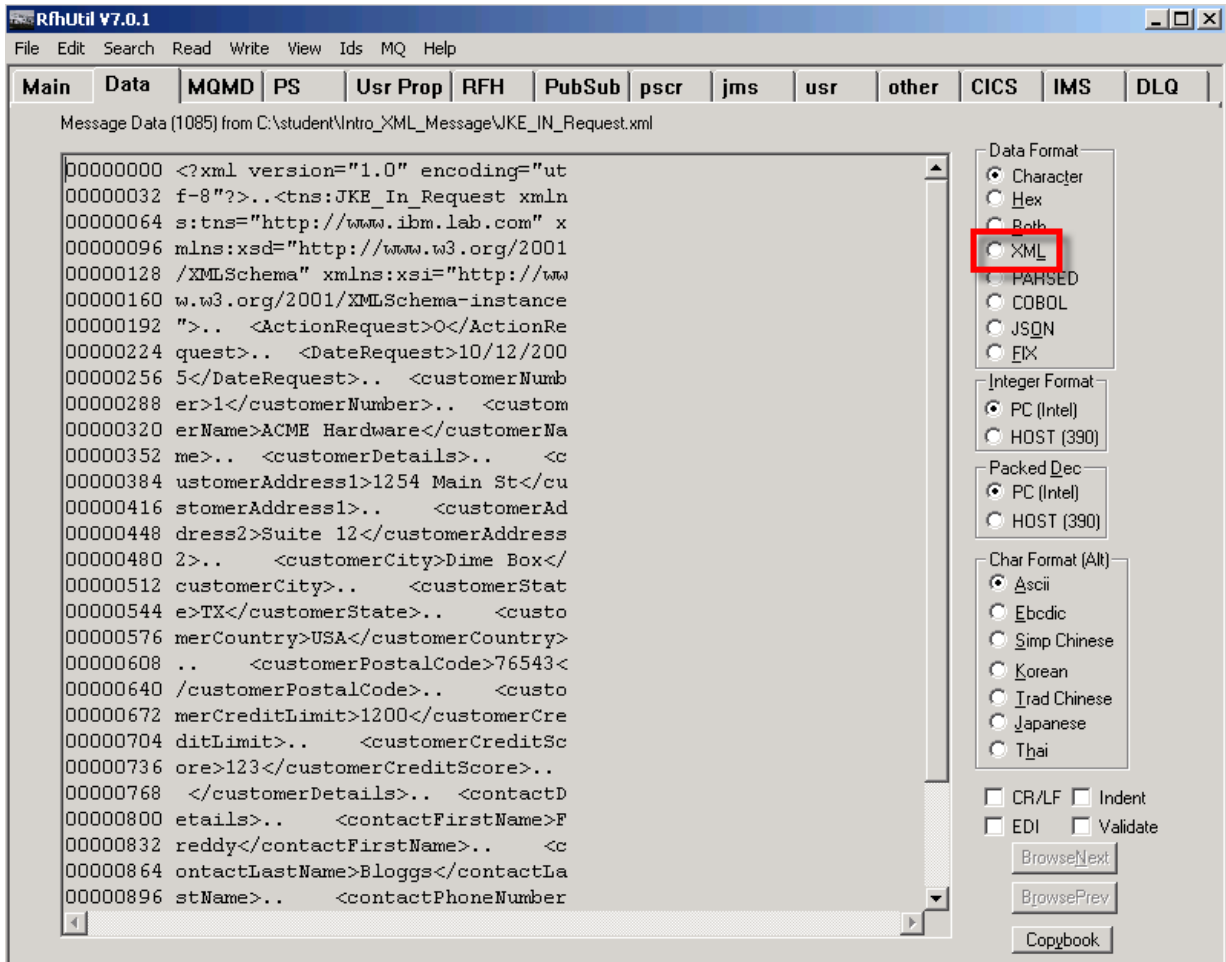
- __21. Navigate to the **C:\Student\Intro_XML_Message** directory. Note that in this Image My Document has been linked to C:\Student.



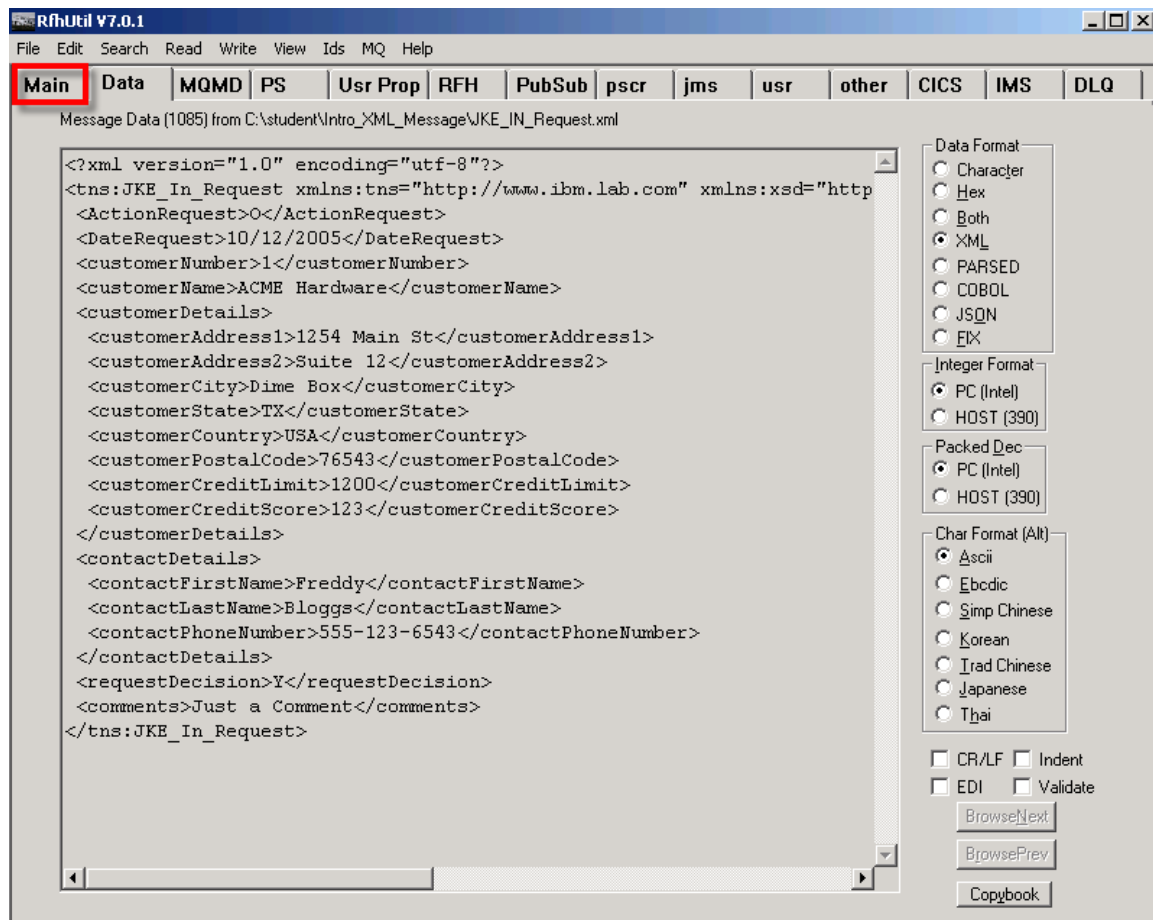
- __22. Select **JKE_IN_Request.xml**.
- __23. Click **Open**.



__24. Click on the **Data** tab to see the information that will be used as the payload of the message....

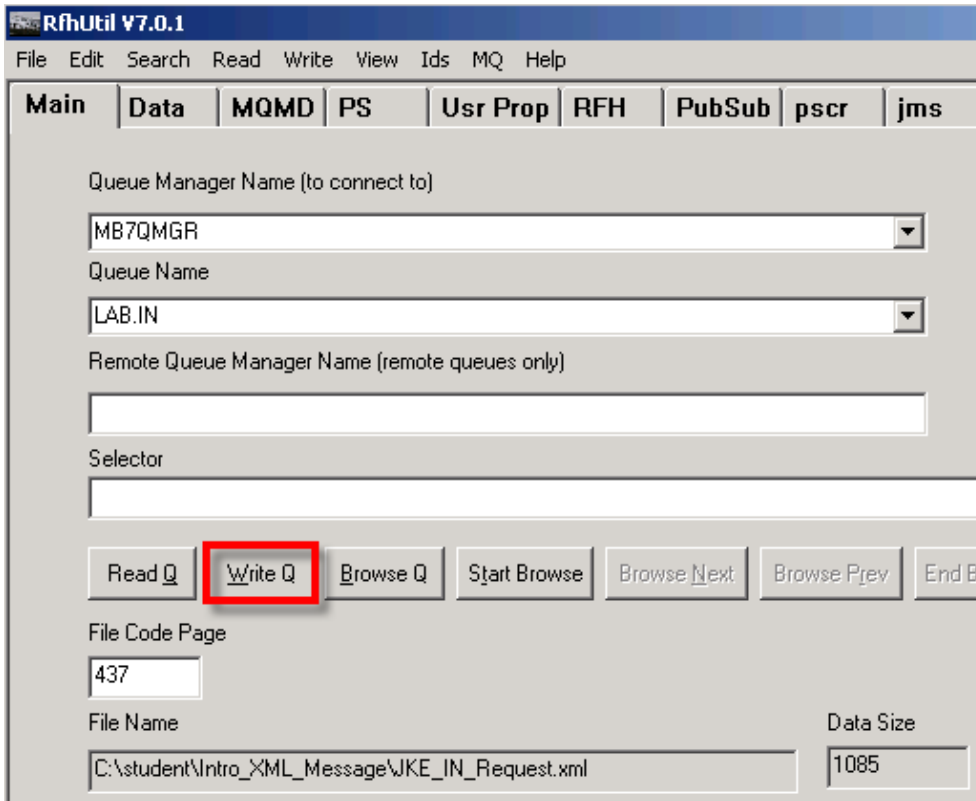


__25. Since this is XML data, click on the **XML** radio button on the right to see a formatted display.

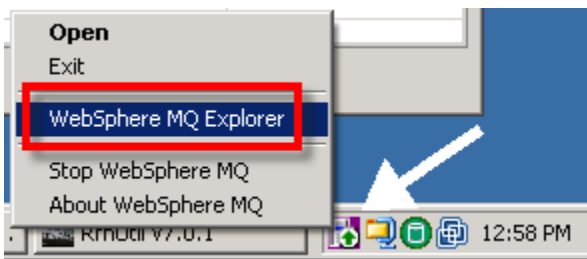


The XML structure is now formatted and much easier for a human to read.

__26. Click on the **Main** tab.



- __27. Click on the **Write Q** button and your message will be placed on the LAB.IN queue. You now need to check the output queue to see if there is a message from the flow.



WebSphere MQ Explorer will now be started. The icon in the system tray can be used.

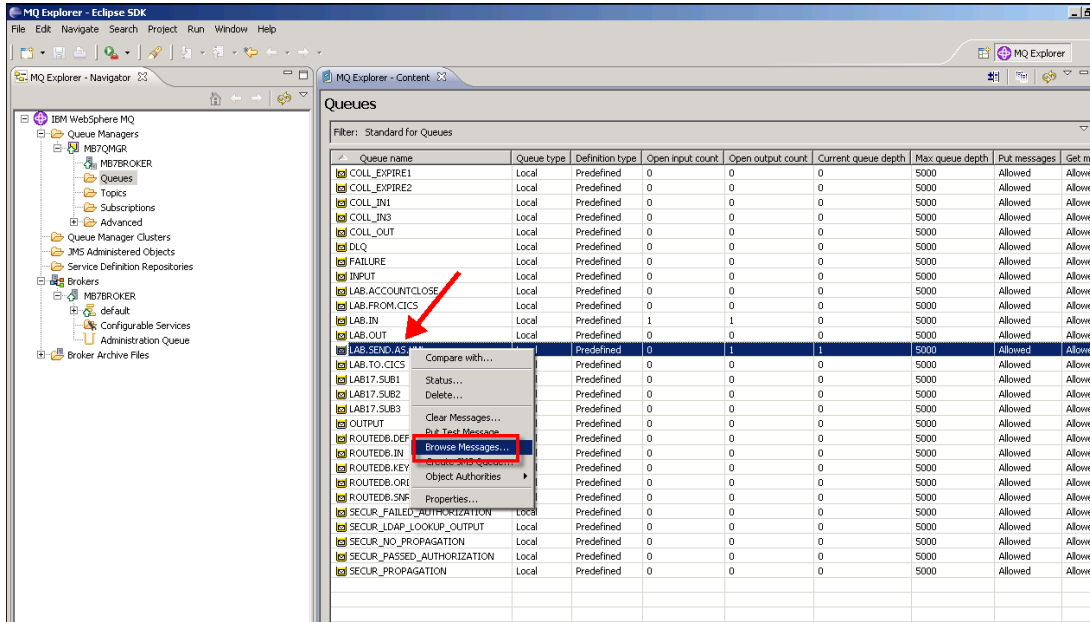
- __28. Select the WebSphere MQ icon in the Windows tray as shown above.
- __29. Press the right mouse button.
- __30. Select **WebSphere MQ Explorer** from the menu.

The screenshot shows the MQ Explorer interface. On the left, the Navigator pane shows the hierarchy: IBM WebSphere MQ > Queue Managers > MB7QMGR > MB7BROKER > Queues. The main pane displays a table of queues with the following columns: Queue name, Queue type, Definition type, Open input count, Open output count, Current queue depth, Max queue depth, Put messages, and Get messages. The 'LAB.SEND.AS.XML' queue is highlighted with a red arrow and a red box around its 'Current queue depth' value of 1.

Queue name	Queue type	Definition type	Open input count	Open output count	Current queue depth	Max queue depth	Put messages	Get messages
COLL_EXPIRE1	Local	Predefined	0	0	0	5000	Allowed	Allowed
COLL_EXPIRE2	Local	Predefined	0	0	0	5000	Allowed	Allowed
COLL_IN1	Local	Predefined	0	0	0	5000	Allowed	Allowed
COLL_IN3	Local	Predefined	0	0	0	5000	Allowed	Allowed
COLL_OUT	Local	Predefined	0	0	0	5000	Allowed	Allowed
DLQ	Local	Predefined	0	0	0	5000	Allowed	Allowed
FAILURE	Local	Predefined	0	0	0	5000	Allowed	Allowed
INPUT	Local	Predefined	0	0	0	5000	Allowed	Allowed
LAB.ACCOUNTCLOSE	Local	Predefined	0	0	0	5000	Allowed	Allowed
LAB.FROM.CICS	Local	Predefined	0	0	0	5000	Allowed	Allowed
LAB.IN	Local	Predefined	1	1	0	5000	Allowed	Allowed
LAB.OUT	Local	Predefined	0	0	0	5000	Allowed	Allowed
LAB.SEND.AS.XML	Local	Predefined	0	1	1	5000	Allowed	Allowed
LAB.TO.CICS	Local	Predefined	0	0	0	5000	Allowed	Allowed
LAB17.SUB1	Local	Predefined	0	0	0	5000	Allowed	Allowed
LAB17.SUB2	Local	Predefined	0	0	0	5000	Allowed	Allowed
LAB17.SUB3	Local	Predefined	0	0	0	5000	Allowed	Allowed
OUTPUT	Local	Predefined	0	0	0	5000	Allowed	Allowed
ROUTEDB.DEFAULT	Local	Predefined	0	0	0	5000	Allowed	Allowed
ROUTEDB.IN	Local	Predefined	0	0	0	5000	Allowed	Allowed
ROUTEDB.KEYNOTFOUND	Local	Predefined	0	0	0	5000	Allowed	Allowed
ROUTEDB.ORDERLOCK	Local	Predefined	0	0	0	5000	Allowed	Allowed
ROUTEDB.SNRSTAFF	Local	Predefined	0	0	0	5000	Allowed	Allowed
SECUR_FAILED_AUTHORIZATION	Local	Predefined	0	0	0	5000	Allowed	Allowed
SECUR_LDAP_LOOKUP_OUTPUT	Local	Predefined	0	0	0	5000	Allowed	Allowed
SECUR_NO_PROPAGATION	Local	Predefined	0	0	0	5000	Allowed	Allowed
SECUR_PASSED_AUTHORIZATION	Local	Predefined	0	0	0	5000	Allowed	Allowed
SECUR_PROPAGATION	Local	Predefined	0	0	0	5000	Allowed	Allowed

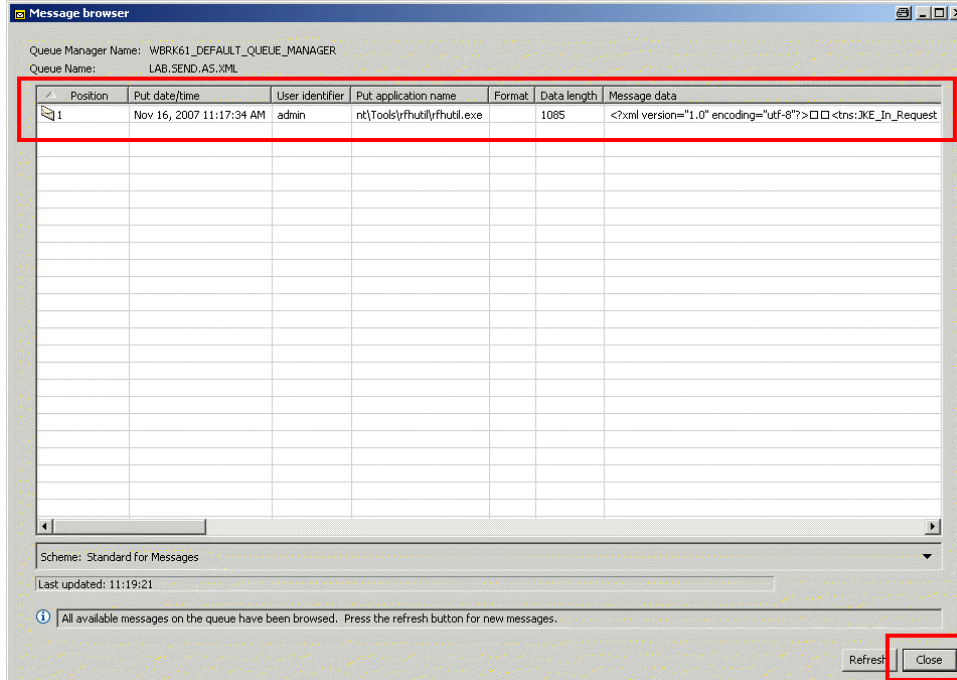
- __31. Expand the **MB7QMGR** folder in the navigator pane.
- __32. Click on **Queues**. A display is presented that shows various metrics for the queues including **Current queue depth** which is what you are interested in. The **LAB.SEND.AS.XML** queue contains a message....good news!

The order of information can be changed by altering the Scheme used to define the order in which properties are displayed.



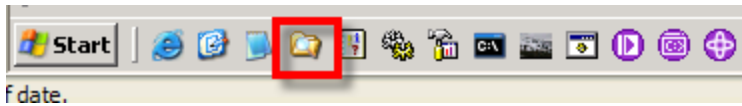
But, to be sure it is a recent message you should examine more details about it.

- __33. Right click on the **LAB.SEND.AS.XML** queue.
- __34. Select **Browse Messages** from the pull-down list.

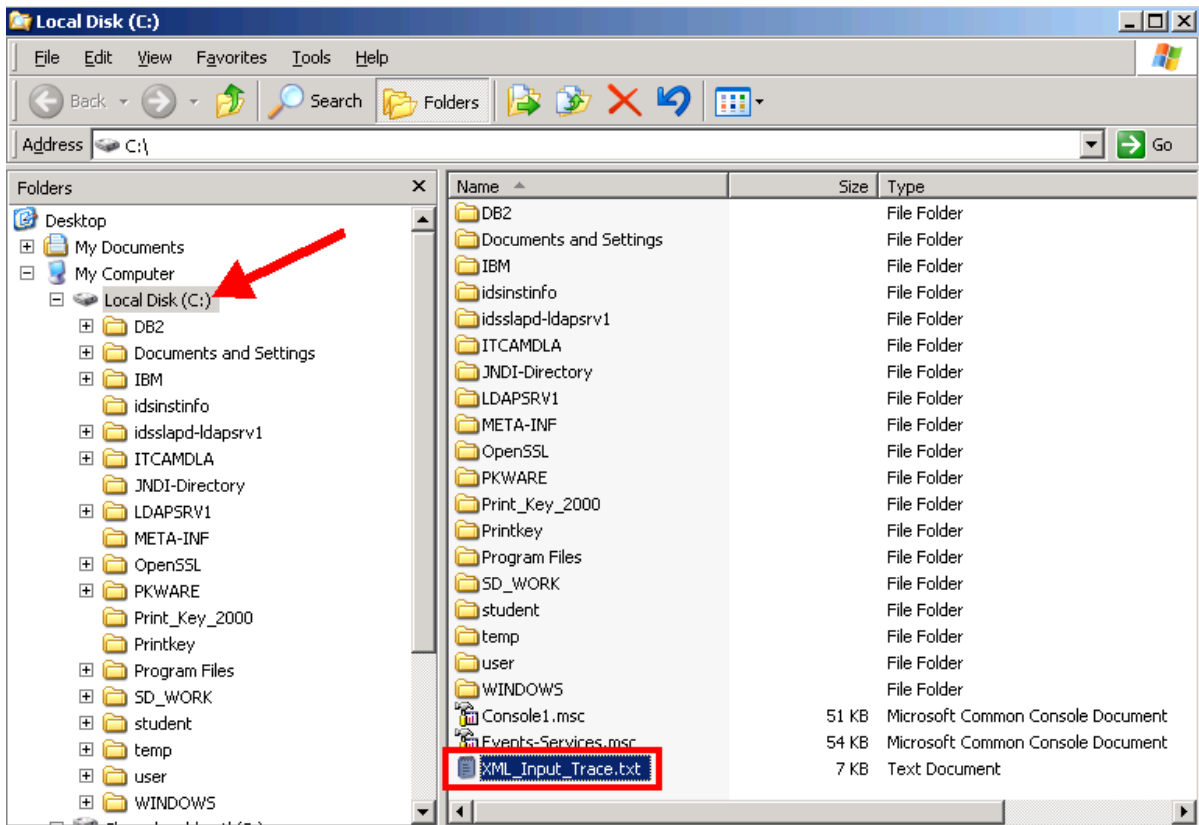


Here are some details about the message including its PUT date and time. Using this information you can be assured that this message came from your flow and it is working as you intended.

- __35. **Close** this window.



- __36. Bring up Windows Explorer. An icon has been provided. You need to examine the output from the Trace Node.



- __37. Double click on **XML_Input_Trace.txt**

Lab 3 Working with XML Messages

3.1 Overview

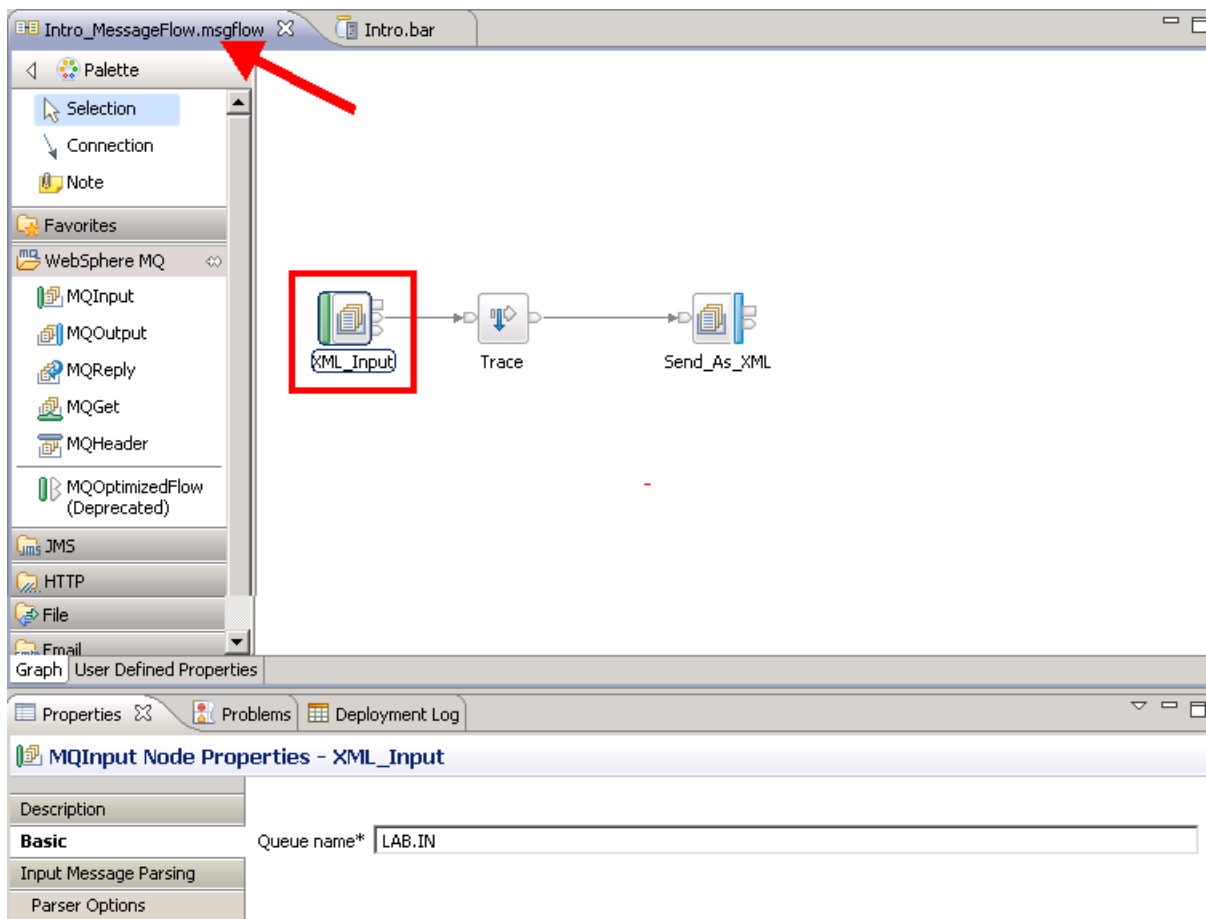
In this lab you will simply modify the Intro_MessageFlow to identify the parser (XMLNSC) to be used to process the message.

The steps are very simple.

You will modify the properties of the Input node and then update the archive file and deploy it.

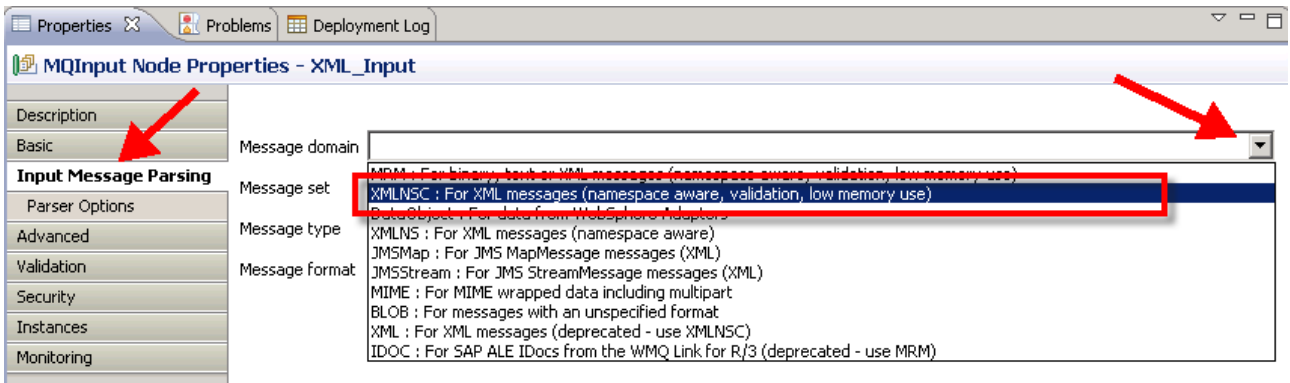
You will reuse your RfhUtil session and run another test.


You will view the trace file and see the difference.

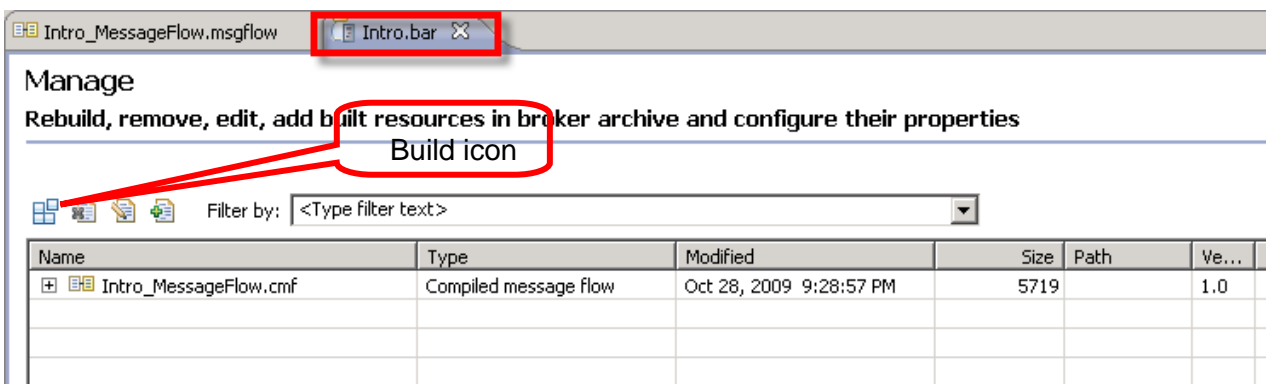


You need to modify the message flow so that it uses the XMLNSC parser to process the input message.

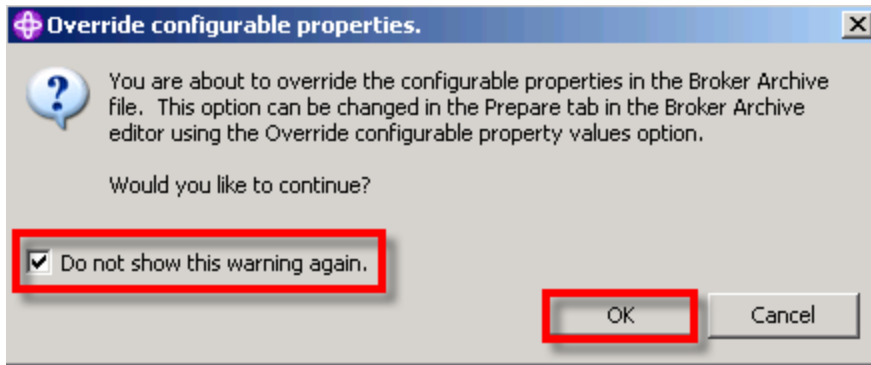
- __1. Return to the WebSphere Message Broker toolkit.
- __2. Click on the **Intro_MessageFlow** tab to bring the message flow into view.
- __3. Click on the **XML_Input** node to bring its properties into view.



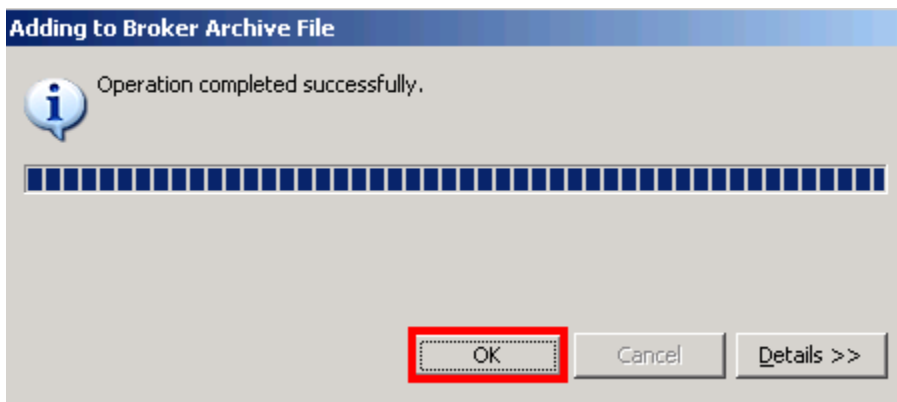
- __4. Click the **Input Message Parsing** tab of the Properties view. Since nothing was specified when the node was added, the Message domain (i.e. the parser) defaults to BLOB – which you saw in the trace. Without any information about the message format that is the only approach the broker can take.
- __5. Click the pull-down for the **Message domain**. The various parsers are listed along with a short description. Depending on the Message domain selection, the other fields may be enabled or disabled.
- __6. For this exercise, you will be using the **XMLNSC** parser. This tells the broker what parser to use for interpreting the message. In this case you are using an XML parser that supports Namespaces (the NS part) and builds a Compact tree (the C part).
- __7.  Save the message flow <Ctrl S> .




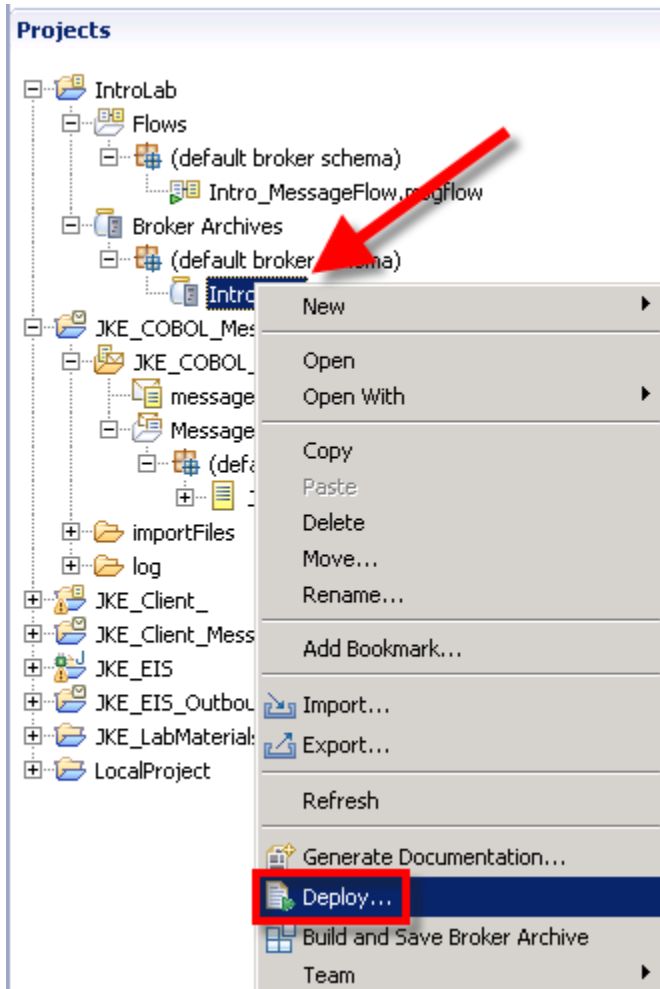
- __8. Click on the **Intro.bar** tab to bring the editor into view.
- __9. If necessary select the **Manage** tab.
- __10. Click on the **Build** icon to rebuild the archive file.



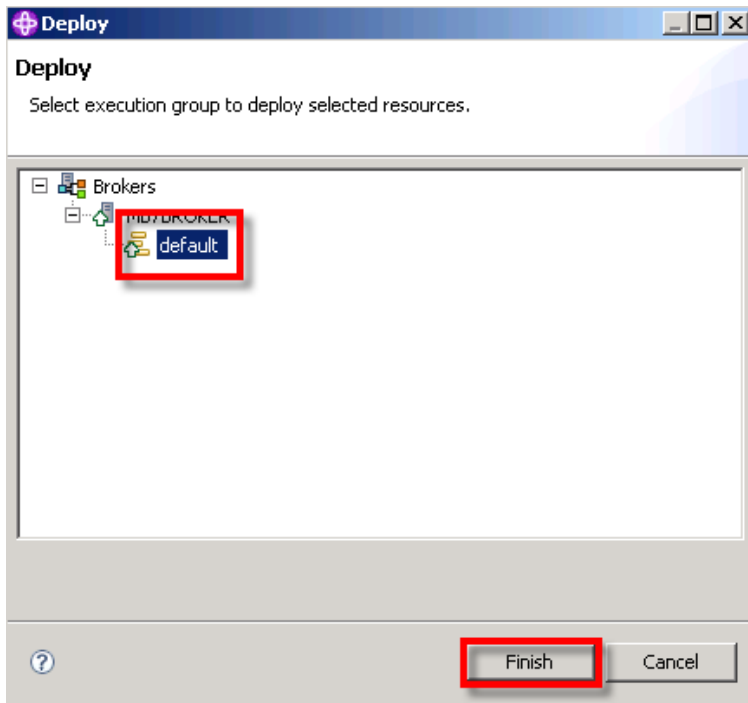
- __11. Click the **check box** to turn off this warning and click **OK** to dismiss this panel.



- __12. Click **OK** to dismiss the confirmation panel for the Broker archive file build.
- __13. When the rebuild is completed,  save the BAR file **<Ctrl-S>**.

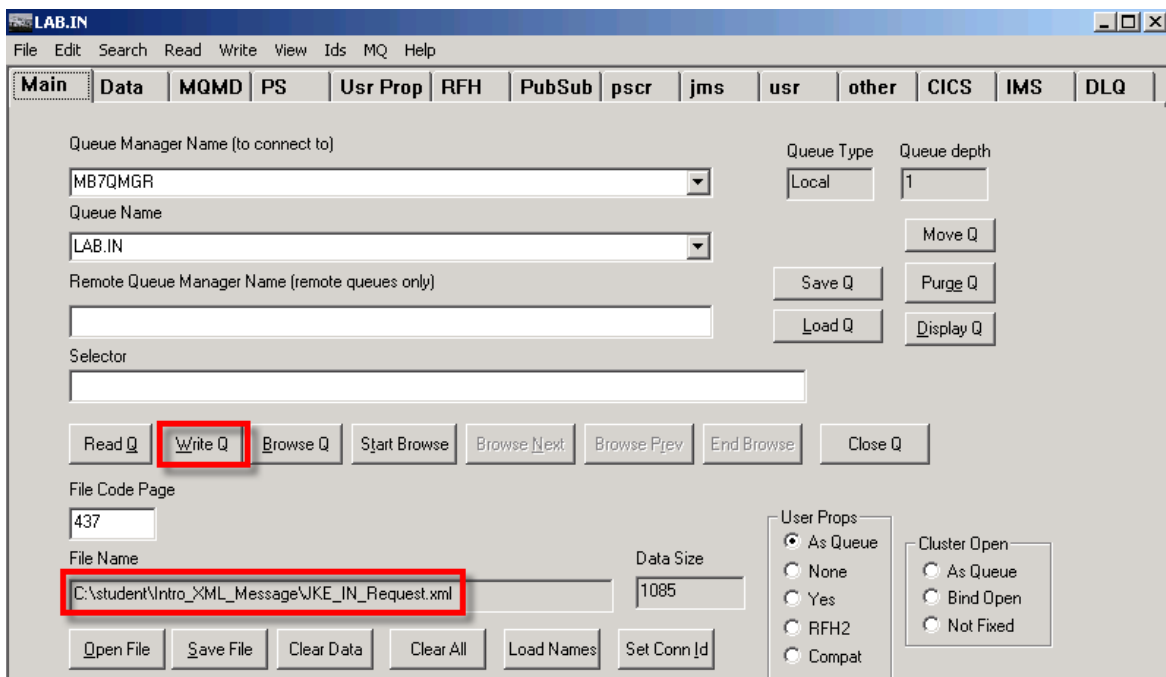


- __14. Select the **Intro.bar** message broker archive file in the navigator pane.
- __15. Press the right mouse button.
- __16. Select **Deploy** from the menu.



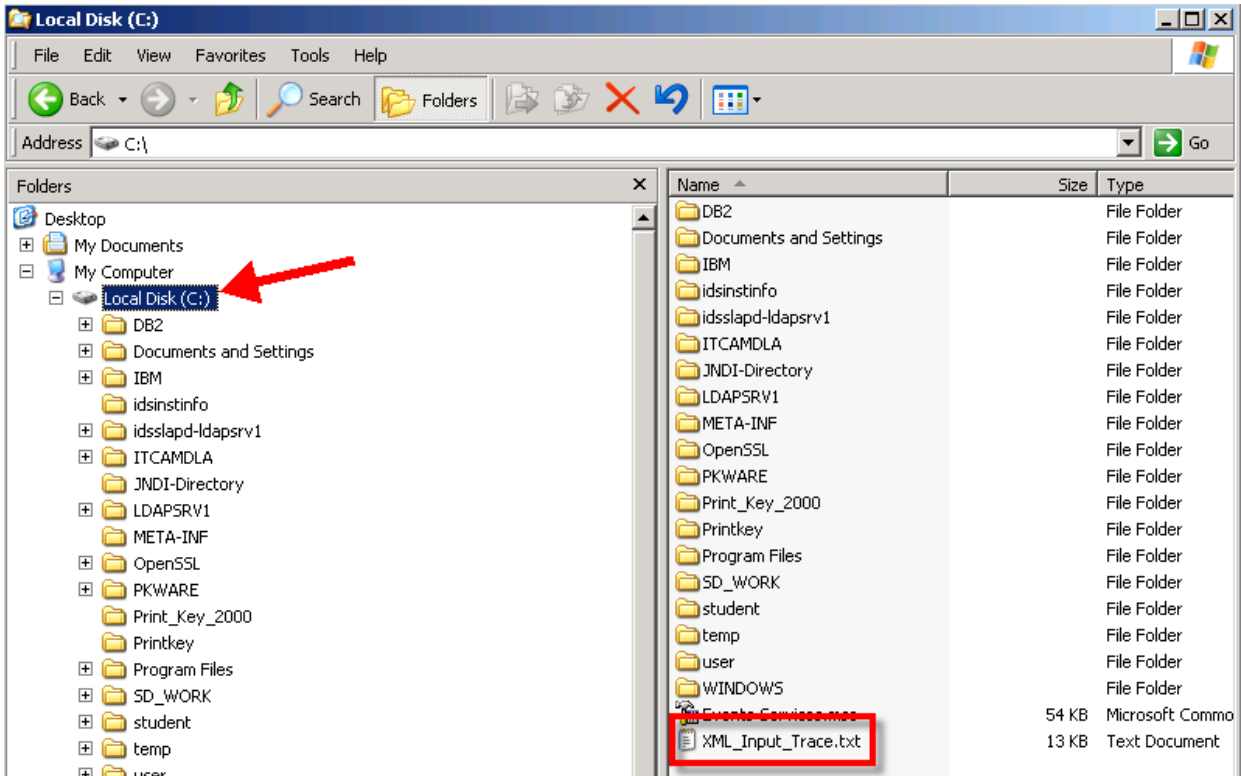
__17. Select the **default** Execution Group.

__18. Click **Finish**.

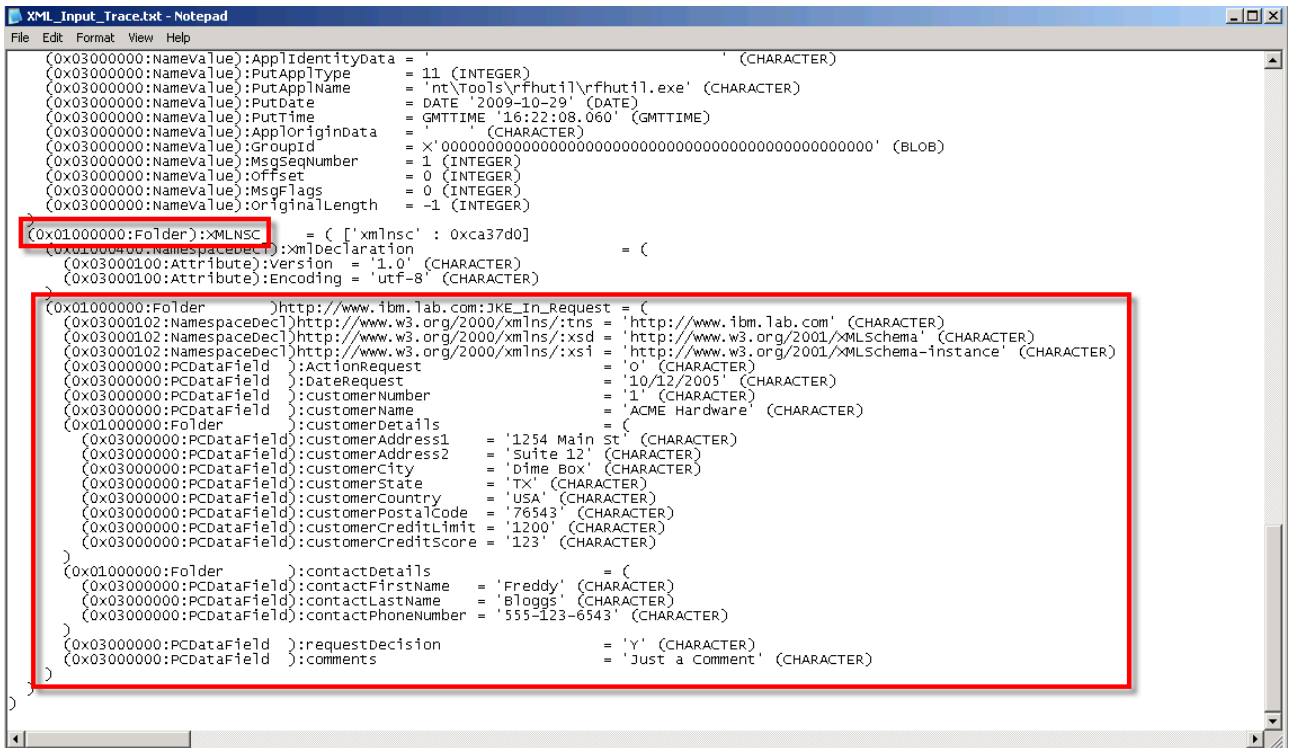


__19. Bring RfhUtil back into view by clicking on the **LAB.IN** tab in the Windows Task Bar at the bottom of the screen.

__20. Click on the **Write Q** button to send another test message into the flow



- __21. Bring up the Windows Explorer.
- __22. Double click on **XML_Input_trace.txt** file.



Trace output is placed at the end of any existing content in a file so scroll down to the bottom of the file and view the results. Much more pleasing...here is a nicely formatted message tree that will allow you to conveniently access the fields in the XML message by name. Notice the **XMLNSC** Domain name.

- __23. Scroll to the end of the file (**Ctrl + End** works).
- __24. Close the Notepad window.
- __25. Minimize the Windows Explorer and RFHUtil windows.

This is the end of lab 3.

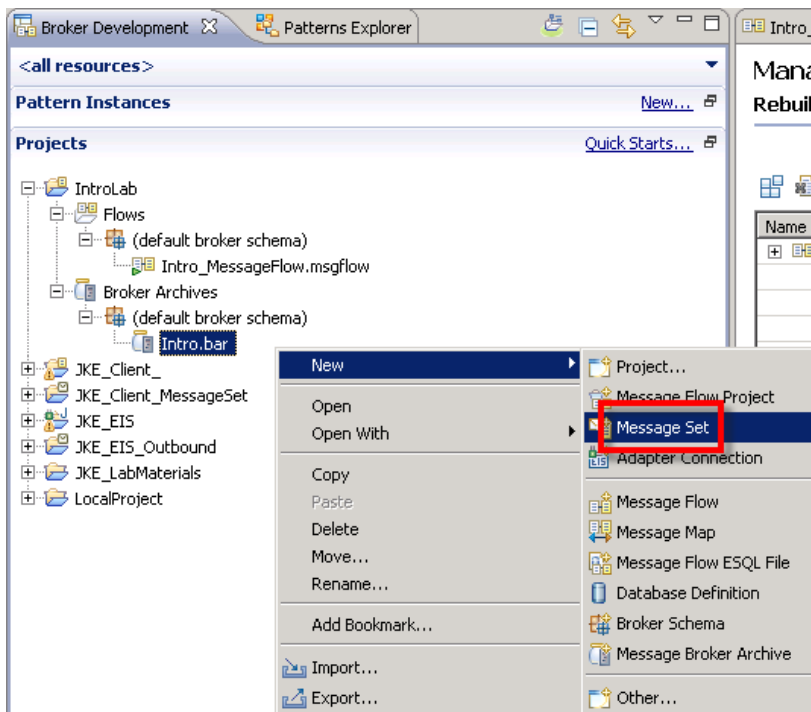
Lab 4 Working with Fixed Format (COBOL) Messages

4.1 Overview

In this lab you will first build a Message Set (JKE_COBOL_MessageSet) and then populate it by importing a COBOL copybook.

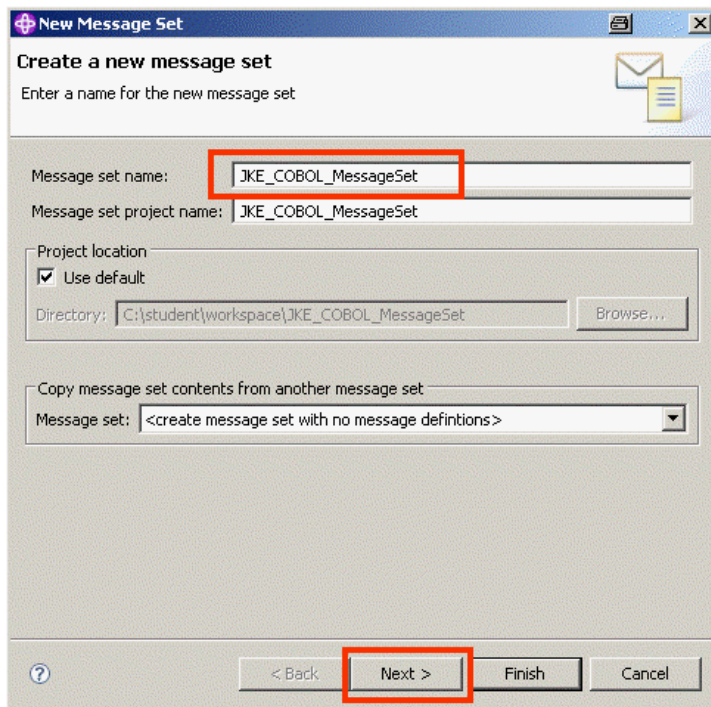
In the second part of the lab, you will modify the IntroLab message flow to parse a COBOL message.

The JKE_COBOL_MessageSet will be used again later in other labs.

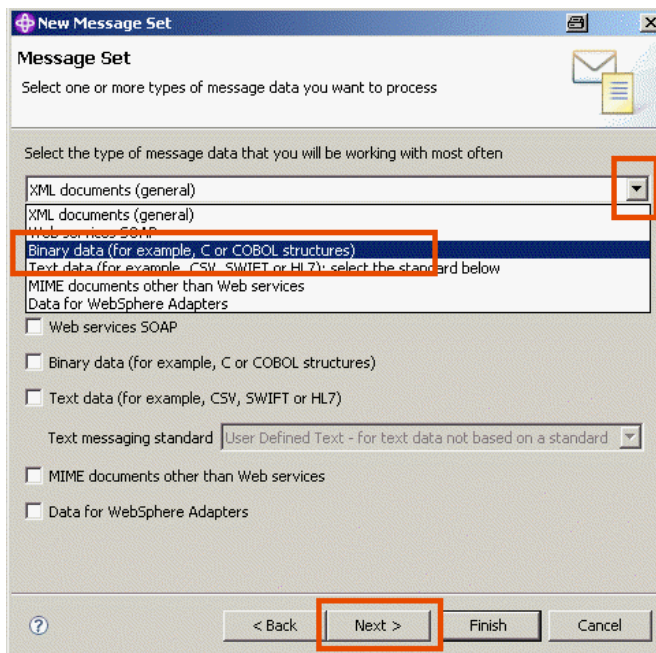


You will now create a Message Set and a Message Set Project.

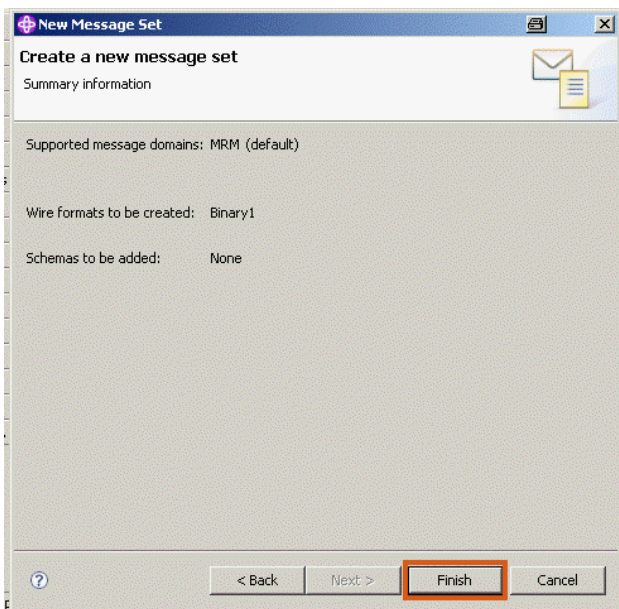
- __1. **Right-click in the white space** in the **Broker Development** pane.
- __2. Select **New**.
- __3. Select **Message Set**.



- __4. Enter JKE_COBOL_MessageSet as the Message set name. The Message set project name will be set also. **Be sure to enter this name correctly including case!! Other supplied artifacts depend on it.**
- __5. Click **Next**.



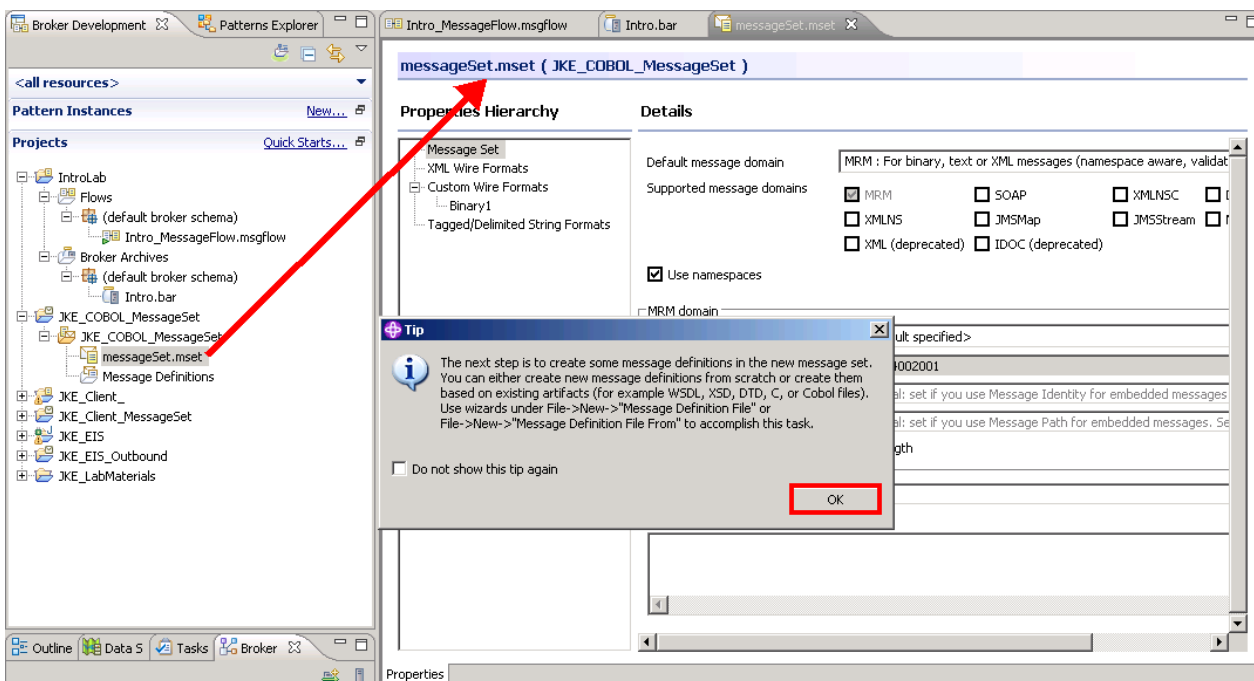
- __6. Use the pull-down for the message data type and select **Binary data**.
- __7. Click **Next**.



This is a summary panel of the options chosen. A Confirmation panel is displayed and the new Message Set is opened automatically. There are no changes required for this lab.

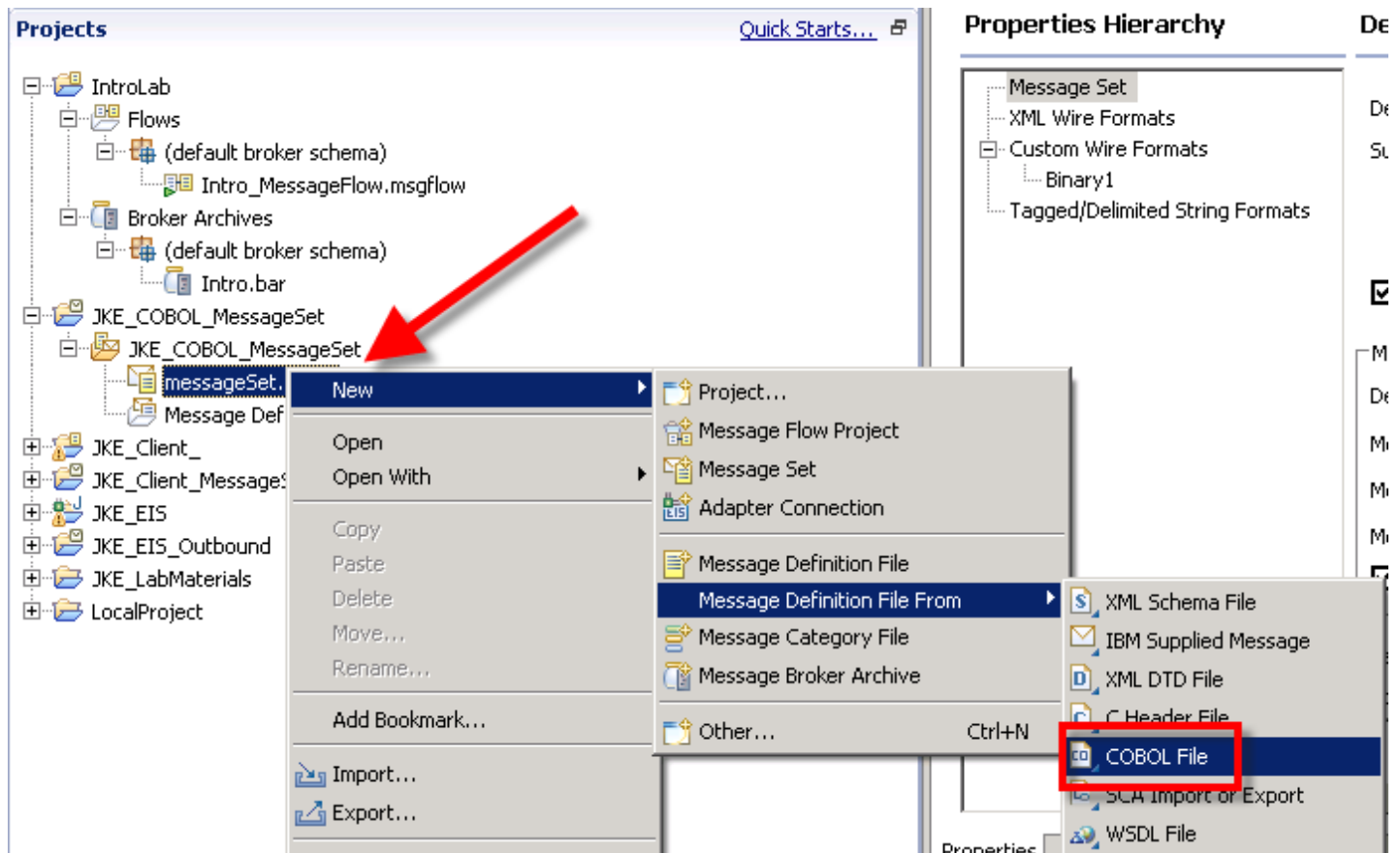
__8. Click **Finish**.

The next step will be to provide the message definition information by importing a COBOL copybook.

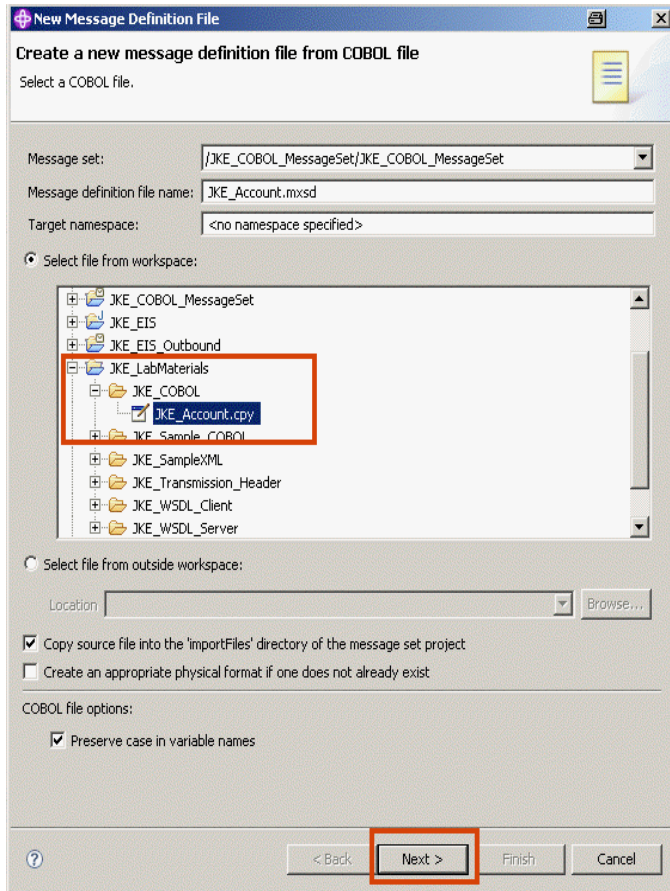


The new message set definition is opened automatically. A “Hint” is shown that identifies the next step.

__9. Click **OK** to dismiss the “Hint”.

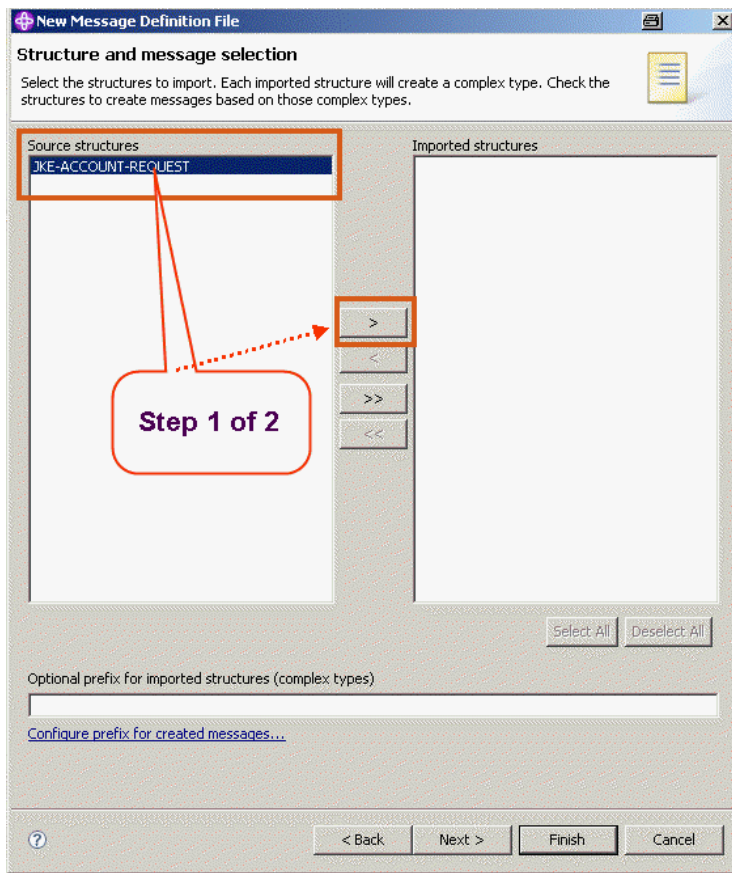


- __10. Highlight the **messageSet.mset** under the **JKE_COBOL_MessageSet**
- __11. Press the right mouse button.
- __12. Select **New** from the menu.
- __13. Select **Message Definition File From->COBOL File**.



This panel allows you to identify the location of the definition to be imported. There are two options: The default is to search the files in the workspace. The other option is to search the file system.

- __14. Expand **JKE_LabMaterials->JKE_COBOL**.
- __15. Select **JKE_ACCOUNT.cpy**.
- __16. Click **Next**.

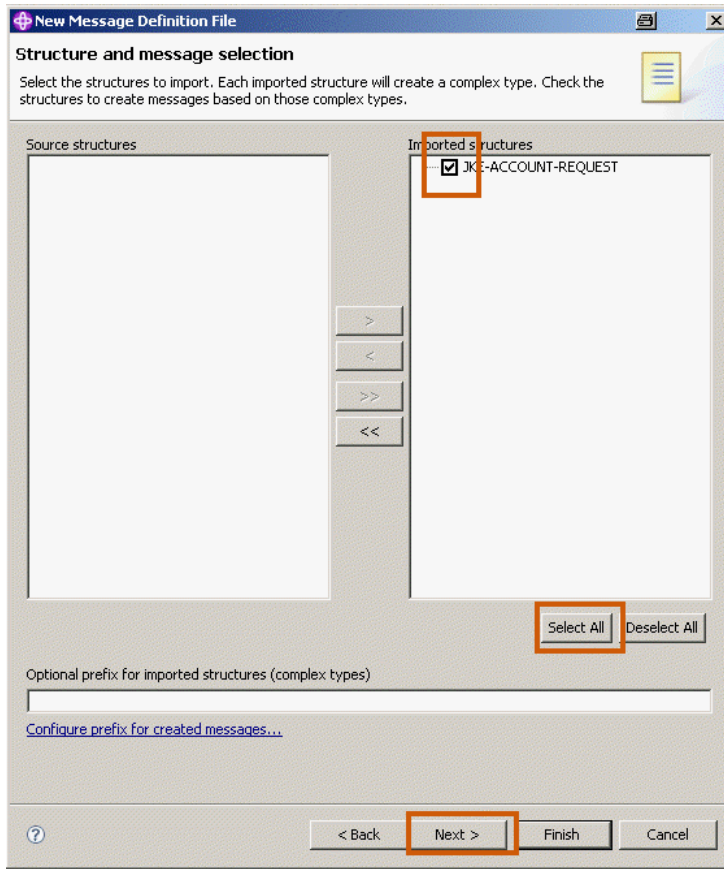


A COBOL copybook may contain multiple record definitions. Each record definition is shown on the left side.

__17. Highlight **JKE-ACCOUNT-REQUEST**.

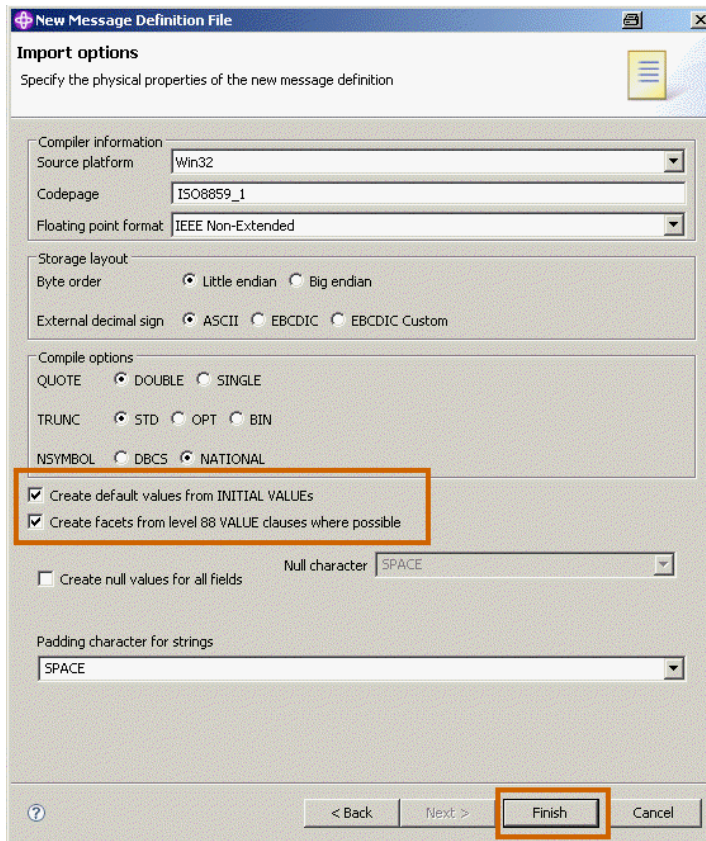
__18. Click the ➤ symbol in the middle.

The second step is shown on the next page.



__19. Click **Select All**

__20. Click **Next**.



Many of the options presented on this panel are related to compiler options.

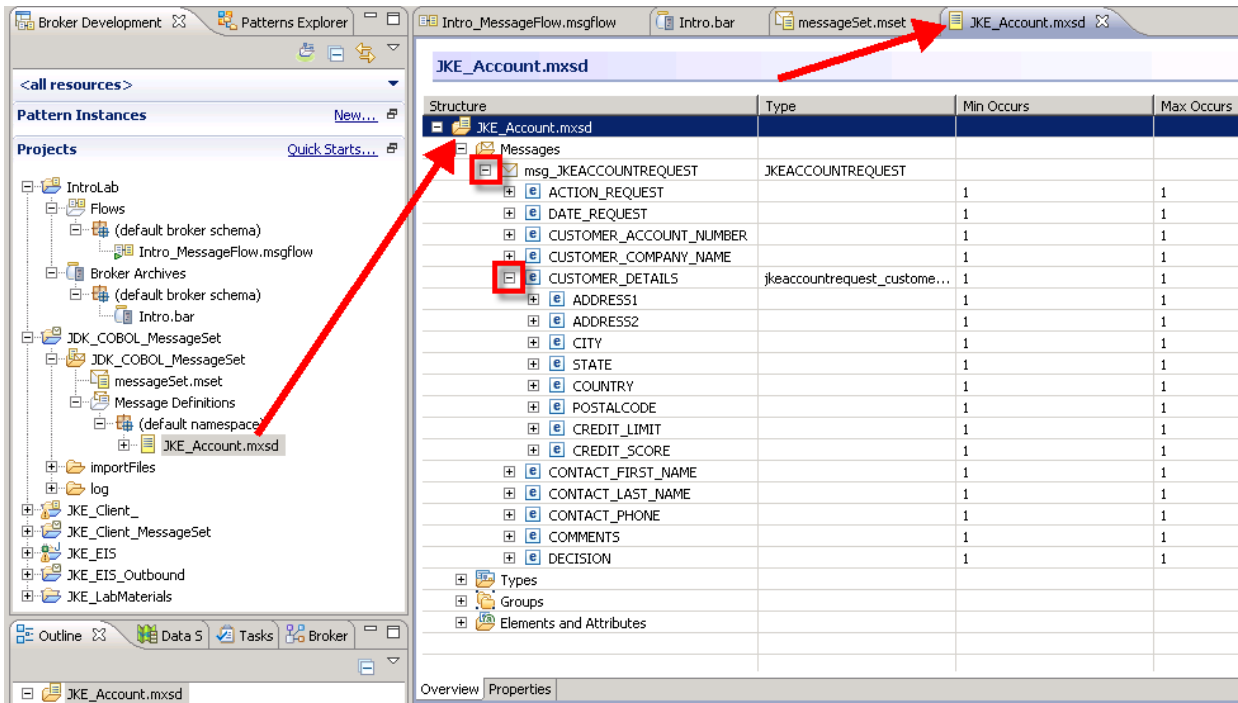
__21. Select the check boxes for **Create default values...** and **Create facets from ...**

When dealing with fixed format definitions such as COBOL or C, every field in the structure must have some content. The Create default values options allows the Importer to create a default value automatically when it is part of the copybook. You may also provide such definitions by manually updating the message definition.

The Create facets option creates constants based on a level 88 definition which is used for the same purpose by a COBOL program.

These options are not actually used in the labs, but this is an important capability of the importer.

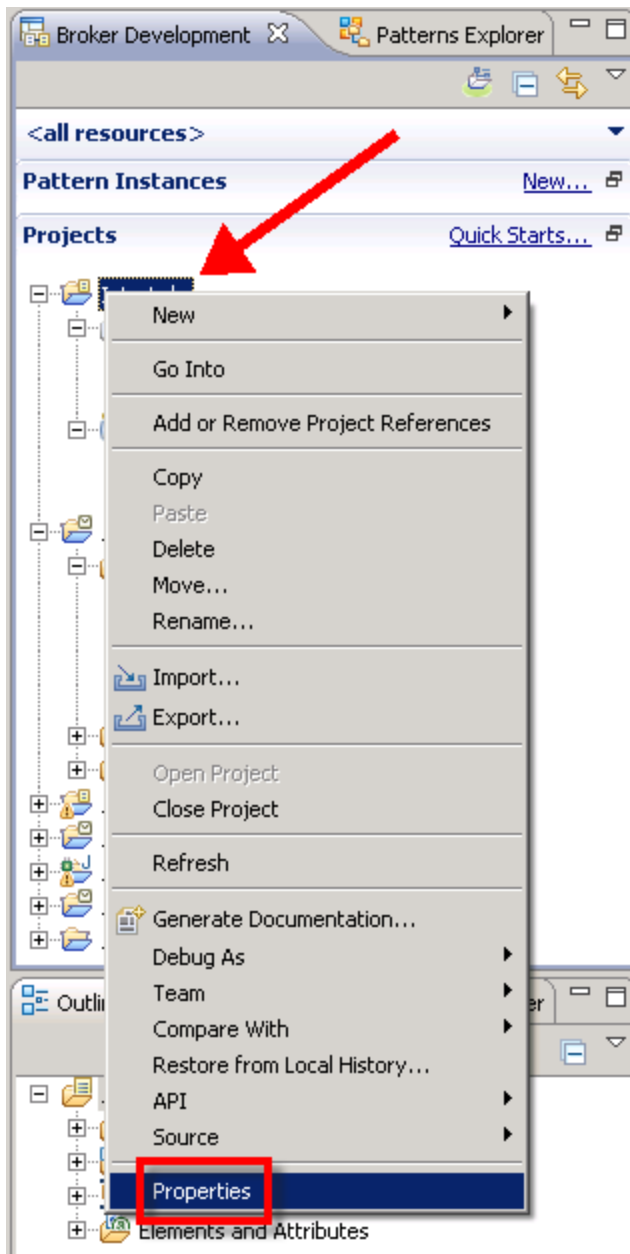
__22. Click **Finish**.



The Message Set definition is opened automatically.

__23. Expand the definitions to see the field names and structure as shown.

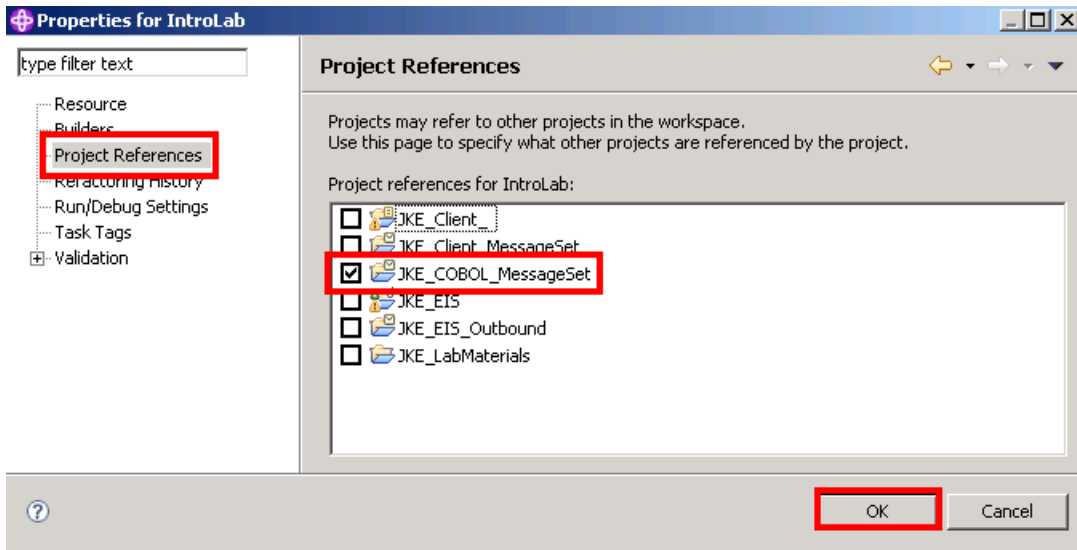
__24. Close this editor and the message set editor.



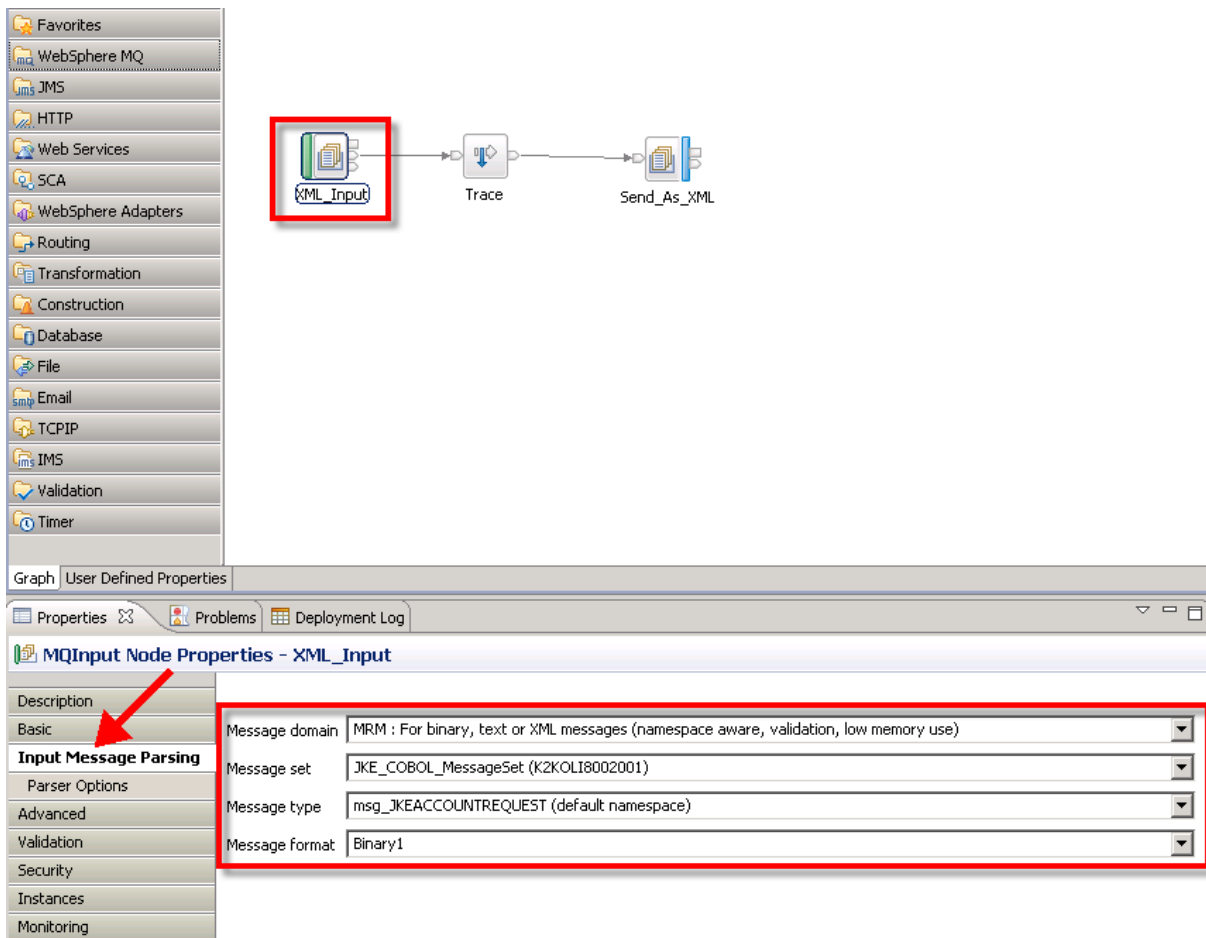
The message flow will now be changed to process a COBOL message rather than an XML message.

Before the message flow can use a message set, a relationship between the **IntroLab** message flow project and the message set must be defined. This is done by creating a **Project Reference**.

- __25. In the **Broker Developer** navigator pane, highlight the **IntroLab** message flow project
- __26. Press the right mouse button.
- __27. Select **Properties**.



- __28. Select **Project References**.
- __29. Click the check box for the **JKE_COBOL_MessageSet** from the list of projects
- __30. Click **OK**.



__31. Return to your message flow and click on the **XML_Input** node

__32. Click **Input Message Parsing**

__33. Use the pull-downs for each of the four selection entries, start at the top and select the following:

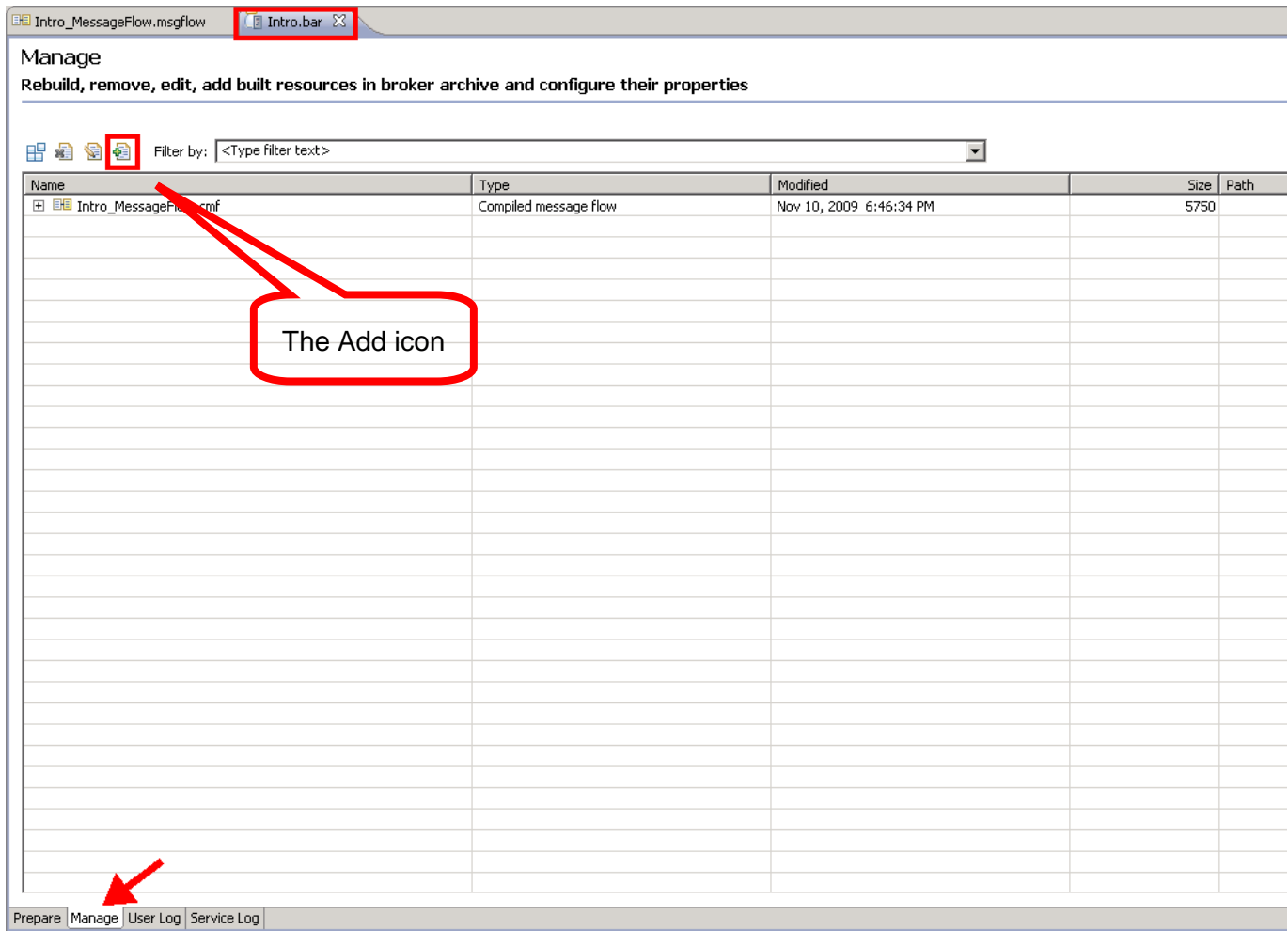
Message domain: **MRM**

Message set: **JKE_COBOL_MessageSet**

Message type: **msg_JKEACCOUNTREQUEST** – this name comes from the 01 level of the COBOL copybook. The importer removes the dashes from the names.

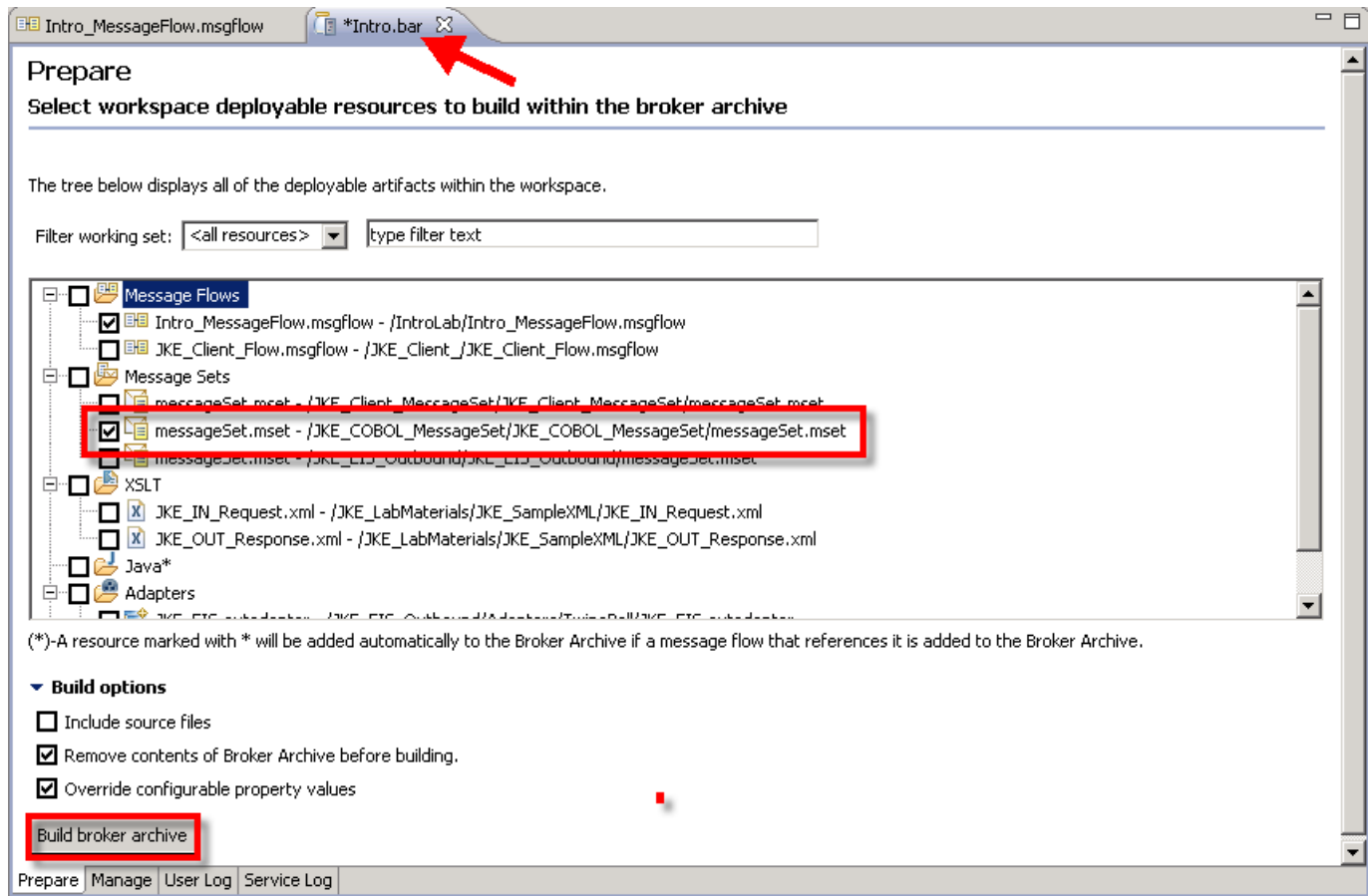
Message format: **Binary1**

__34.  Save the message flow.



__35. Click on the tab for the **Intro.bar** bar file editor session.

__36. Click the **Add** icon (the icon on the right) to add the JKE_COBOL_MessageSet.

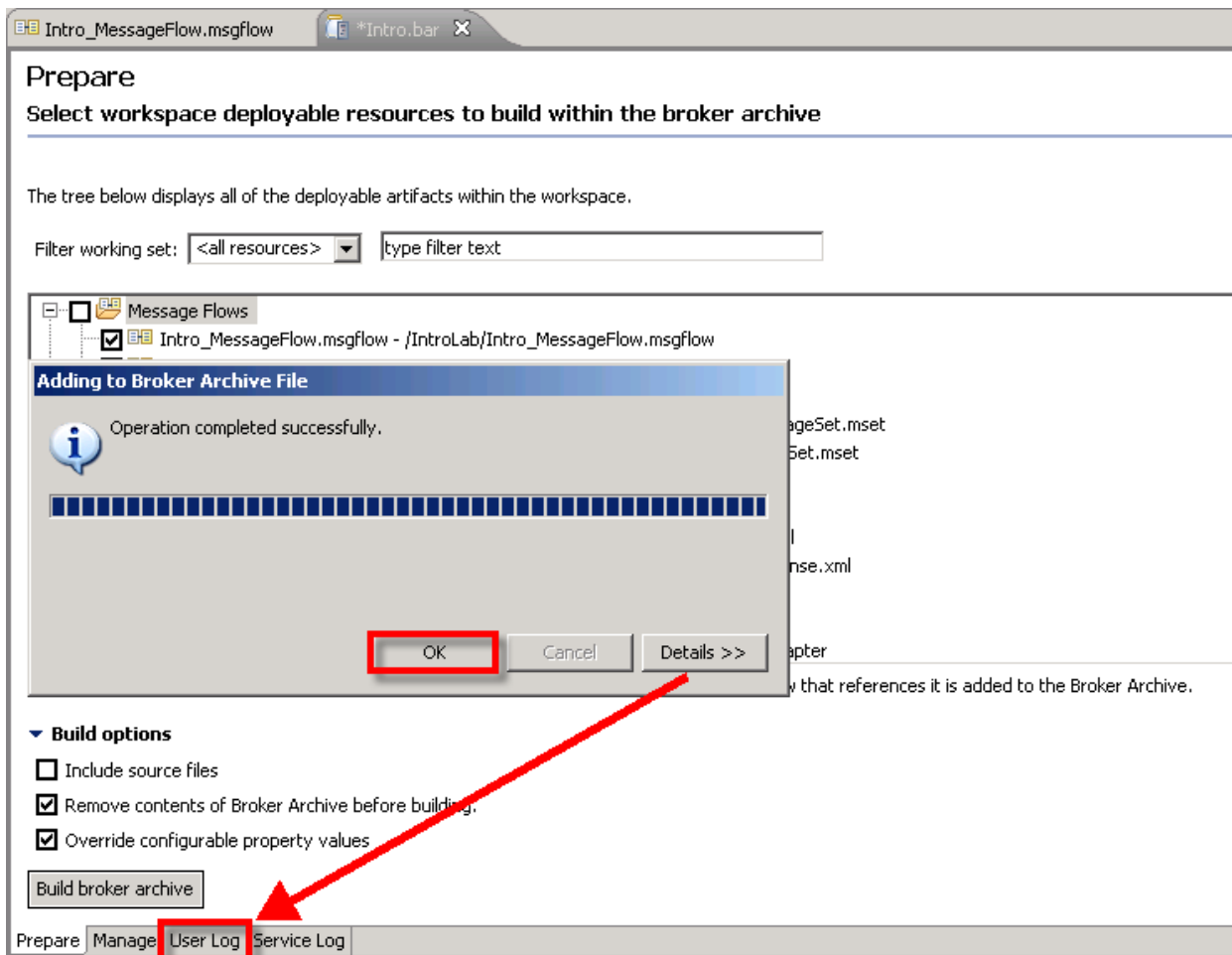


The editor switches to the Prepare view.

__37. Click the check-box for **message.mset - /JKE_COBOL_MessageSet** entry.

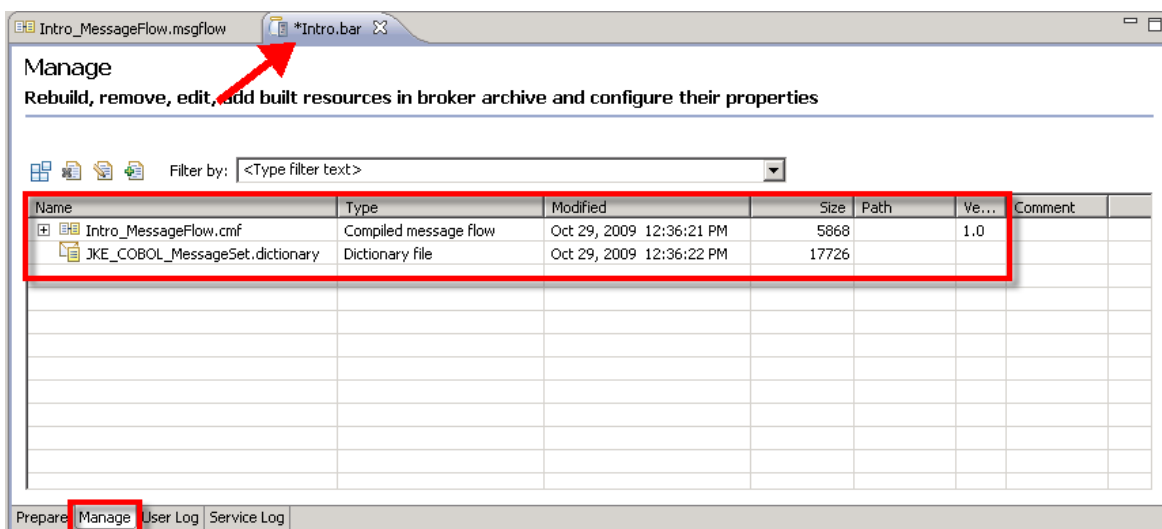
__38. Make sure the Intro_MessageFlow is still checked as it has changed and needs to be refreshed in the BAR file.

__39. Click **Build broker archive**.



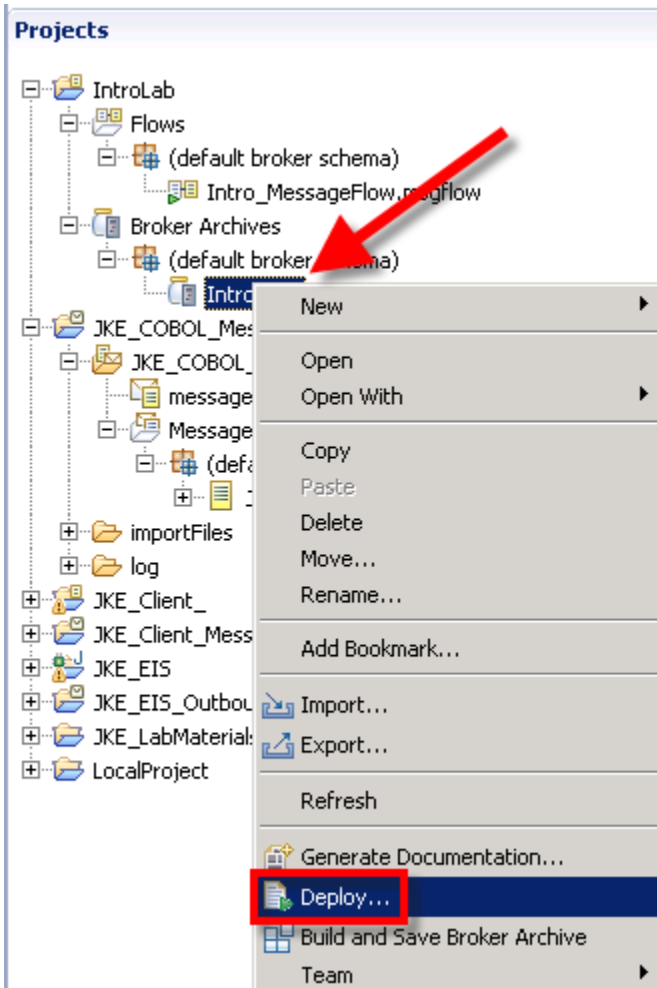
The build progress panel is displayed. The Details option provides more information about the build process for the artifacts listed. This same information is available in the User Log tab at the bottom of the pane. This information is removed anytime the bar file is rebuilt.

__40. Click **OK** to dismiss this panel.



__41. Click on the **Manage** tab. There are now two items listed in the bar file.

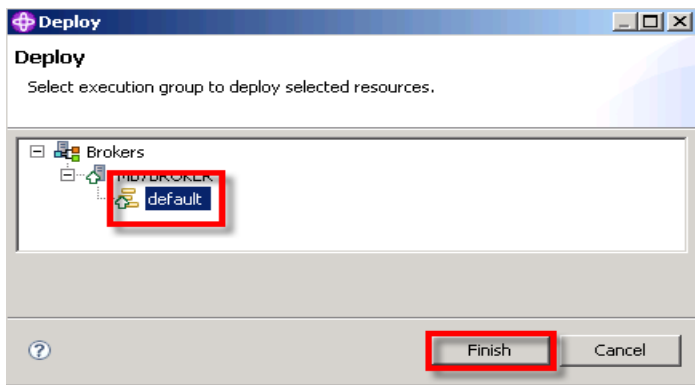
__42.  Save the bar file.



__43. Select the **Intro**.bar file in the navigator pane.

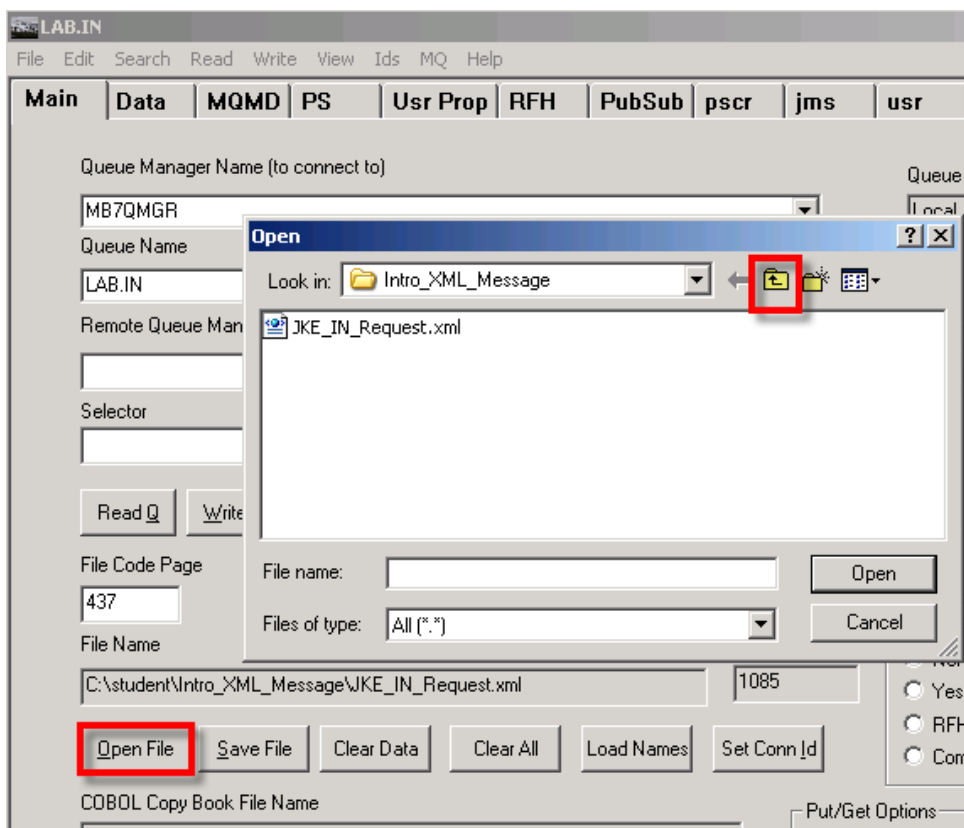
__44. Press the right mouse button.

__45. Select **Deploy** from the menu.



__46. Select the **default** Execution Group.

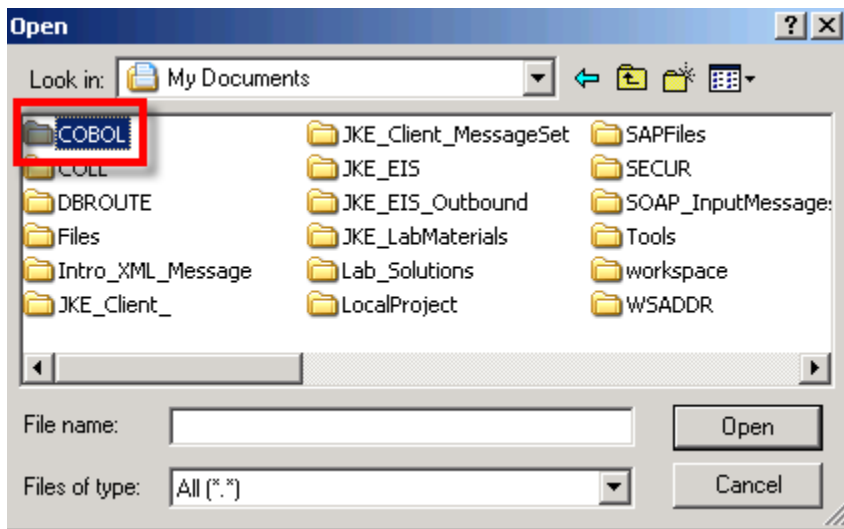
__47. Click **Finish**.



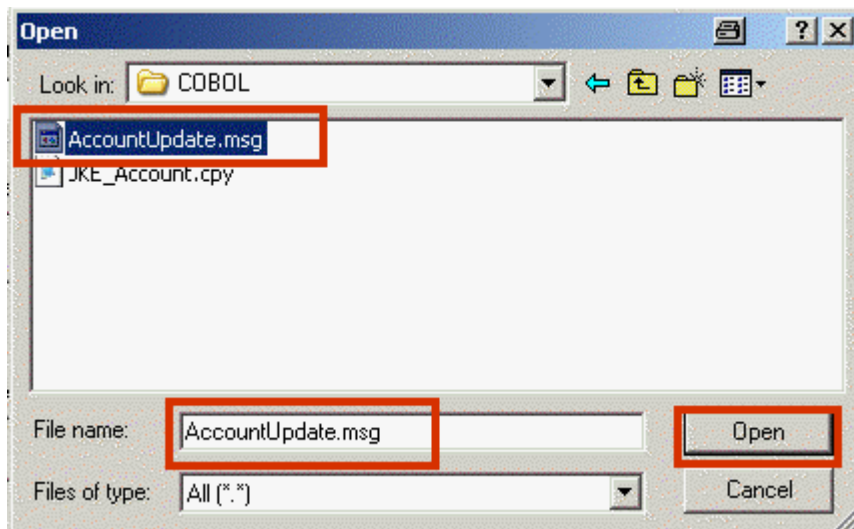
__48. Click **LAB.IN** in the Windows taskbar to bring the RFHUtil window into focus.

__49. Click **Open File**.

__50. Click the **Up Arrow**. The COBOL message is in another directory.

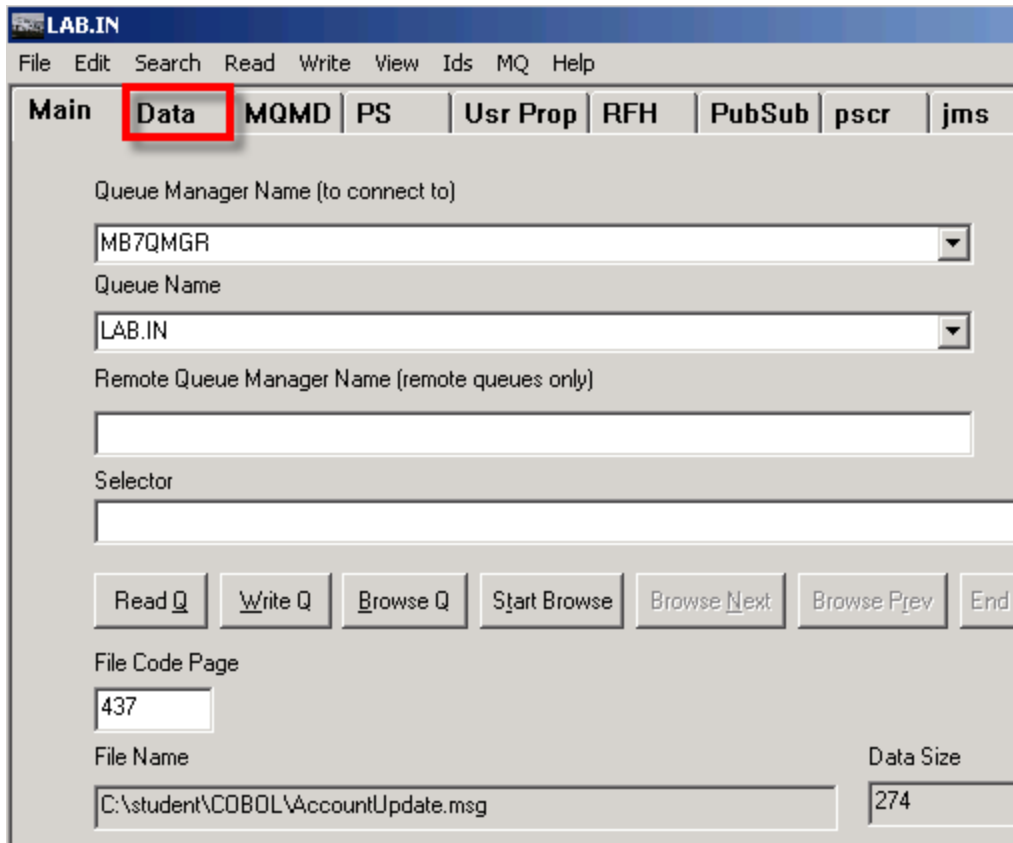


__51. Select the **COBOL** directory.

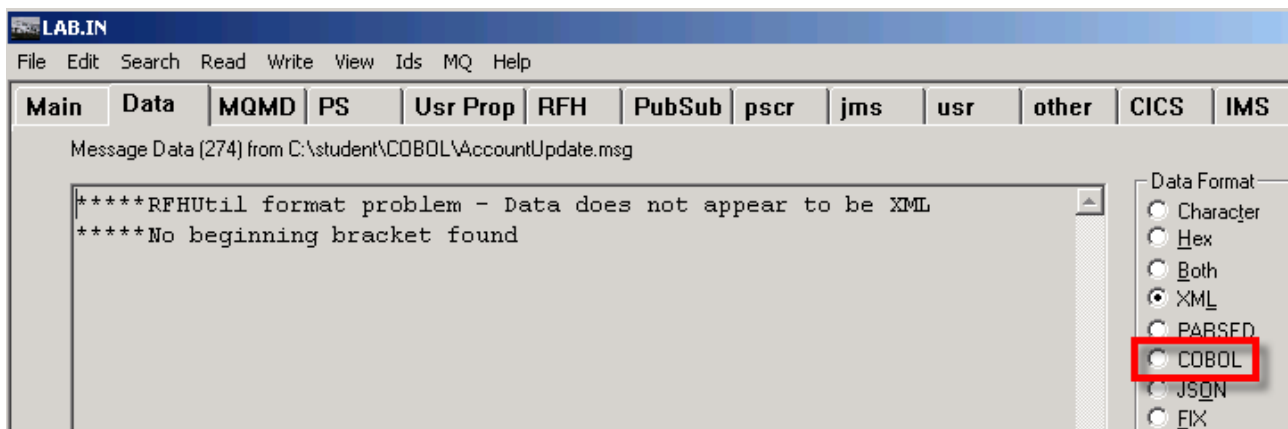


__52. Select **AccountUpdate.msg**.

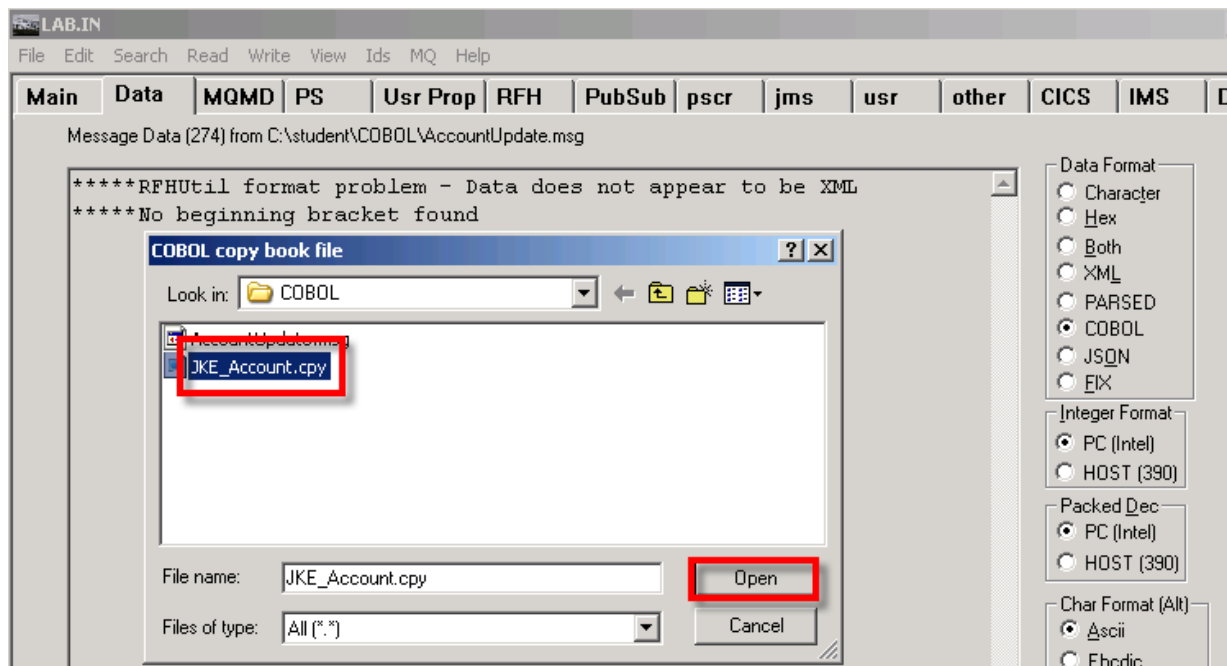
__53. Click **Open**.



__54. Click the RfhUtil **Data** tab



__55. The last time you were using an XML message. Click the radio button for **COBOL**



The location of the copybook must be specified.

__56. Select **JKE_Account.cpy**

__57. Click **Open**.

LAB.IN
File Edit Search Read Write View Ids MQ Help

Main Data MQMD PS Usr Prop RFH PubSub pscr jms usr other CICS IMS

Message Data (274) from C:\student\COBOL\AccountUpdate.msg

Level	Ofs	Len	Type	Occ	Variable Name	Value
01	0	274			JKE-ACCOUNT-REQUEST	
05	0	1	CHAR		ACTION-REQUEST	U
05	1	10	CHAR		DATE-REQUEST	10/12/2007
05	11	10	CHAR		CUSTOMER-ACCOUNT-NUMBER	1
05	21	30	CHAR		CUSTOMER-COMPANY-NAME	ACME
05	51	119			CUSTOMER-DETAILS	
10	51	30	CHAR		ADDRESS1	1254 Main St
10	81	30	CHAR		ADDRESS2	Suite 12
10	111	20	CHAR		CITY	Dime Box
10	131	2	CHAR		STATE	TX
10	133	20	CHAR		COUNTRY	USA
10	153	10	CHAR		POSTALCODE	76543
10	163	5	PD		CREDIT-LIMIT	000001200
10	168	2	INT		CREDIT-SCORE	0082
05	170	15	CHAR		CONTACT-FIRST-NAME	Freddy
05	185	20	CHAR		CONTACT-LAST-NAME	Bloggs
05	205	18	CHAR		CONTACT-PHONE	555-123-6543
05	223	50	CHAR		COMMENTS	Just a Comment
05	273	1	CHAR		DECISION	Y

Copy book size 274, Data area size 274

Data Format
 Character
 Hex
 Both
 XML
 PARSED
 COBOL
 JSON
 FIX

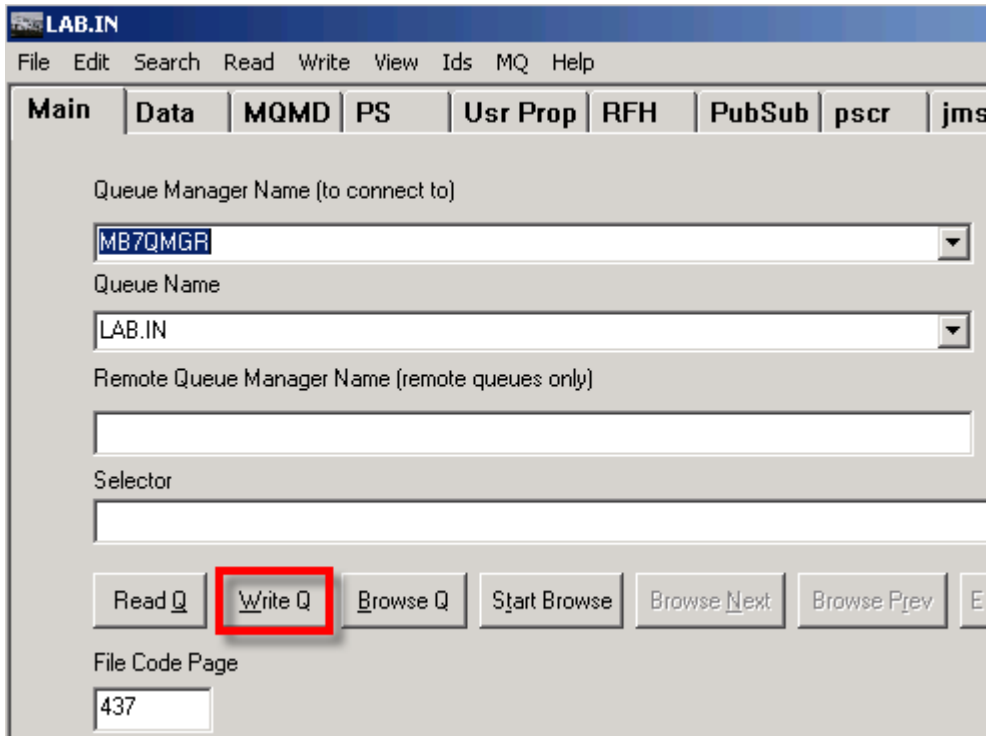
Integer Format
 PC (Intel)
 HOST (390)

Packed Dec
 PC (Intel)
 HOST (390)

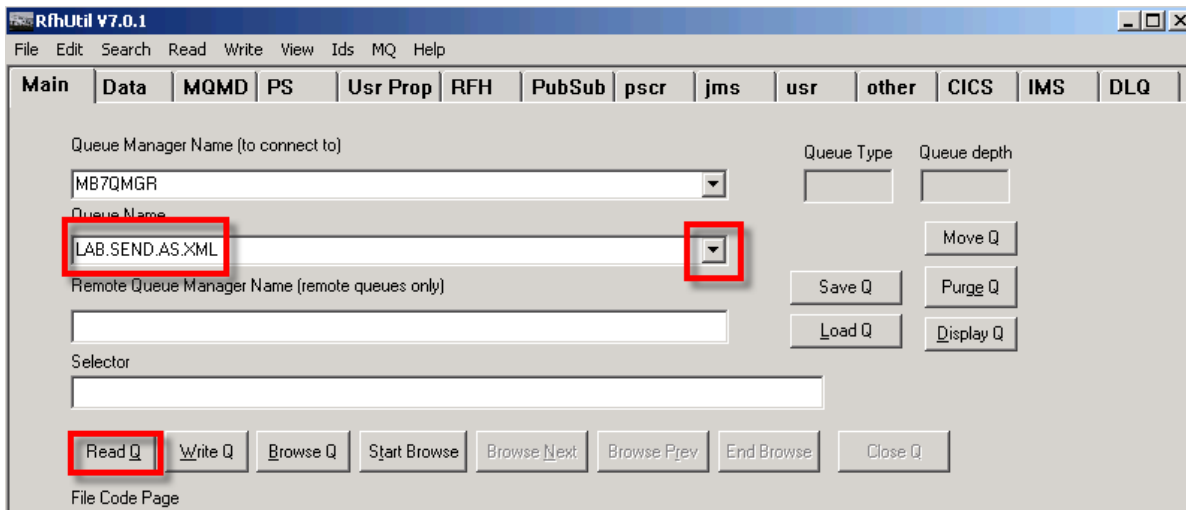
Char Format (Alt)
 Ascii
 Ebcidic
 Simp Chinese
 Korean
 Trad Chinese
 Japanese
 Thai

The message is now matched against the copybook. There are many other display options available as shown on the right side.

__58. Click on the **Main** tab.



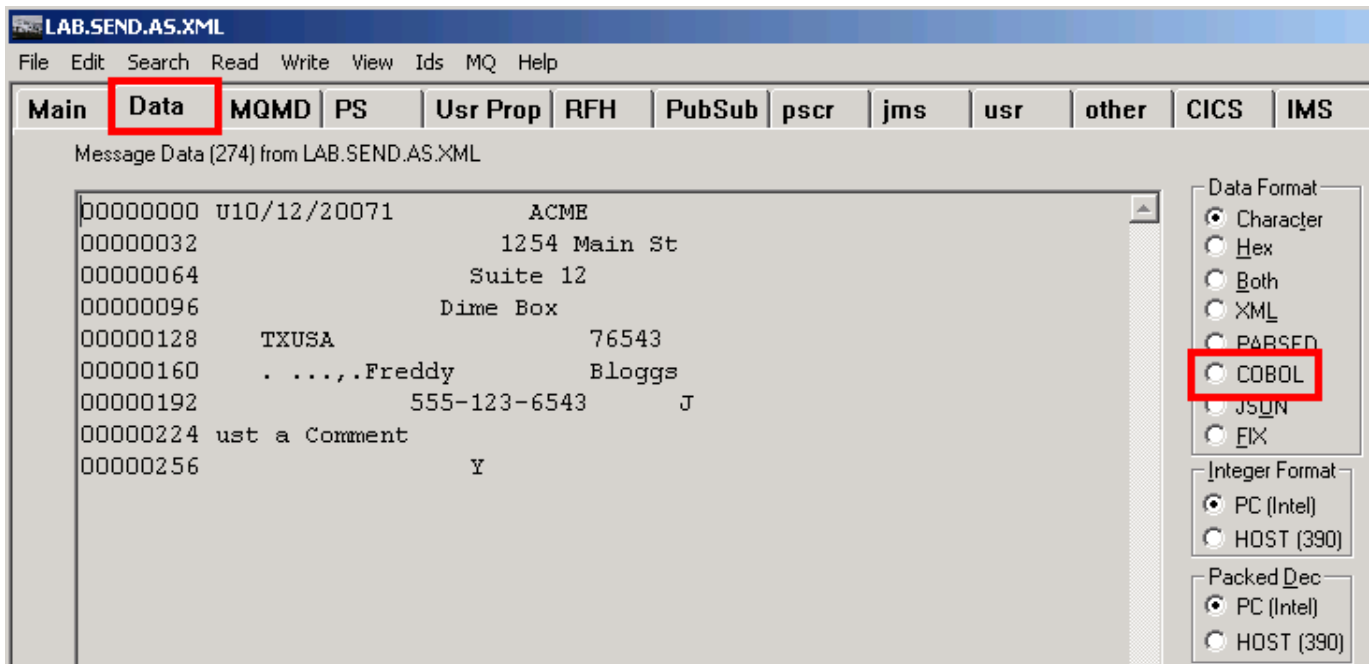
__59. Click **Write Q** to send the message.



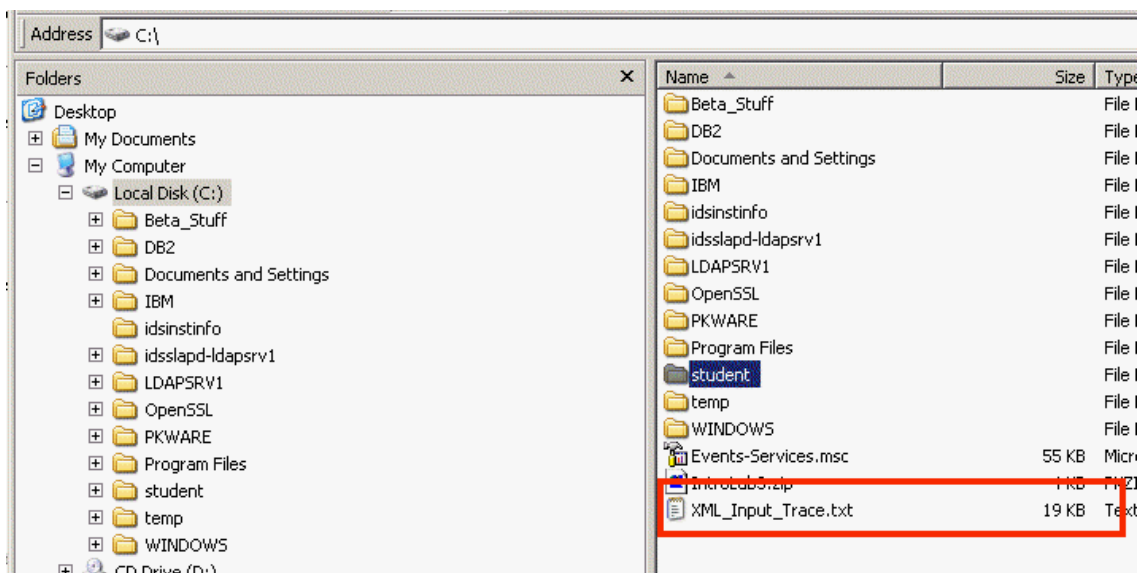
__60. Start another copy of RFHUtil.

__61. Use the Queue Name pull-down menu to select **LAB.SEND.AS.XML** as the **Queue Name**.

__62. Click **Read Q**. You may have multiple messages in the queue from prior executions of the flow so you may have to click the Read Q button multiple times. You are looking for a message that is 274 bytes in length.



- __63. Switch to the Data tab. Here is the raw, fixed format COBOL output message.
- __64. If you click the COBOL radio button and select the JKE_Account.cpy file the copybook will be used to format the data.
- __65. Close both copies of RFHUtil.



- __66. Open Windows Explorer and open the **XML_Input_Trace.txt** file.

```

Here is some text
(['MQOROOT' : 0xca35a81
(0x01000000:Name):Properties = ( ['MQPROPERTYPARSER' : 0x13fd9d70]
(0x03000000:NameValue):MessageSet = 'JKE_COBOL_MessageSet' (CHARACTER)
(0x03000000:NameValue):MessageType = '{}:msg_JKEACCOUNTREQUEST' (CHARACTER)
(0x03000000:NameValue):MessageFormat = 'Binary1' (CHARACTER)
(0x03000000:NameValue):Encoding = 546 (INTEGER)
(0x03000000:NameValue):CodedCharSetId = 437 (INTEGER)
(0x03000000:NameValue):Transactional = TRUE (BOOLEAN)
(0x03000000:NameValue):Persistence = FALSE (BOOLEAN)
(0x03000000:NameValue):CreationTime = GMTTIMESTAMP '2009-10-29 17:39:40.320' (GMTTIMESTAMP)
(0x03000000:NameValue):ExpirationTime = -1 (INTEGER)
(0x03000000:NameValue):Priority = 0 (INTEGER)
(0x03000000:NameValue):ReplyIdentifier = X'000000000000000000000000000000000000000000000000' (BLOB)
(0x03000000:NameValue):ReplyProtocol = 'MQ' (CHARACTER)
(0x03000000:NameValue):Topic = NULL
(0x03000000:NameValue):ContentType = '' (CHARACTER)
(0x03000000:NameValue):IdentitySourceType = '' (CHARACTER)
(0x03000000:NameValue):IdentitySourceToken = '' (CHARACTER) ←
(0x03000000:NameValue):IdentitySourcePassword = '' (CHARACTER)
(0x03000000:NameValue):IdentitySourceIssuedBy = '' (CHARACTER)
(0x03000000:NameValue):IdentityMappedType = '' (CHARACTER)
(0x03000000:NameValue):IdentityMappedToken = '' (CHARACTER)
(0x03000000:NameValue):IdentityMappedPassword = '' (CHARACTER)
(0x03000000:NameValue):IdentityMappedIssuedBy = '' (CHARACTER)
)
(0x01000000:Name):MQMD = ( ['MQHMD' : 0xe6cee8]
(0x03000000:NameValue):SourceQueue = 'LAB.IN' (CHARACTER)
(0x03000000:NameValue):Transactional = TRUE (BOOLEAN)
(0x03000000:NameValue):Encoding = 546 (INTEGER)
(0x03000000:NameValue):CodedCharSetId = 437 (INTEGER)
(0x03000000:NameValue):Format = '' (CHARACTER)
(0x03000000:NameValue):Version = 2 (INTEGER)
(0x03000000:NameValue):Report = 0 (INTEGER)
(0x03000000:NameValue):MsgType = 8 (INTEGER)
(0x03000000:NameValue):Expiry = -1 (INTEGER)
(0x03000000:NameValue):Feedback = 0 (INTEGER)
(0x03000000:NameValue):Persistence = 0 (INTEGER)
(0x03000000:NameValue):MsgID = X'414d51204d4237514d47522020202020edc7e84a20003b04' (BLOB)
(0x03000000:NameValue):CorrelId = X'000000000000000000000000000000000000000000000000' (BLOB)
(0x03000000:NameValue):BackoutCount = 0 (INTEGER)
(0x03000000:NameValue):ReplyToMgr = '' (CHARACTER)
(0x03000000:NameValue):ReplyToMgr = 'MB7QMGR' (CHARACTER)
(0x03000000:NameValue):UserIdentifier = 'admin' (CHARACTER)
(0x03000000:NameValue):AccountingToken = X'1601051500000092e03c779b0bc11e75b9754ee030000000000000000000000' (BLOB)
)
)
    
```

__67. Scroll down to the bottom of the file and then scroll up to where you can see the Properties folder (Ctrl + End works.)

The first three items contain the information that you selected in the XML_Input node.


At the bottom of the Properties folder there is information about additional userid verification that will be discussed on the last day of the Proof of Technology (PoT).

In Lab 9, you will be dealing with the MsgID and CorrelId fields in the WebSphere MQ Message Descriptor (MQMD). These are also highlighted.

```

(0x03000000:Namevalue):PutTime           = GMTTIME '19:22:47.950' (GMTTIME)
(0x03000000:Namevalue):ApplooriginData    = ' ' (CHARACTER)
(0x03000000:Namevalue):GroupID           = X'0000000000000000000000000000000000000000000000000000000000000000' (BLOB)
(0x03000000:Namevalue):MsgSeqNumber      = 1 (INTEGER)
(0x03000000:Namevalue):Offset            = 0 (INTEGER)
(0x03000000:Namevalue):MsgFlags          = 0 (INTEGER)
(0x03000000:Namevalue):OriginalLength    = -1 (INTEGER)
)
(0x01000021:Name+):MRM                    = ( ['mrm' : 0x7930ca8]
(0x0300000B:Namevalue+):ACTION_REQUEST    = 'U' (CHARACTER)
(0x0300000B:Namevalue+):DATE_REQUEST      = '10/12/2007' (CHARACTER)
(0x0300000B:Namevalue+):CUSTOMER_ACCOUNT_NUMBER = '1' (CHARACTER)
(0x0300000B:Namevalue+):CUSTOMER_COMPANY_NAME = 'ACME' (CHARACTER)
(0x01000013:Name+ ):CUSTOMER_DETAILS     = (
(0x0300000B:Namevalue+):ADDRESS1          = '1254 Main St' (CHARACTER)
(0x0300000B:Namevalue+):ADDRESS2          = 'Suite 12' (CHARACTER)
(0x0300000B:Namevalue+):CITY              = 'Dime Box' (CHARACTER)
(0x0300000B:Namevalue+):STATE            = 'TX' (CHARACTER)
(0x0300000B:Namevalue+):COUNTRY          = 'USA' (CHARACTER)
(0x0300000B:Namevalue+):POSTALCODE       = '76543' (CHARACTER)
(0x0300000B:Namevalue+):CREDIT_LIMIT     = 202001200 (INTEGER)
(0x0300000B:Namevalue+):CREDIT_SCORE     = 8315 (INTEGER)
)
(0x0300000B:Namevalue+):CONTACT_FIRST_NAME = 'Freddy' (CHARACTER)
(0x0300000B:Namevalue+):CONTACT_LAST_NAME  = 'Bloggs' (CHARACTER)
(0x0300000B:Namevalue+):CONTACT_PHONE     = '555-123-6543' (CHARACTER)
(0x0300000B:Namevalue+):COMMENTS         = 'Just a Comment' (CHARACTER)
(0x0300000B:Namevalue+):DECISION          = 'Y' (CHARACTER)
)
)
)

```



At the bottom of the file is the COBOL message. Note that the MRM parser is identified in the first line above the ACTION_REQUEST field. Also, note that the 01 level name from the COBOL copybook is not present in the message tree. This was used to name the message definition.

- __68. Close notepad editors.
- __69. Close the Windows Explorer.
- __70. Leave the WebSphere Message Broker Toolkit open.
- __71. Close any open editors (message flow, bar file, etc) in the WebSphere Message Broker Toolkit but not the WebSphere Message Broker Toolkit itself.

This is the end of Lab 4 and the Introduction Labs.

Lab 5 Building the JKE Server – Web Services

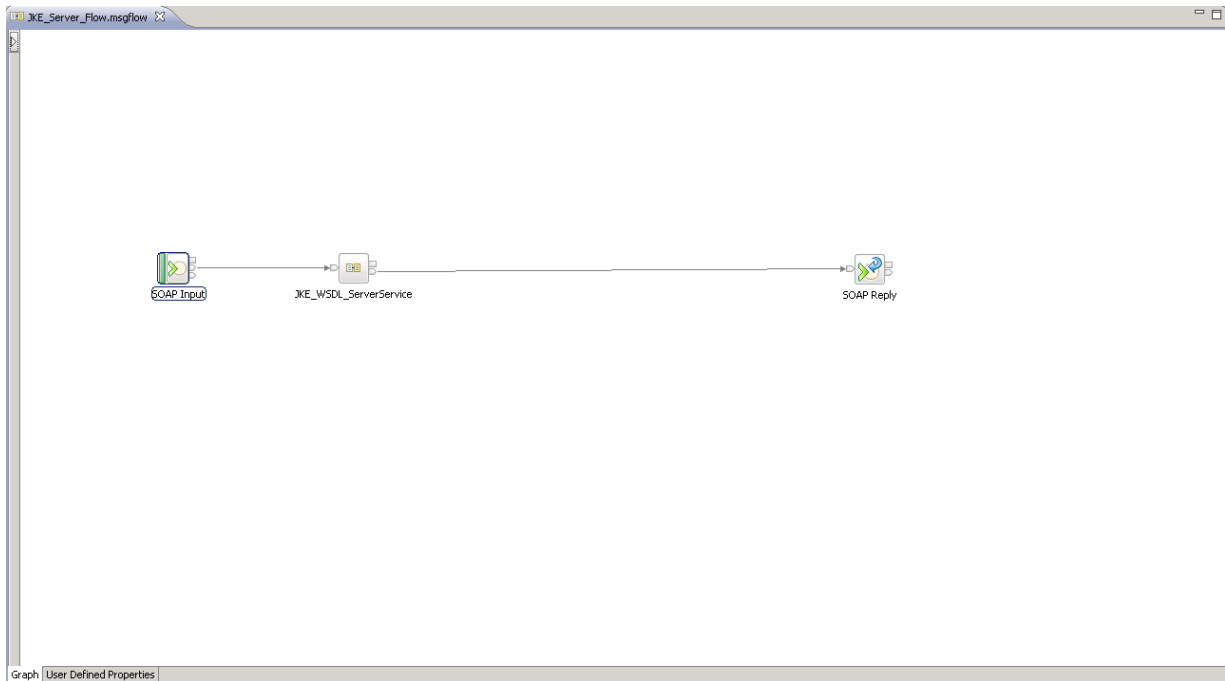
5.1 Lab Overview

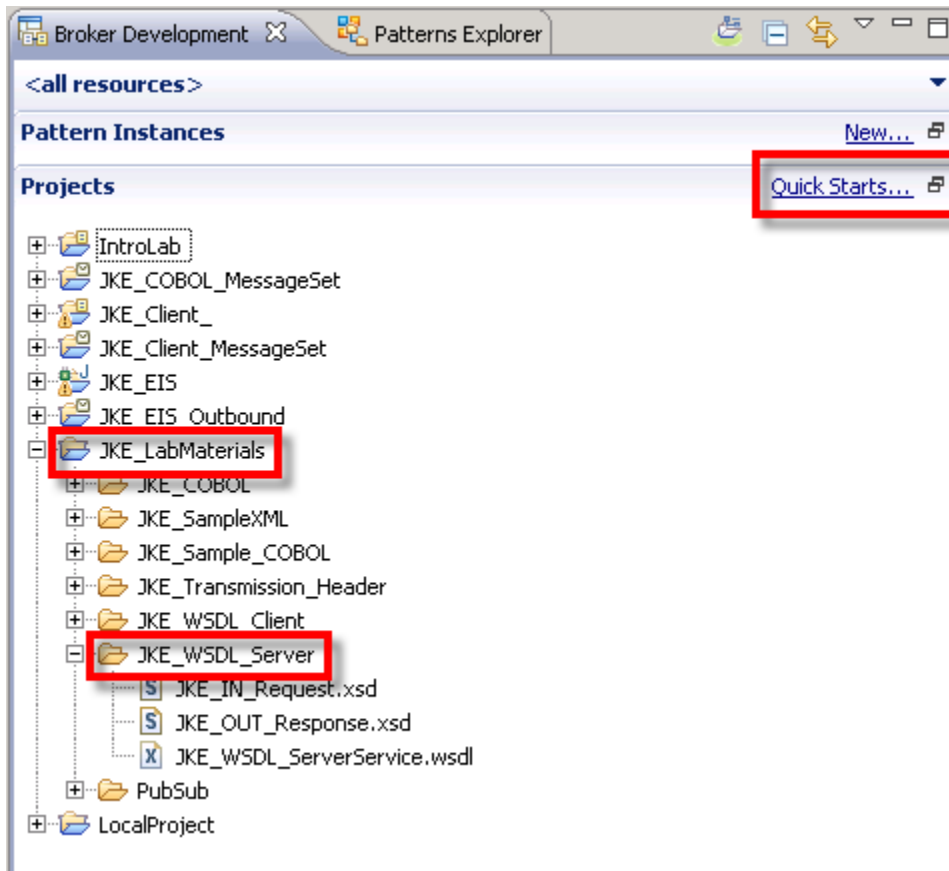
This is the first of five labs where you will build the JKE_Server Message Flow and its associated Message Sets. Each lab will build upon the prior lab and will be preceded by a set of lectures.

In this lab you will be using one of the wizards to build both a message set and a message flow based on a Web Services Description Language (WSDL) definition file that is provided.

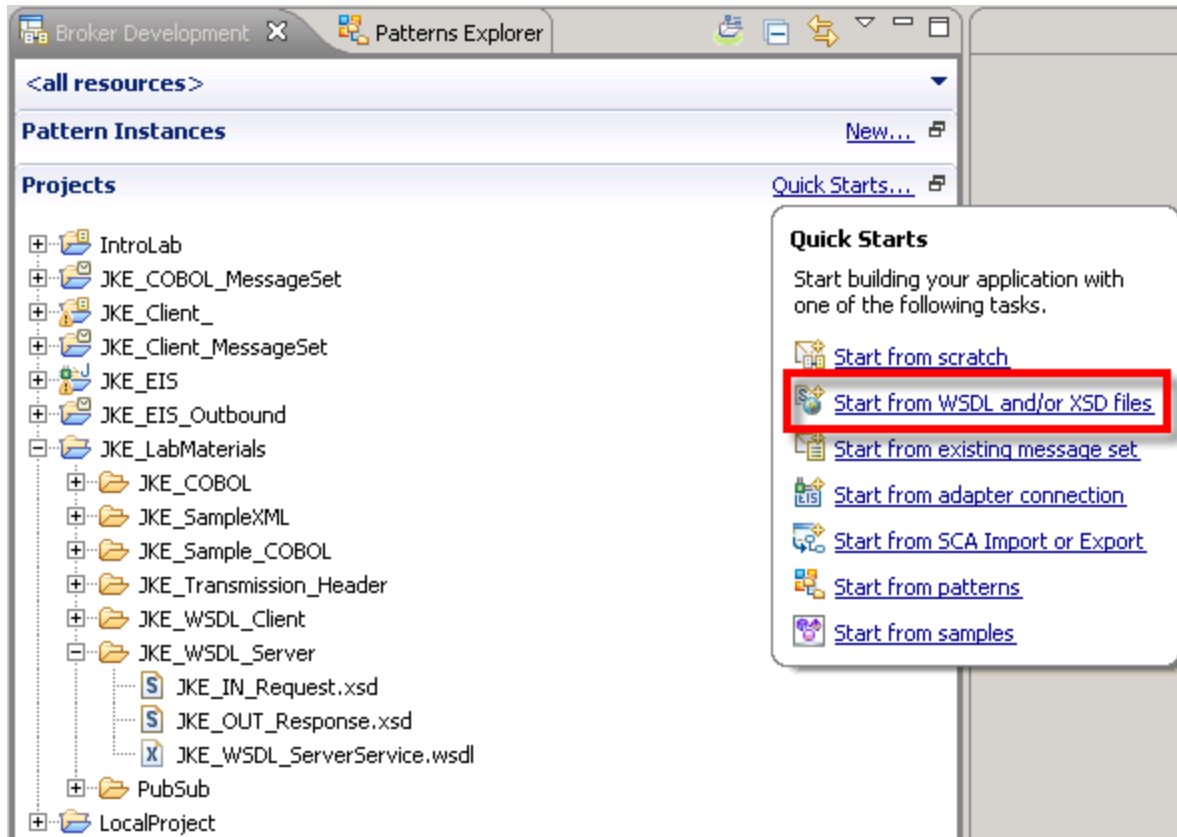
You will also use a “drag and drop” wizard to build the basic message flow that uses the SOAP nodes to implement the message flow as a web service. You will then test the message flow using the Test Client.

The following is what you will build and test. This will be expanded in the next four Labs.





- __1. Expand the **JKE_LabMaterials** folder.
- __2. Expand the **JKE_WSDL_Server** folder. This is where the WSDL and associated schemas are stored. You may also use a WSDL and its schemas from the file system.
- __3. Select the **Quick Starts** hot spot to display the quick starts menu.



4. Click on **Start from WSDL and/or XSD files**. This starts a wizard that will create a message set, import the WSDL to define any required messages and create a message flow project and message flow based on the names you provide to the wizard, as described in the following steps.

Quick Start

New Message Broker Application

Set up the basic resources required to develop a Message Broker application using WSDL and XSD files as a starting point.

Message flow project name:

Message set project name:

Message set name:

Message flow creation

Create a new message flow in my flow project

Message flow name:

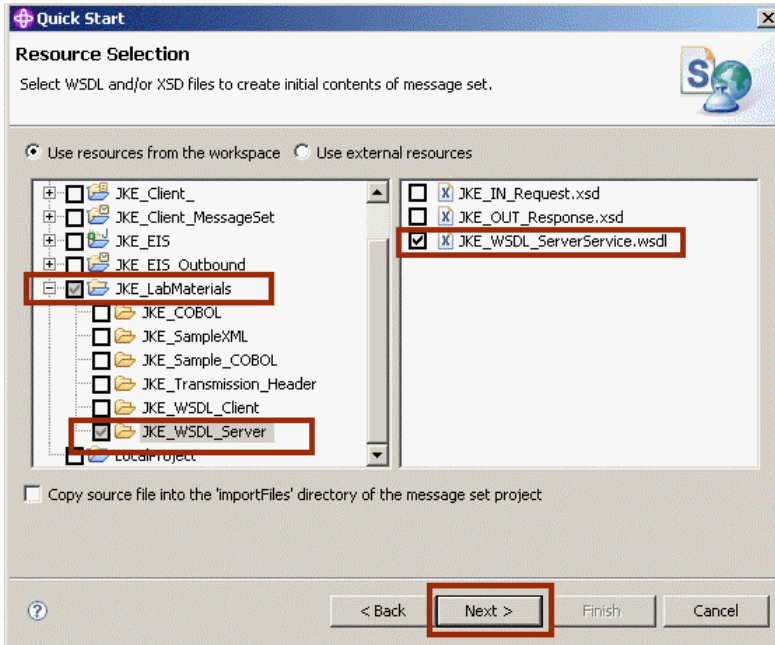
Working set creation

Create a new working set for these resources

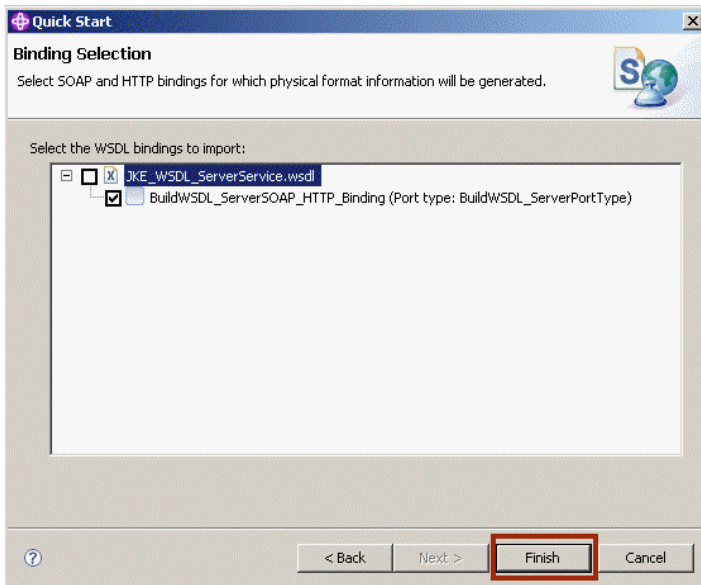
Working set name:

< Back **Next >** Finish Cancel

- __5. Enter **JKE_Server_** for the Message flow project name – **note that the last underscore in the name is intentional so please enter it.** The names are tied together; the other fields will be populated based on this entry. You may change the other names as desired. However, the lab guides assume that the suggested names are used.
- __6. Click **Next**.

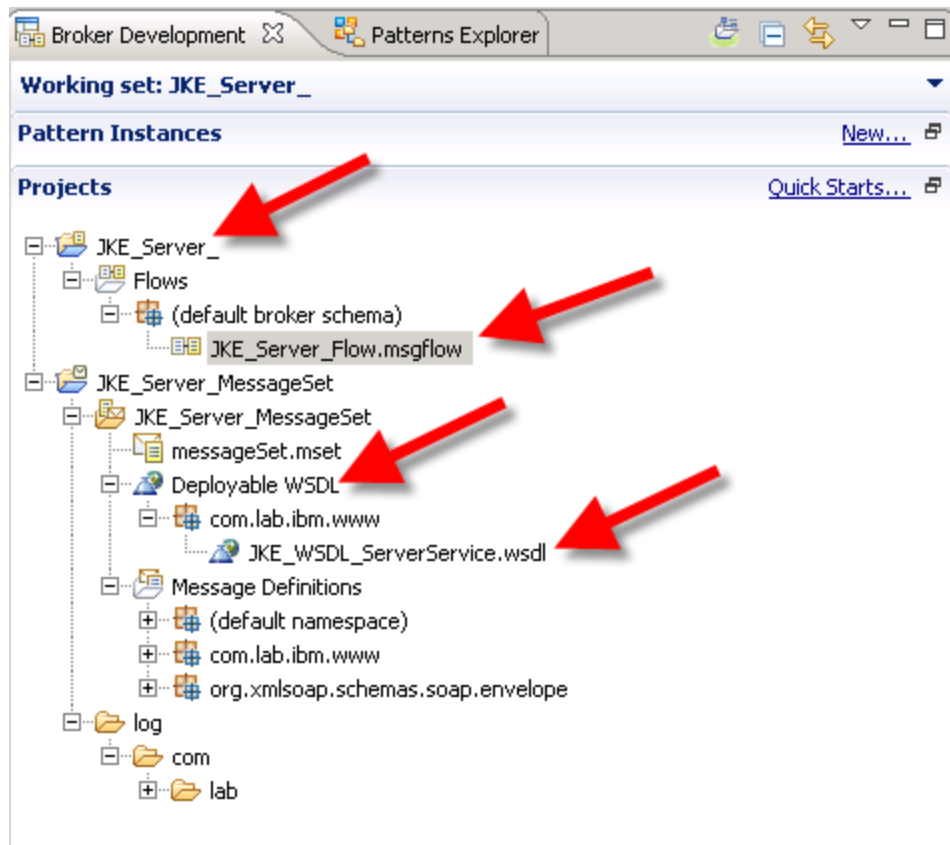


- __7. A list of projects in the workspace is displayed. Expand the **JKE_LabMaterials** folder
- __8. Click on the **JKE_WSDL_Server** folder – *NOT its check box*.
- __9. Click the check box for the **JKE_WSDLServerService.wsdl** on the right side. The two xsd files will be included automatically.
- __10. Click **Next**.

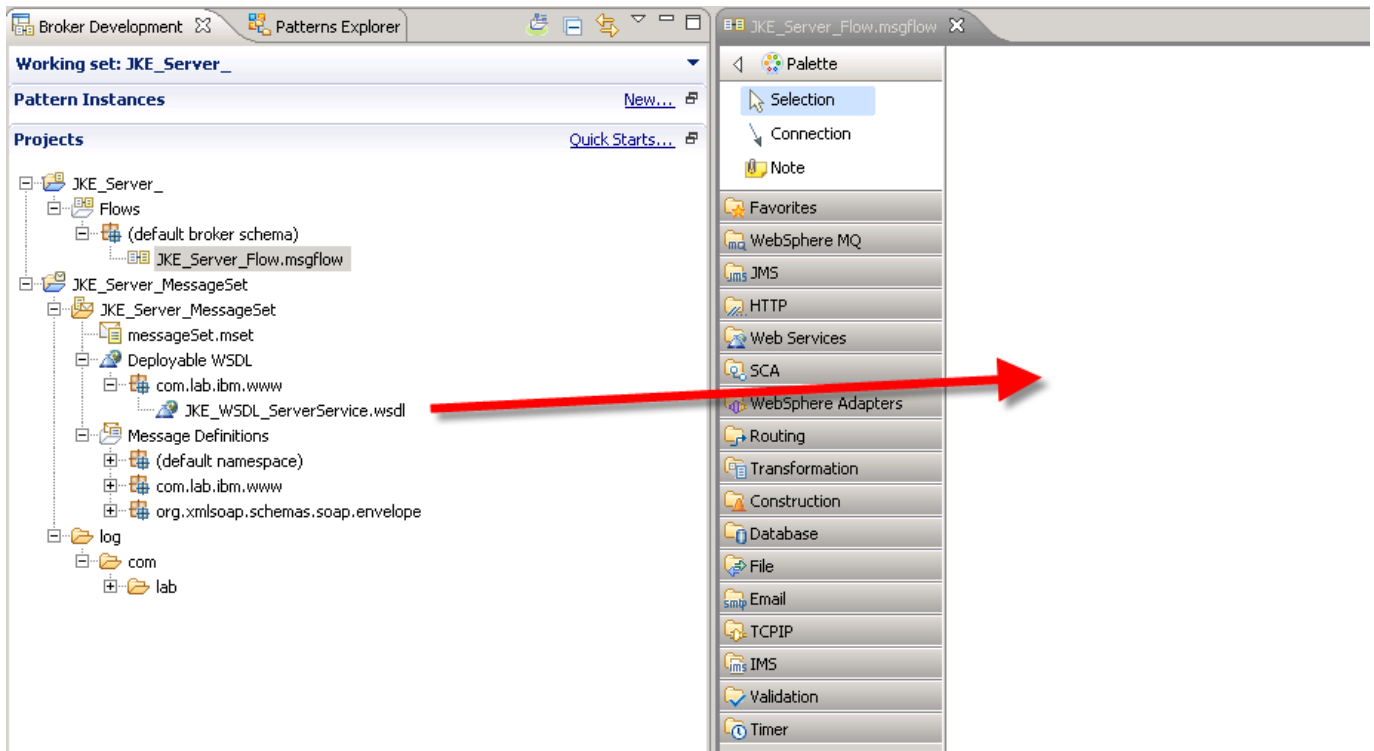


A WSDL may have multiple bindings. In this case there is only one.

- __11. Click **Finish**.

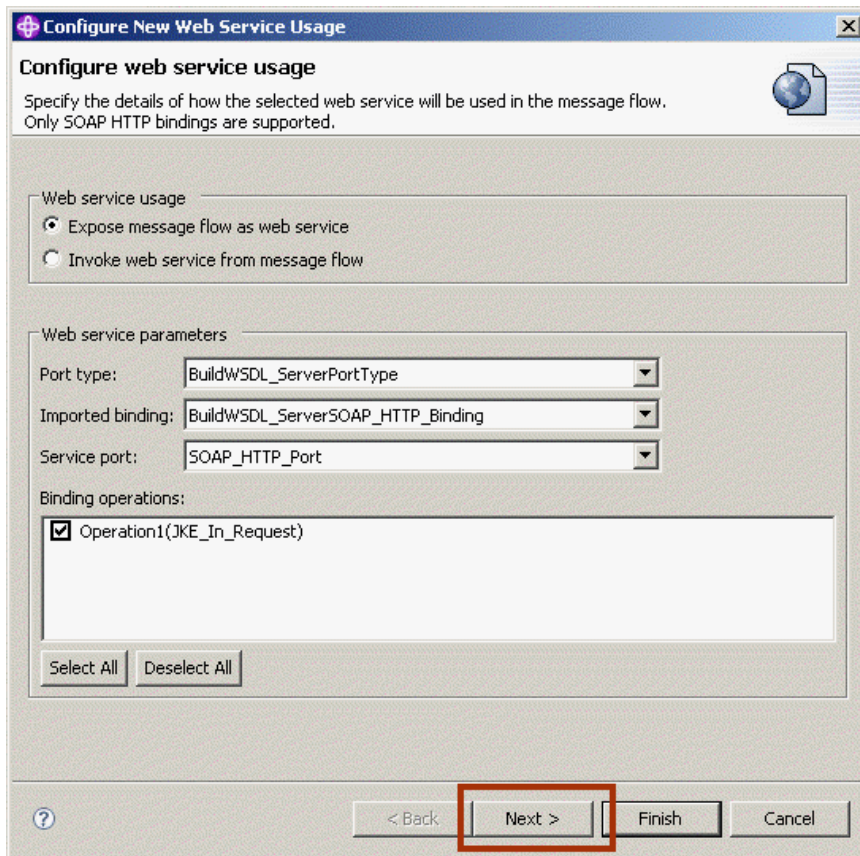


When the wizard finishes several things have been done. A message set project has been created (**JKE_Server_MessageSet**). The message set project contains message definitions for all of the possible inputs to and outputs from the web service that are defined in the WSDL file. A message flow project (**JKE_Server_**) has been created. The new message flow project contains an empty message flow that is opened in the message flow editor. The message set is expanded and the WSDL file, under a folder called **Deployable WSDL**, is ready for a drag and drop operation to start building the message flow.



__12. Highlight the **JKE_WSDL_ServerService.wsdl** file.

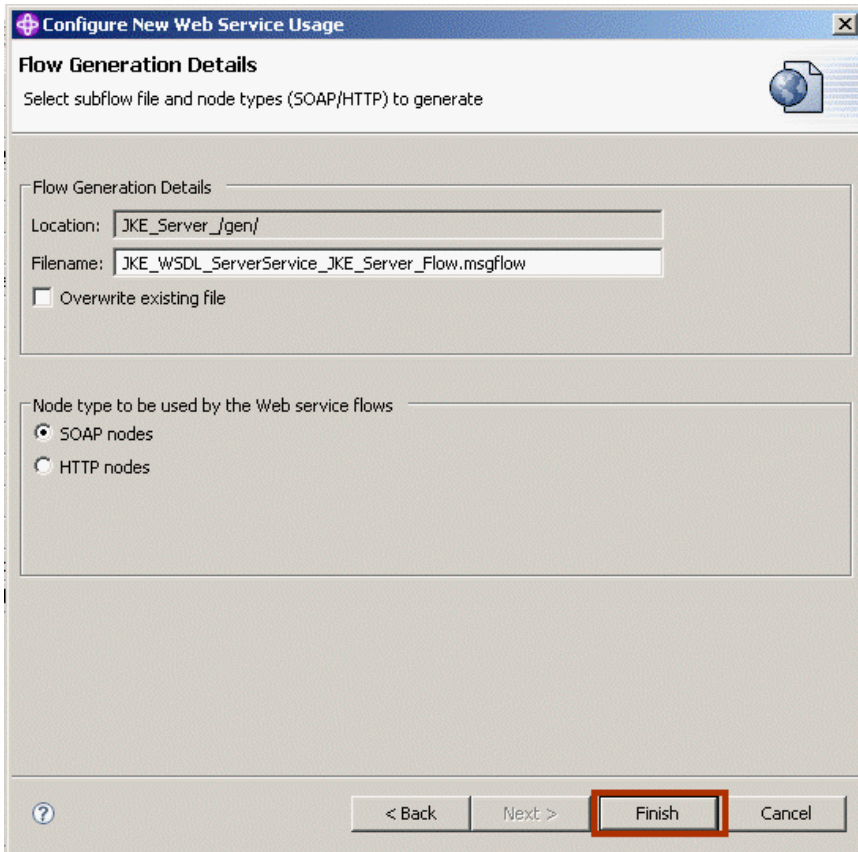
__13. Drag it to the message flow canvas and drop it.



A new wizard is started.

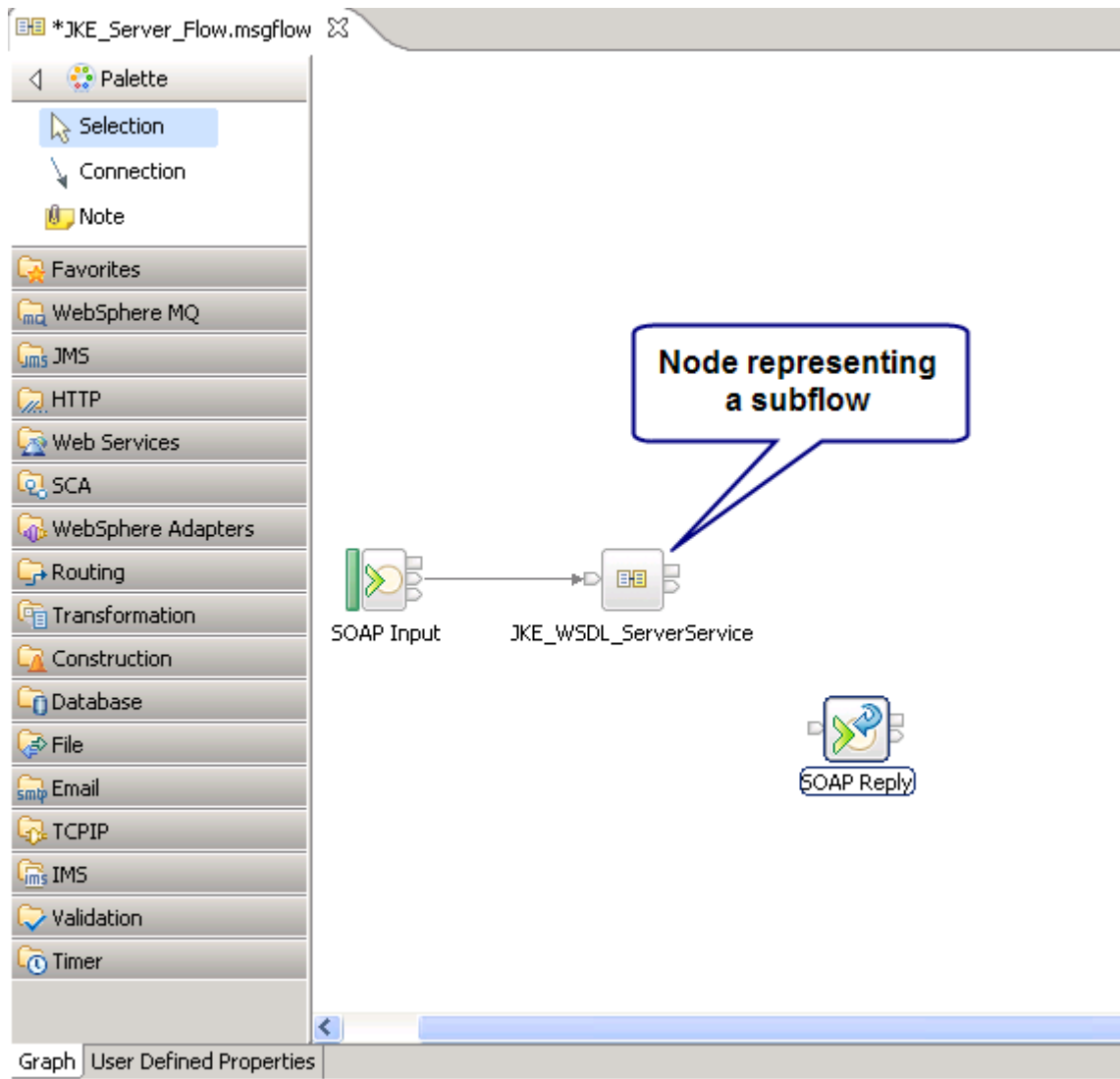
The wizard allows the message flow to be exposed as a web service (the default and the action to take in this case) or to invoke a web service from the message flow. In this case no changes are required.

__14. Click **Next**.

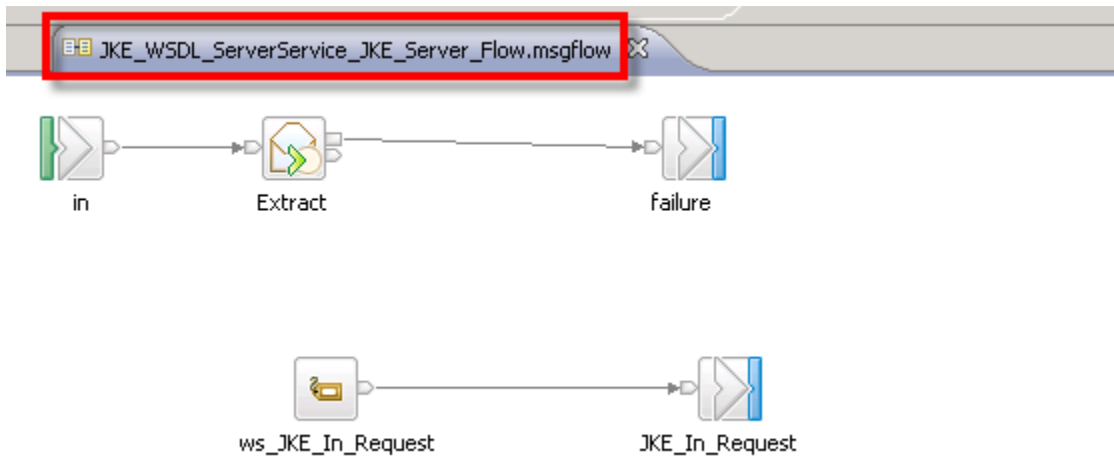


The wizard will build a sub-flow that extracts the Body of the message from the SOAP Envelope. This panel allows you to change the name of the sub-flow. You may also choose between the SOAP nodes and the HTTP nodes. In the lab, you will use the default options.

___15. Click **Finish**.



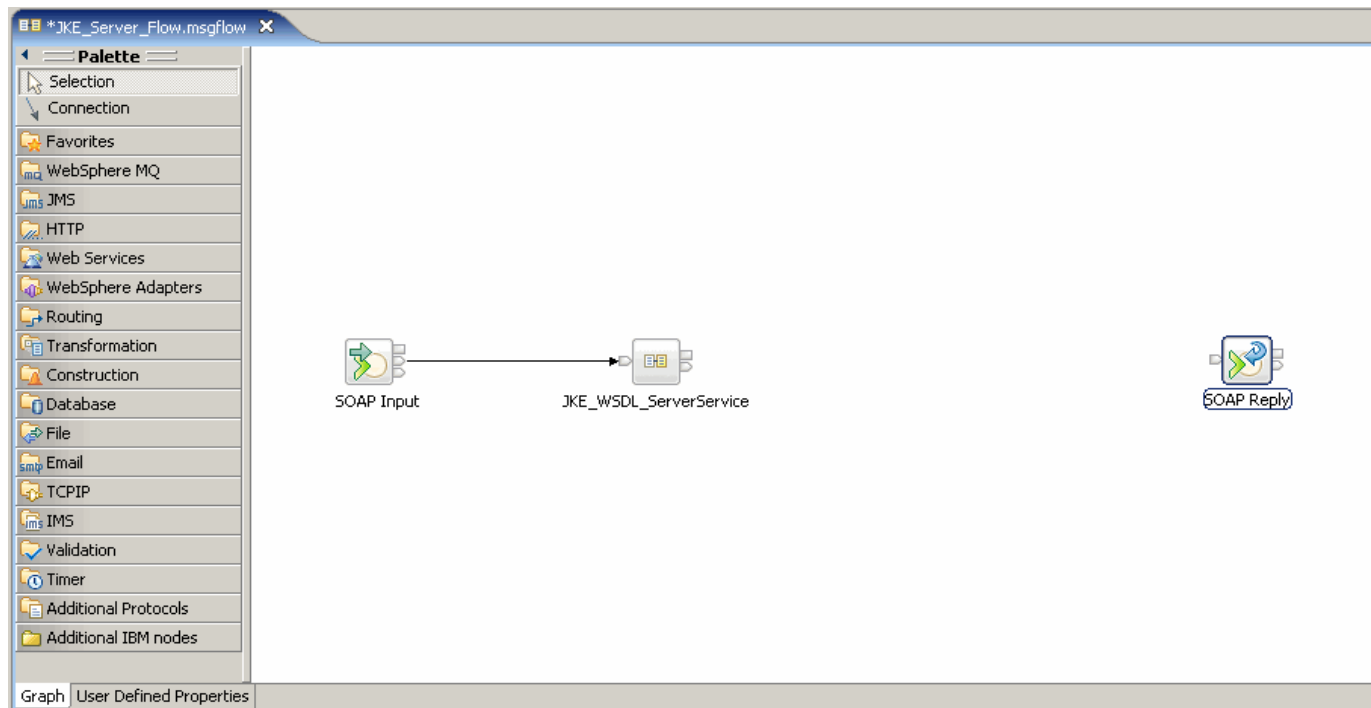
Three nodes are generated by the drag and drop and placed on the canvas. A SOAPInput node is connected to a node representing a sub-flow. The sub flow node is connected to a SOAPReply node.



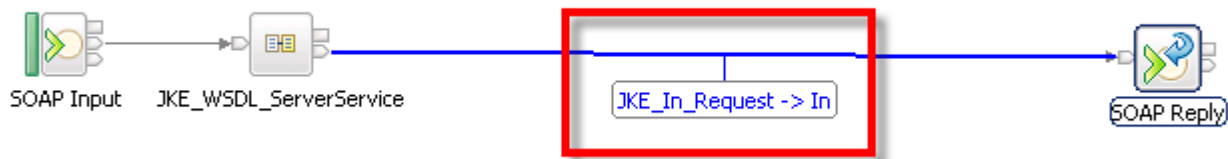
__16. Double-click on the sub-flow node to open the sub-flow.


The SOAPExtract node removes the envelope from the SOAP message. It has a separate output path for each binding in the WSDL (there is only one in this case). The Extract node uses a Route To Label option to drive the output path for the appropriate binding.

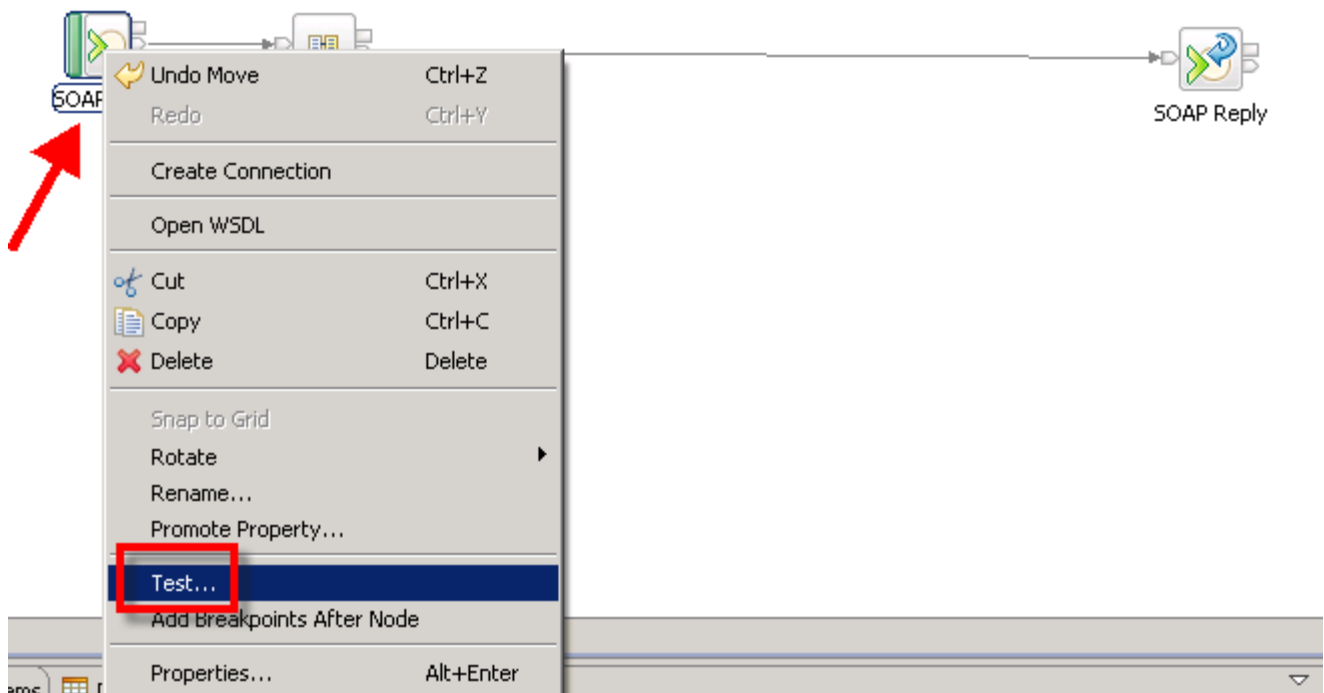
__17. **Close** the subflow.



__18. Arrange the nodes similar to the example above, moving the SOAP Reply node to the right side to provide additional space between the nodes...

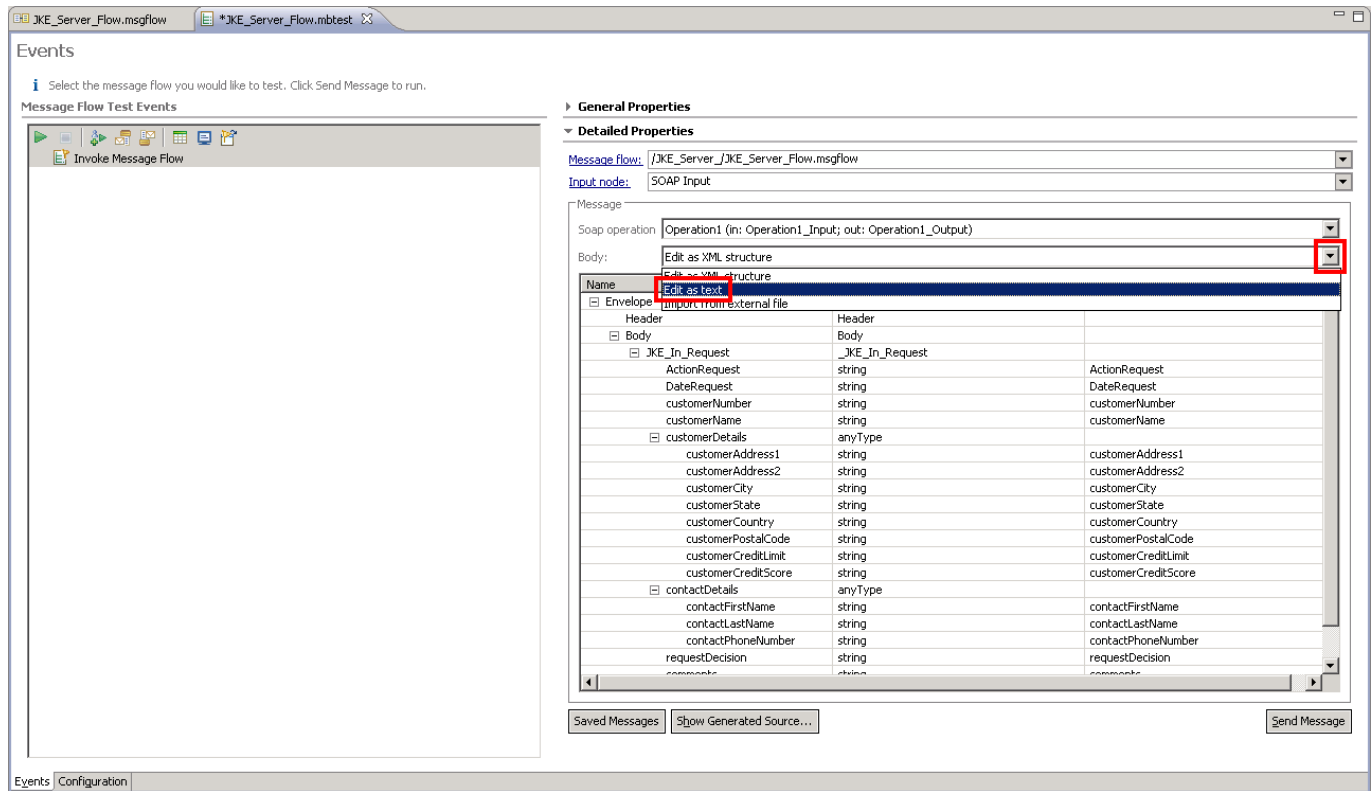


- __19. Wire the **JKE_In_Request** terminal to the **In** terminal of the SOAP Reply node. **Note:** if you place your mouse pointer on a connector, a bubble will pop up (as shown above) that will allow you to verify that you have properly connected the two nodes involved.
- __20.  Save the message flow <Ctrl-S>.

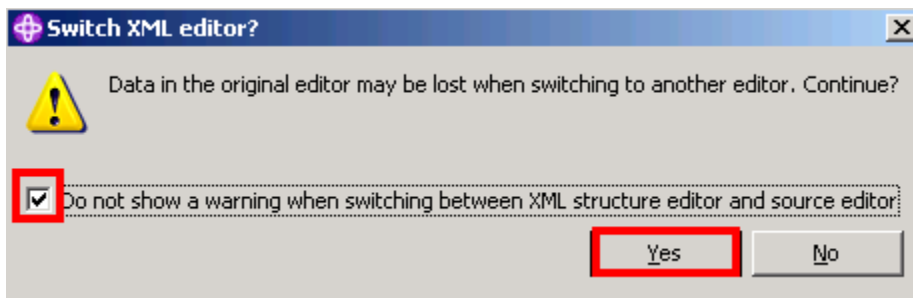


The Test Client can be used to test flows that begin with MQInput, HTTPInput or SOAPInput nodes. In our first four labs we handled all of the tasks that are required to test a message flow ourselves. The Test Client will automate much of that process for us. It will create and populate a BAR file; deploy the BAR file to the Broker; submit a test message; monitor any output nodes for results and display the results.

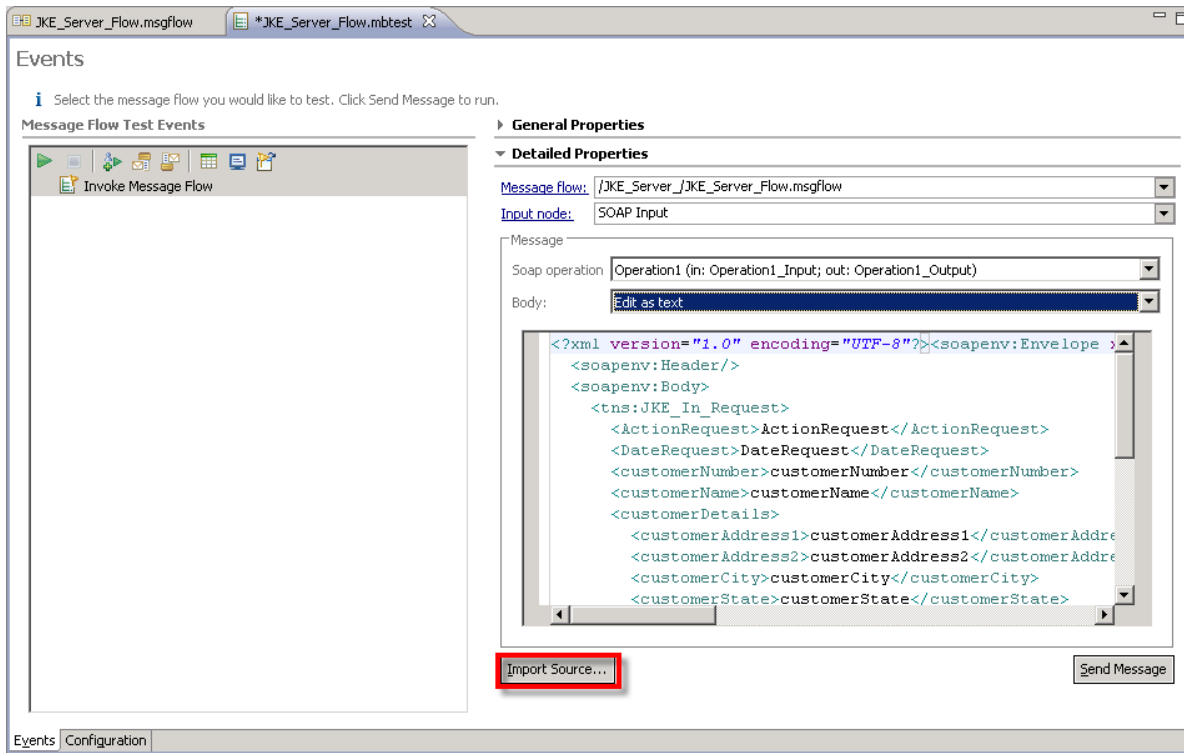
- __21. Right click on the **SOAP Input** node.
- __22. Select **Test** from the menu. This starts the Test Client.



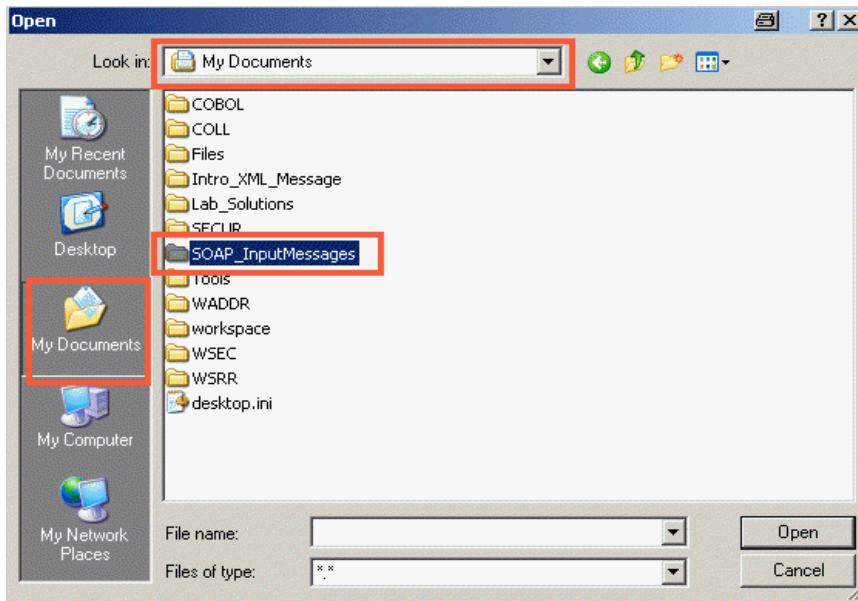
__23. Use the **Body** pull-down menu to select the **Edit as text** option. The buttons at the bottom will change.



__24. A tip panel is displayed. You may want to click the check box before you click **Yes**.

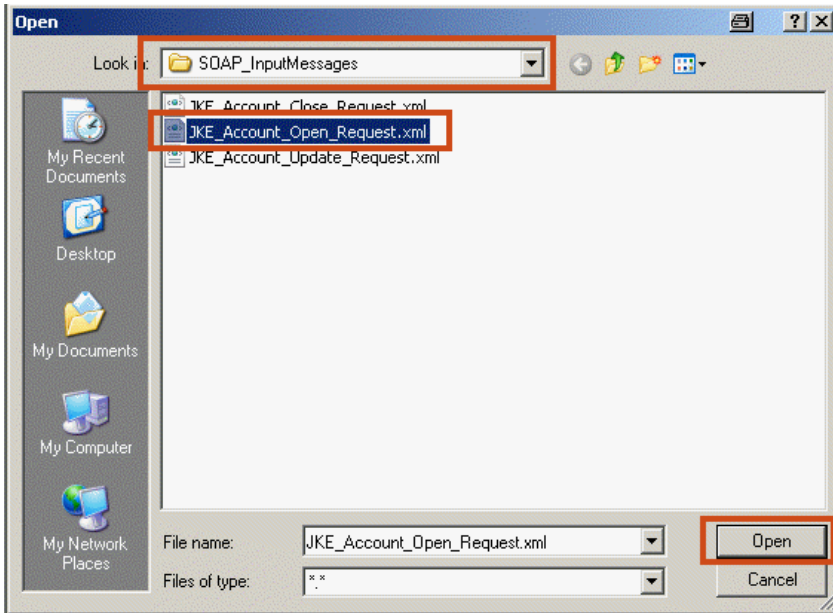


25. Click the **Import Source** button.



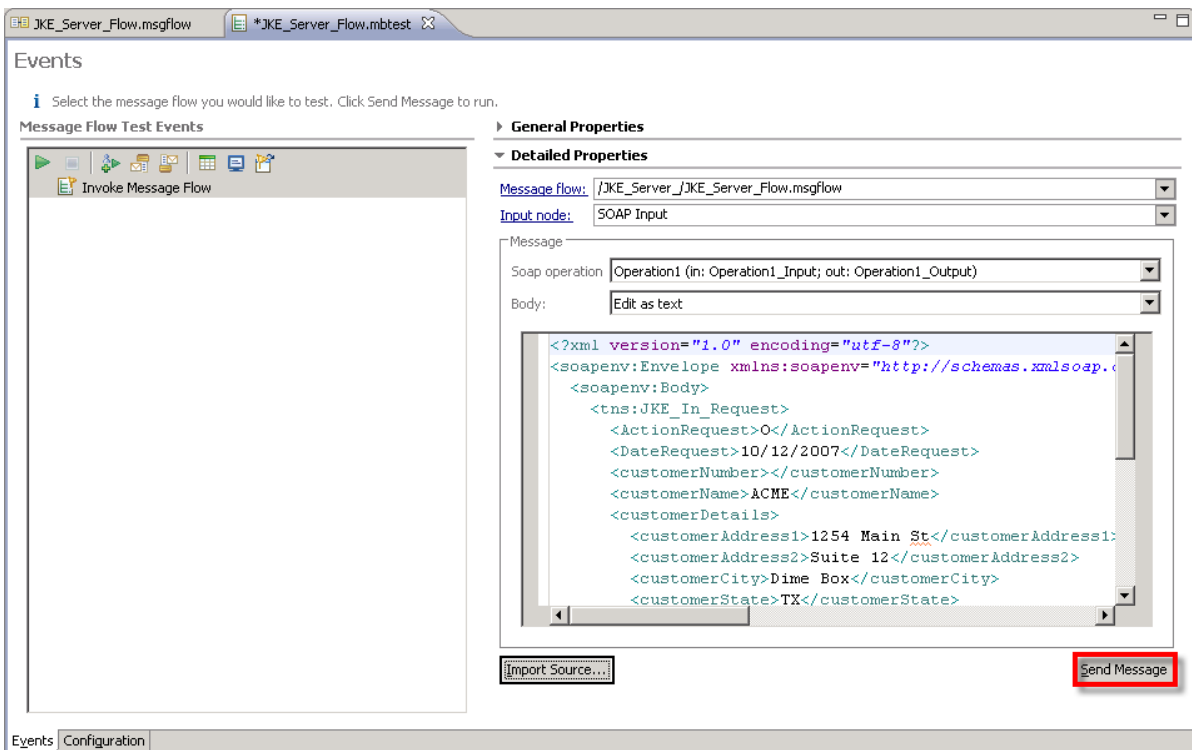
26. If necessary, navigate to **My Documents**, which points to C:\student.

27. Select **SOAP_InputMessages**.

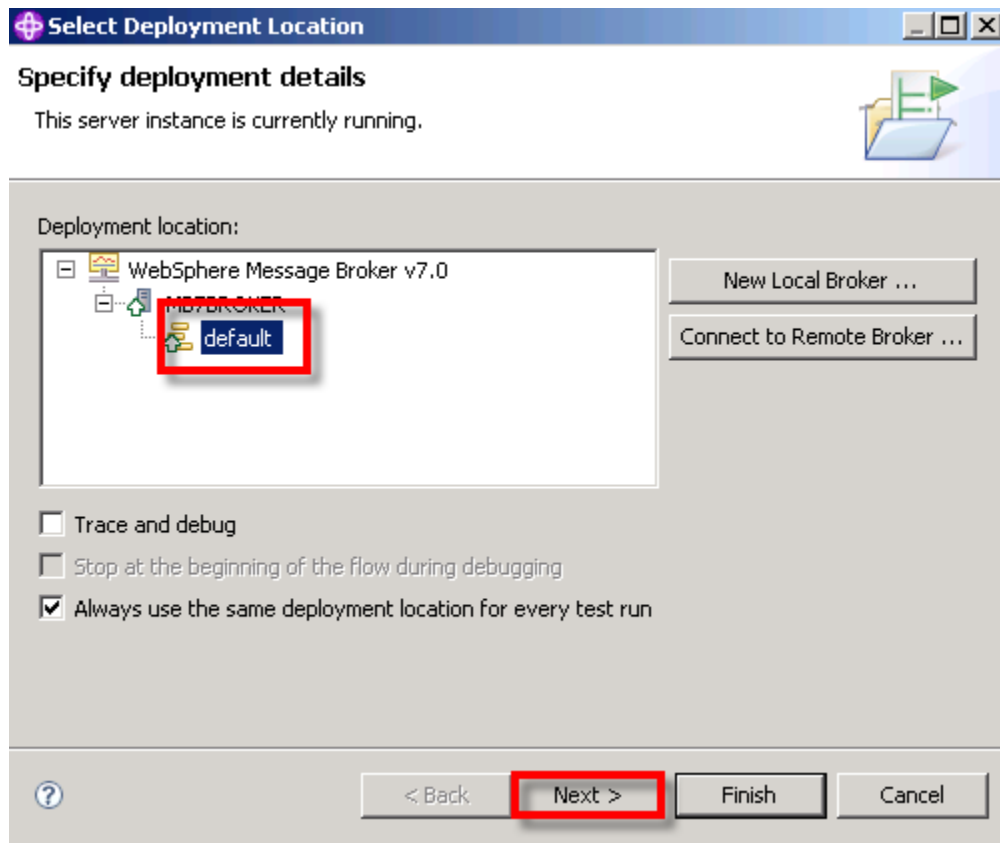


__28. Highlight the **JKE_Account_Open_Request.xml** file.

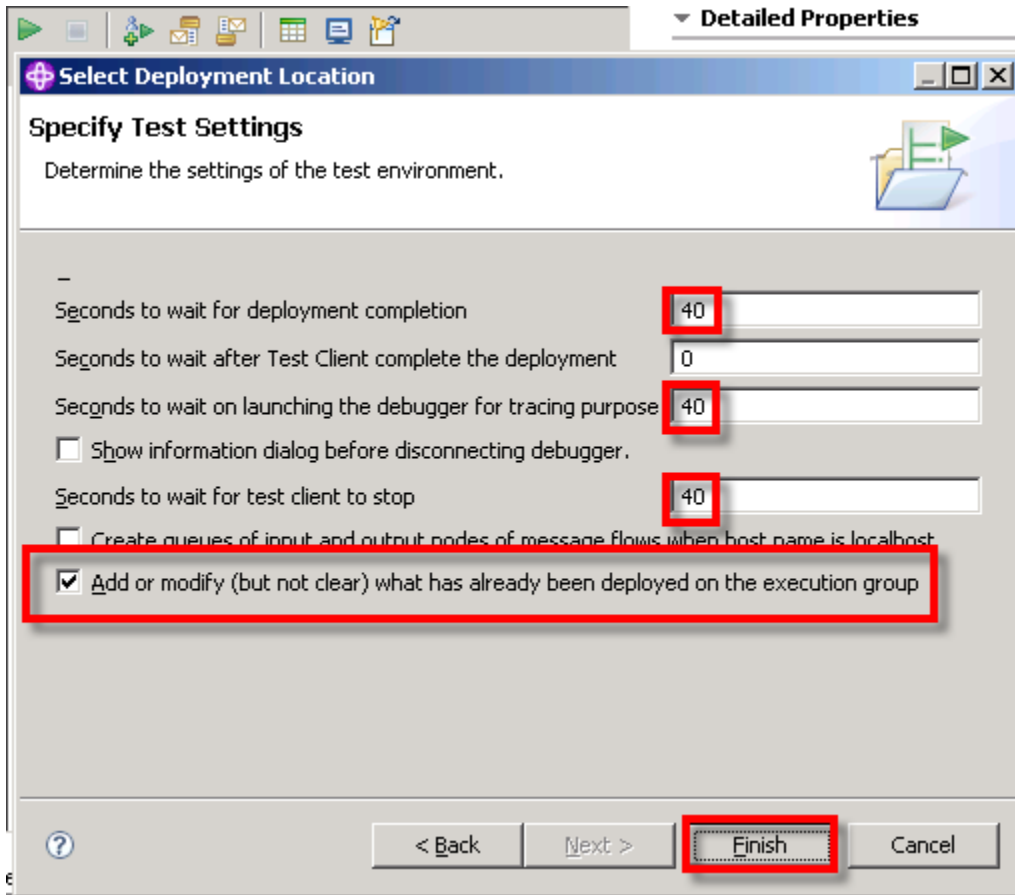
__29. Click **Open**.



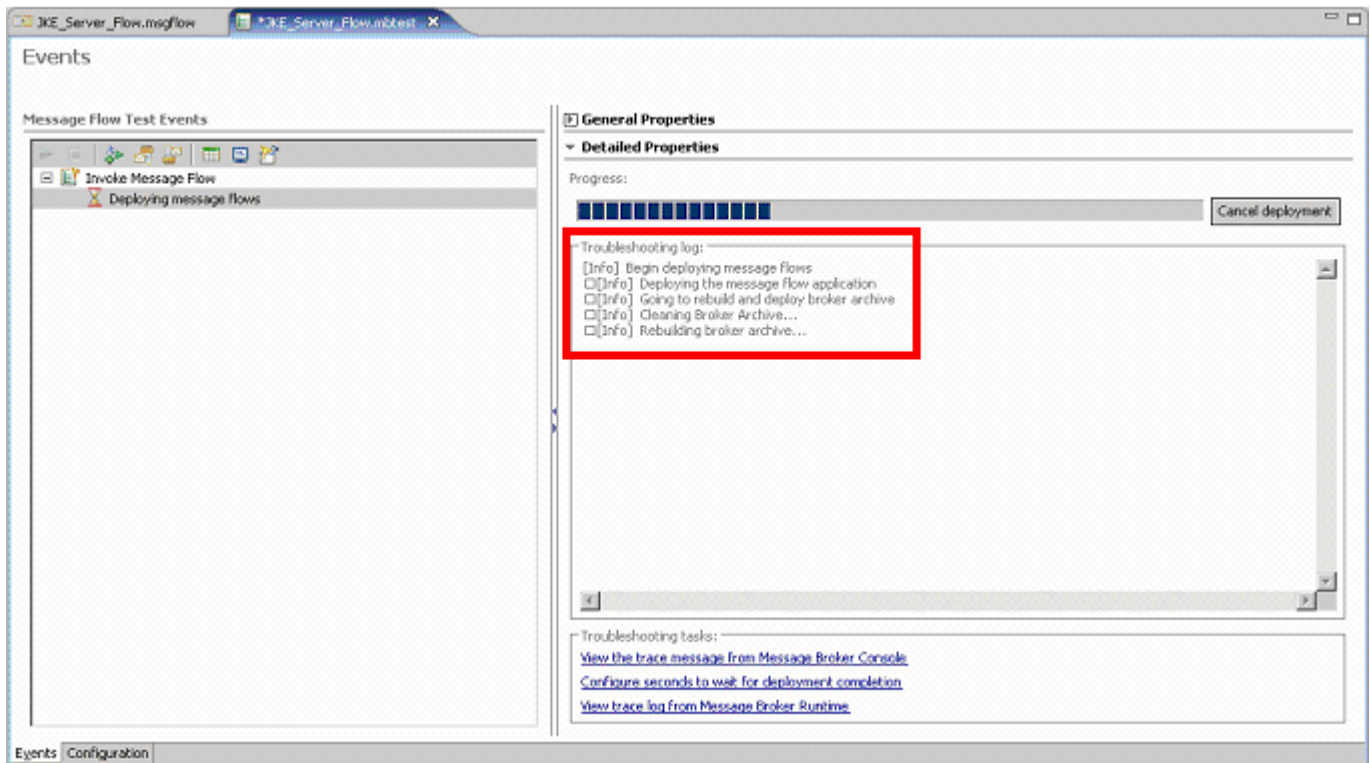
__30. Click on the **Send Message** button.



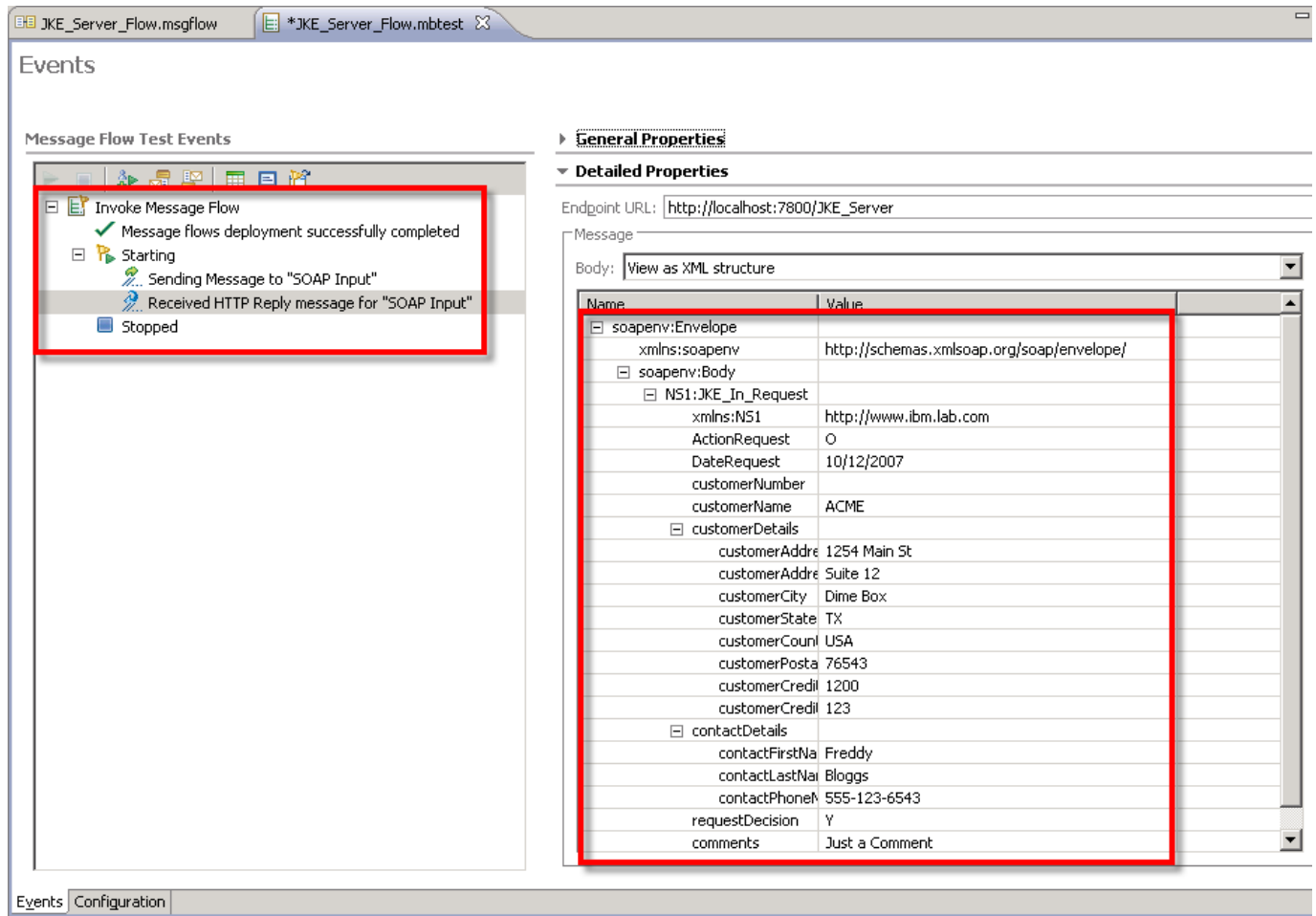
- __31. Highlight the **default** execution group.
- __32. Click **Next** (this time only).



- __33. Click the **Add or modify** ... check box as shown if not already checked.
- __34. Verify that you have a value of 40 in each of the non-zero fields as indicated above.
- __35. Click **Finish**.



The Test Client is deploying the message flow and associated artifacts. The progress of these actions is detailed in the Troubleshooting log.



__36. When the response is received, it is displayed and the test stops.

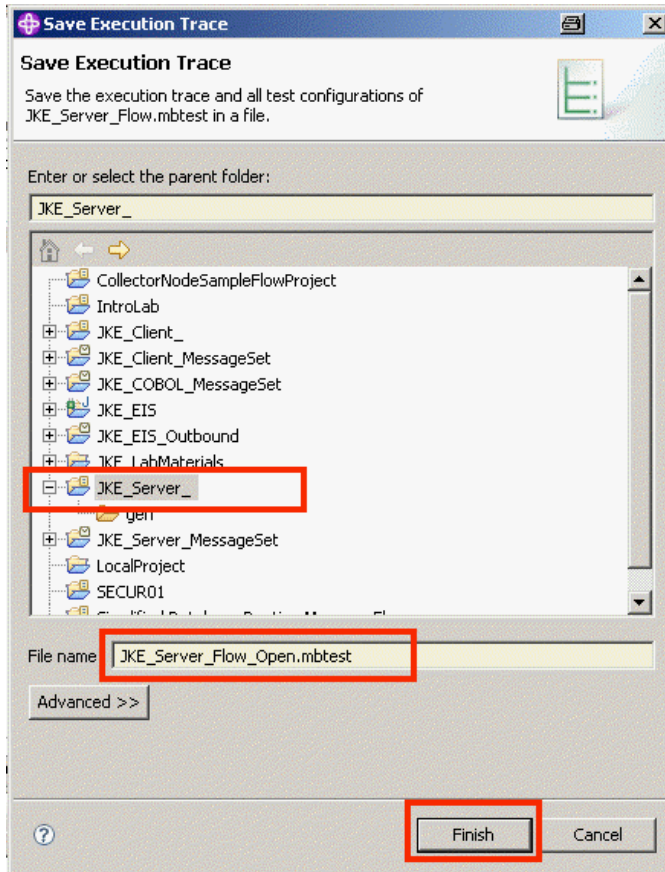
The screenshot displays the IBM Software interface for testing a message flow. The 'Events' pane on the left shows a sequence of events: 'Invoke Message Flow', 'Message flows deployment successfully completed', 'Starting', 'Sending Message to "SOAP Input"', 'Received HTTP Reply message for "SOAP Input"', and 'Stopped'. The 'Detailed Properties' pane on the right shows the 'Endpoint URL' as 'http://localhost:7800/JKE_Server' and the 'Message' body as 'View as XML structure'. Below this, a table lists the message structure:

Name	Value
soapenv:Envelope	
xmlns:soapenv	http://schemas.xmlsoap.org/soap/envelope/
soapenv:Body	
NS1:JKE_In_Request	
xmlns:NS1	http://www.ibm.lab.com
ActionRequest	O
DateRequest	10/12/2007
customerNumber	
customerName	ACME
customerDetails	
customerAddress	1254 Main St
customerAddressSuite	Suite 12
customerCity	Dime Box
customerState	TX
customerCountry	USA
customerPostalCode	76543
customerCreditCardNumber	1200
customerCreditCardExpirationDate	123
contactDetails	

A 'Save Resource' dialog box is overlaid on the interface, with the text: "'JKE_Server_Flow.mbtest' has been modified. Save changes?'. The 'Yes' button is highlighted with a red box.

__37. Close the Test Client.

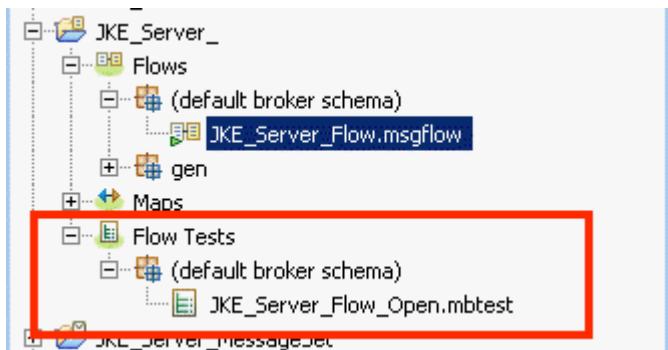
__38. Select **Yes** to save the test. This will enable you to repeat this same test in later labs without needing to rebuild it.



- __39. Highlight the **JKE_Server_** project.
- __40. Add “**_Open**” to the File name as shown above.
- __41. Click **Finish** to save the test client configuration.

This Test Client configuration will be used in later labs. The configuration settings and the test messages will be saved.

The ability to save multiple configurations and messages can be used to build a “test suite”. This could be used for regression testing in the future, for example.



The **JKE_Server_Flow_Open.mbttest** is added to a new folder, Flow Tests, in the **JKE_Server_** project.

This is the end of Lab 5.

Lab 6 Building the JKE Server – Web Services Client

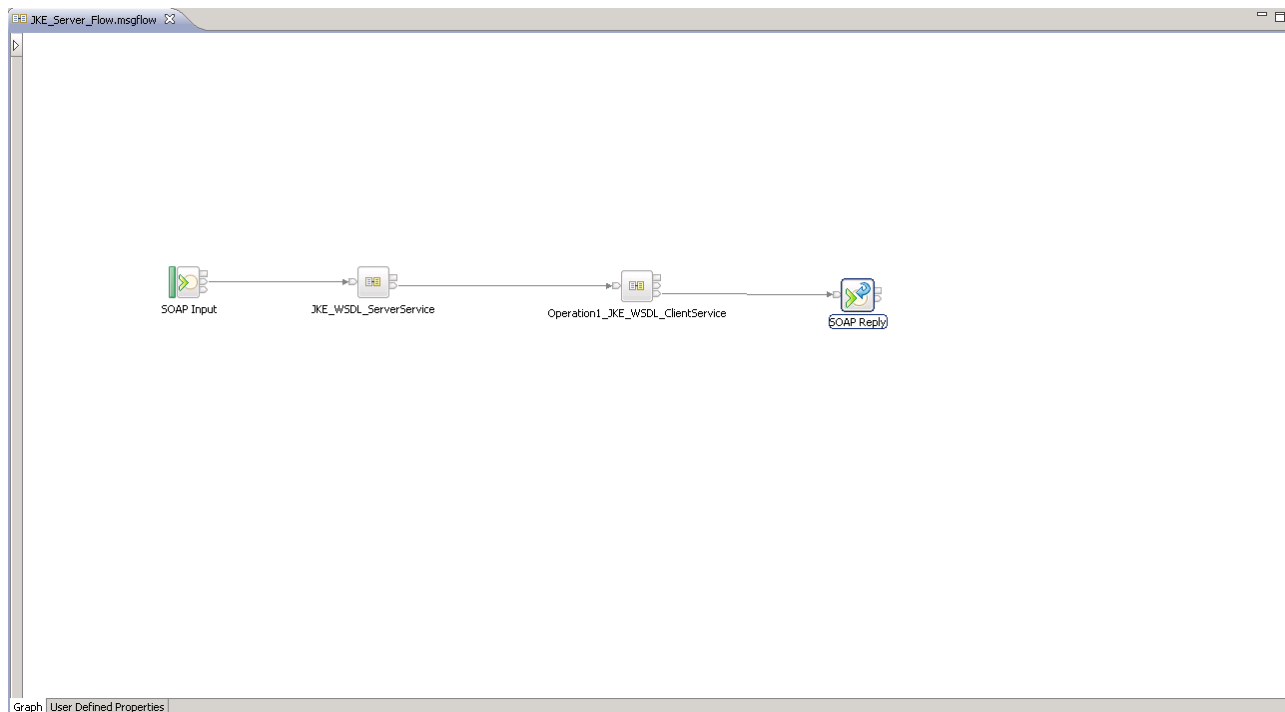
6.1 Overview

In Lab 6, you will add a Web service request to the message flow. This will invoke another Web service (JKE_Client) that is using a separate WSDL definition and Uniform Resource Identifier (URI). However the message structure is identical. This was done to simplify the message flow that you are building. The step by step details of how the WSDL definitions were created as well as how the JKE_Client message flow was built are found in Appendix A and Appendix B.

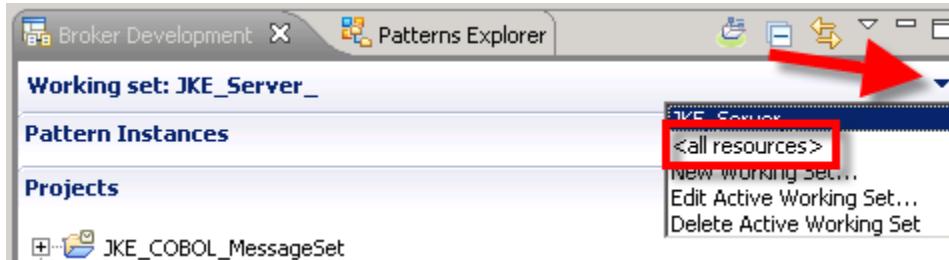
You will add this request to the message flow by doing a drag and drop of the WSDL file that is associated with the JKE_Client service. The JKE_Client Web Service includes the use of a WebSphere enterprise information systems (EIS) adapter. In addition to the SAP, PeopleSoft and Siebel adapters that are included with WebSphere Message Broker, a test adapter called TwineBall is provided. The configuration and use of this adapter is very similar to the other adapters.

The lab is divided into two parts. In the first part you will deploy the JKE_Client message flow and its associated artifacts. A JKE_Client bar file is provided for this purpose.

In the second part, you will add the Web Service request and test the message flow.

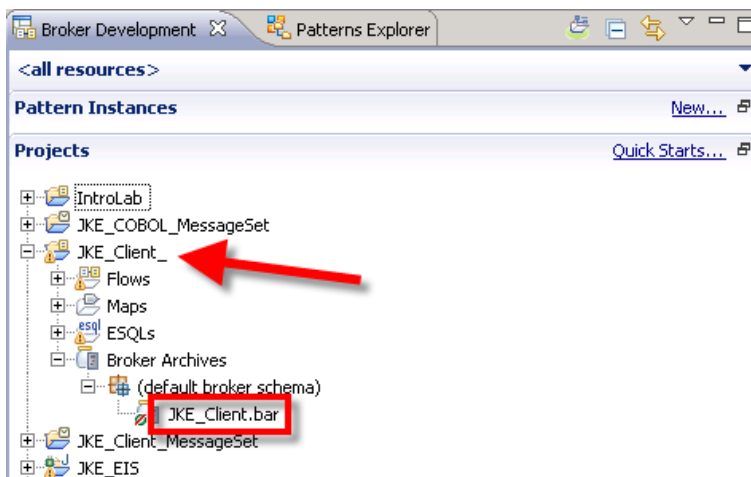


6.2 Deploying the JKE_Client – Web Service Client Message Flow

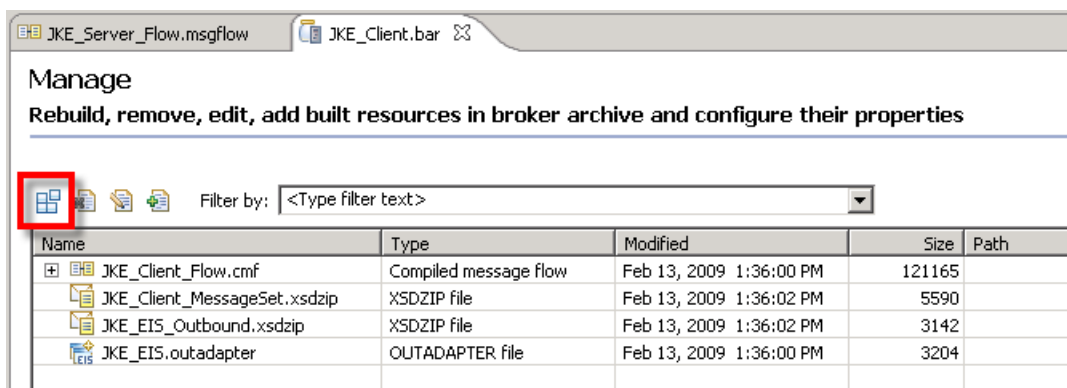


Before testing the message flow, you need to deploy the JKE_Client and its artifacts. A JKE_Client.bar file has been provided for this purpose. The broker archive file is in the **JKE_Client_** project.

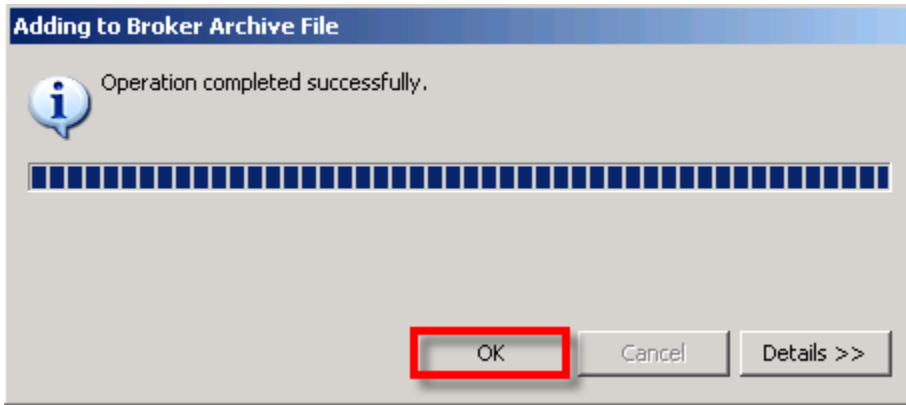
- __1. Select the small triangle opposite the Working set tab.
- __2. Select **<all resources>** from the menu.




- __3. Expand **JKE_Client_ -> Broker Archives -> (default broker schema)**.
- __4. Double-click on the **JKE_Client.bar** broker archive. The archive editor should open.

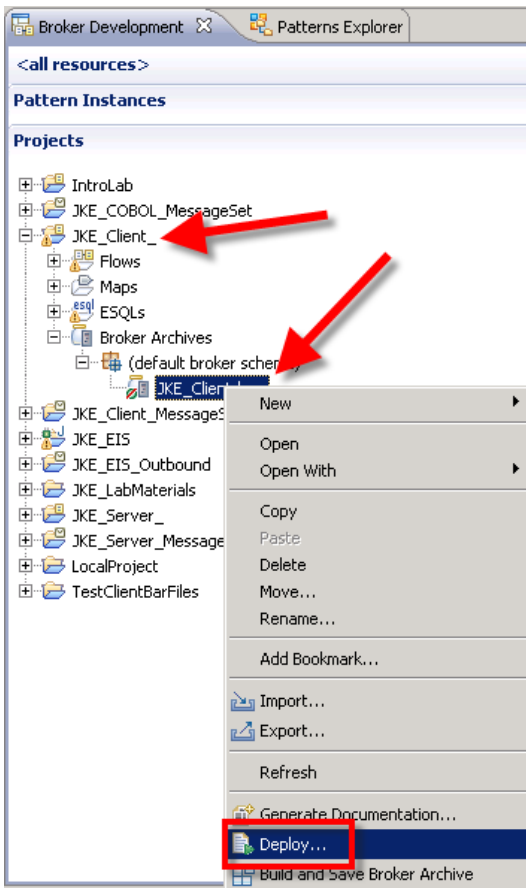


- __5. In the broker archive editor press the **Build** icon.



__6. Press the **OK** button to close the acknowledgement.

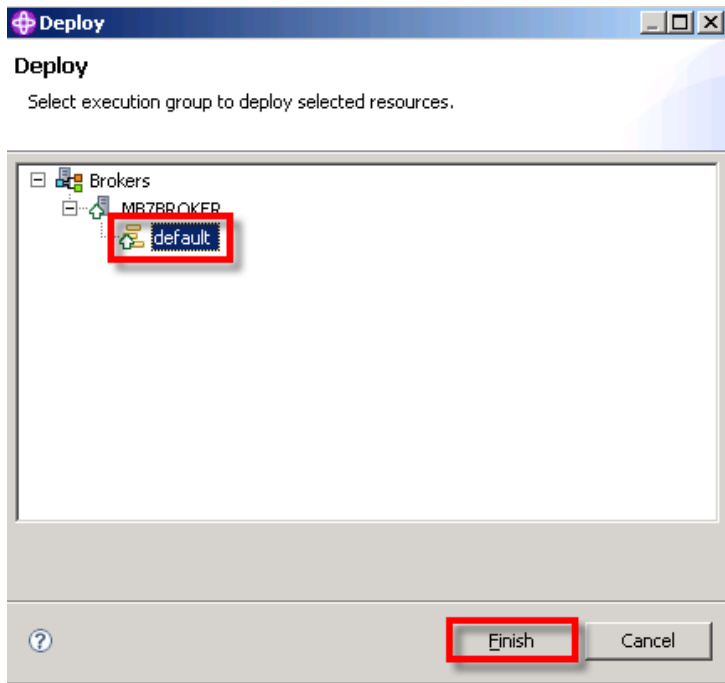
__7.  When finished, **save** the bar file.



__8. Select the **JKE_Client.bar** broker archive.

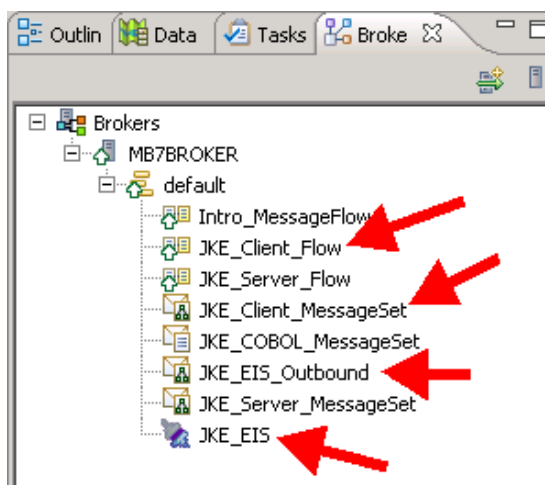
__9. Press the right mouse button.

__10. Select **Deploy** from the menu.



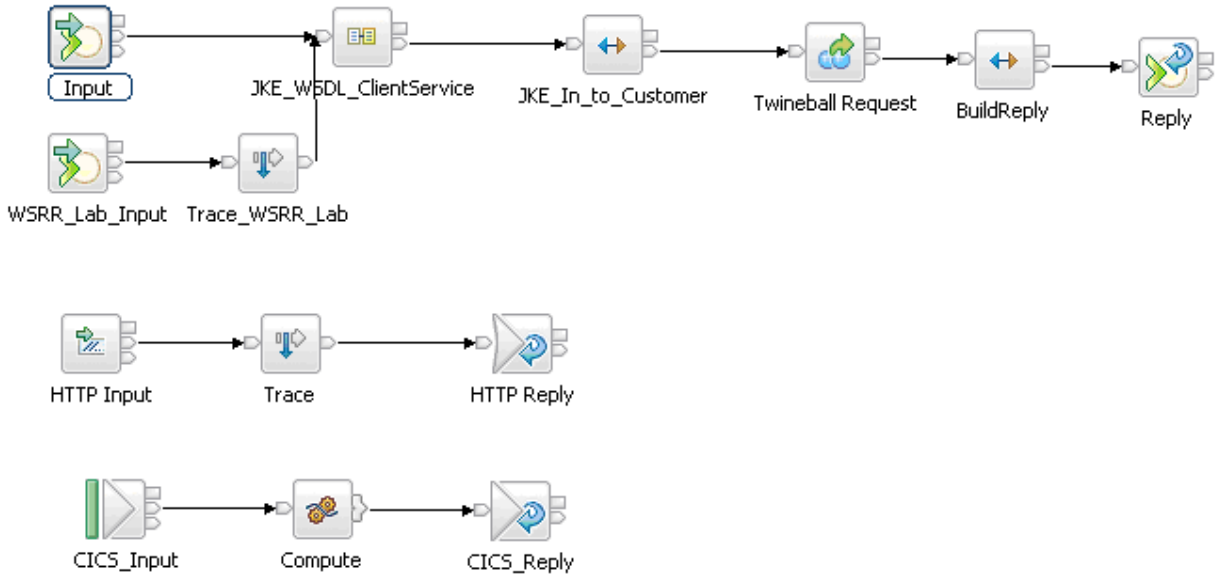
__11. Select the **default** execution group.

__12. Click **Finish**.



When the deploy operation is complete, additional items will be shown on the **Brokers** tab in the lower left pane. These include the JKE_Client_Flow, JKE_Client_MessageSet, JKE_EIS (the adapter interface) and JKE_EIS_Outbound (the adapter message set and connector).

__13. Close the bar file editor.



This is the JKE_Client message flow that is described in Appendix B. Note that the **top sequence of nodes** implements the **Account Open Web service**. The bottom two sequences of nodes are for other purposes that will be described in later labs.

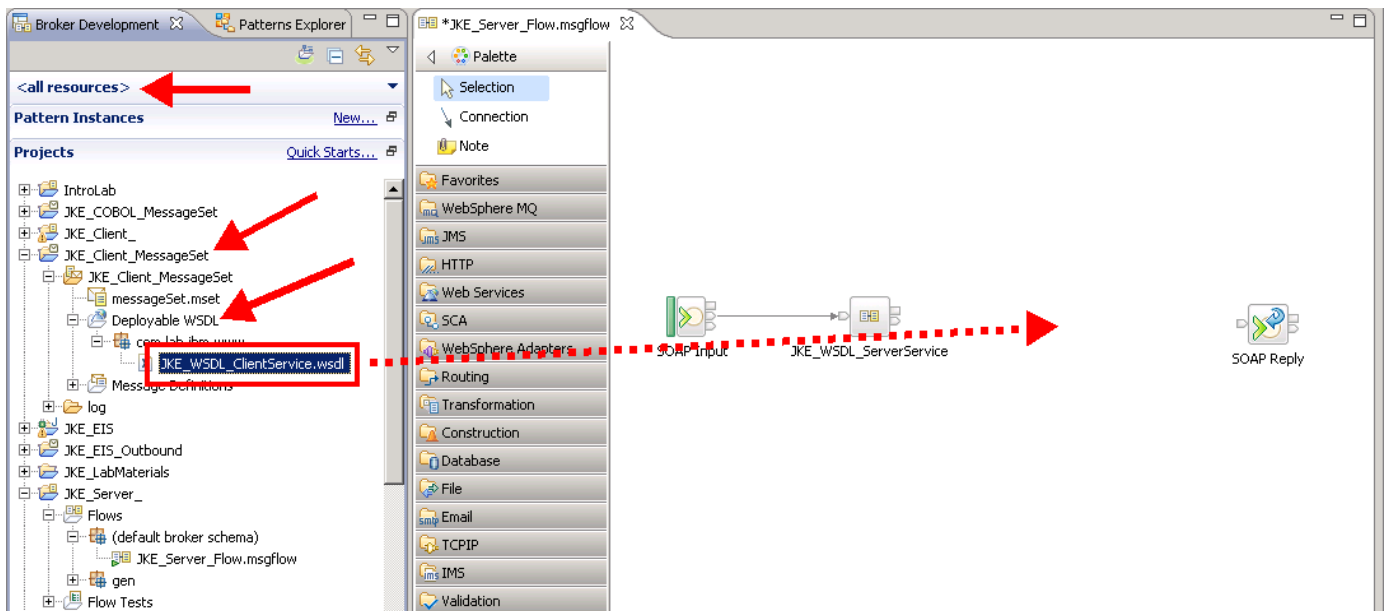
6.3 Adding a Web Service Request to the JKE_Server Message Flow



__1. If the **JKE_Server_Flow.msgflow** is not in focus, click its tab and bring it into focus.

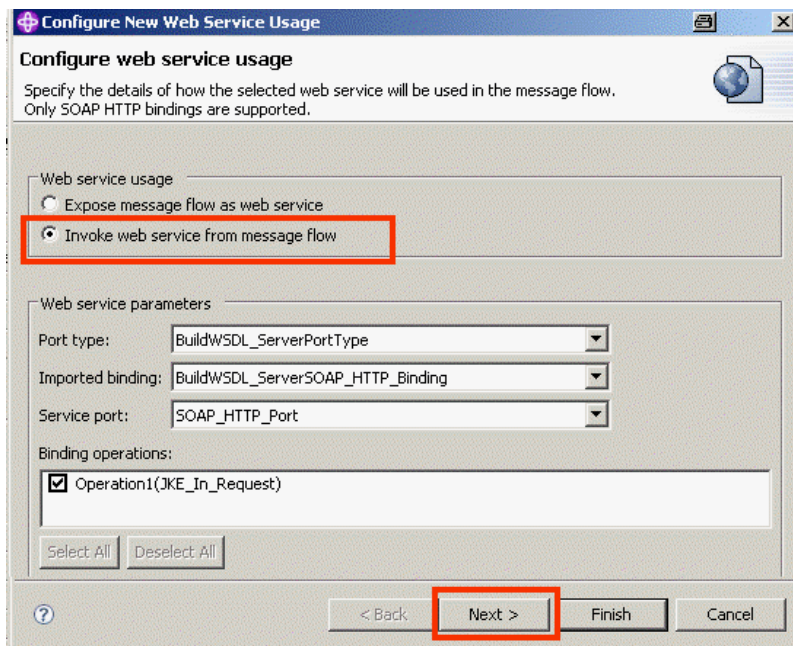
The connector between the **JKE_WSDL_ServerService** node and the **SOAP Reply** node will now be removed.

- __2. Click on the connector to select it.
- __3. Press the right mouse button.
- __4. Select **Delete** from the menu.

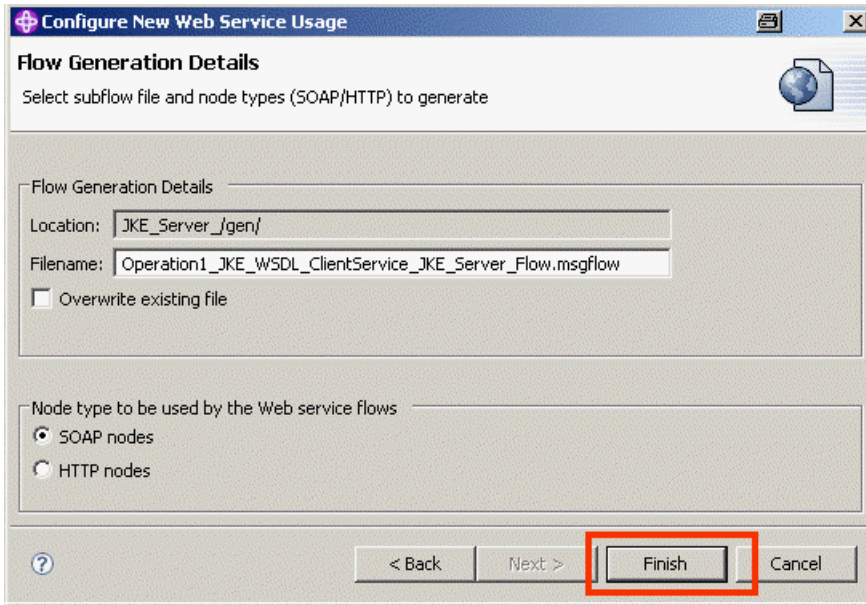


- ___5. Expand the **JKE_Client_MessageSet** folders until you can see the **JKE_WSDL_ClientService** WSDL file. **Make sure you are viewing the Client WSDL and not the Server WSDL!!**
- ___6. Drag and drop the **JKE_WSDL_ClientService.wsdl** file onto the canvas as shown.

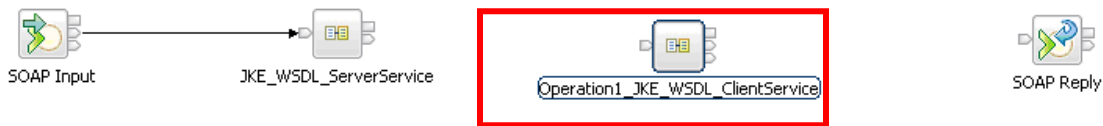
This will start the same wizard as used in the prior lab. This time a Web Service request will be created.



- ___7. Select the **Invoke web service from message flow** radio button.
- ___8. Click **Next**.

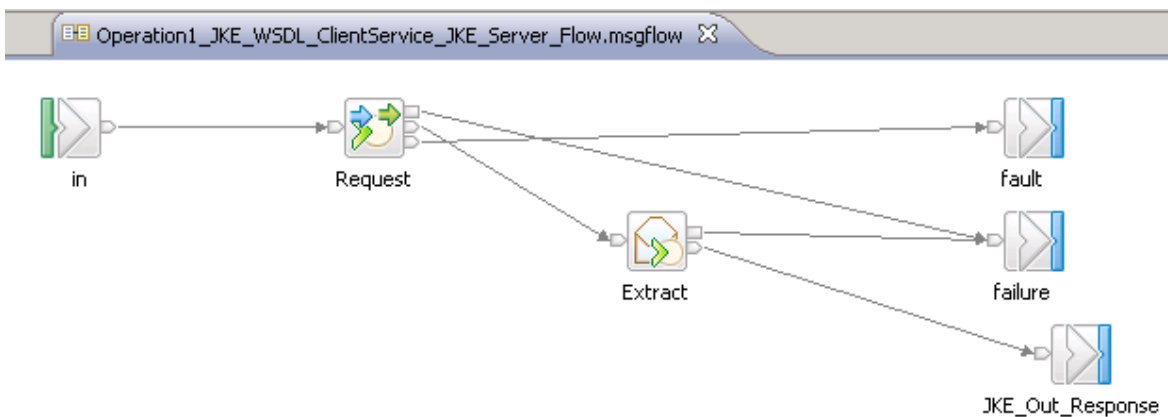


__9. Click **Finish**.

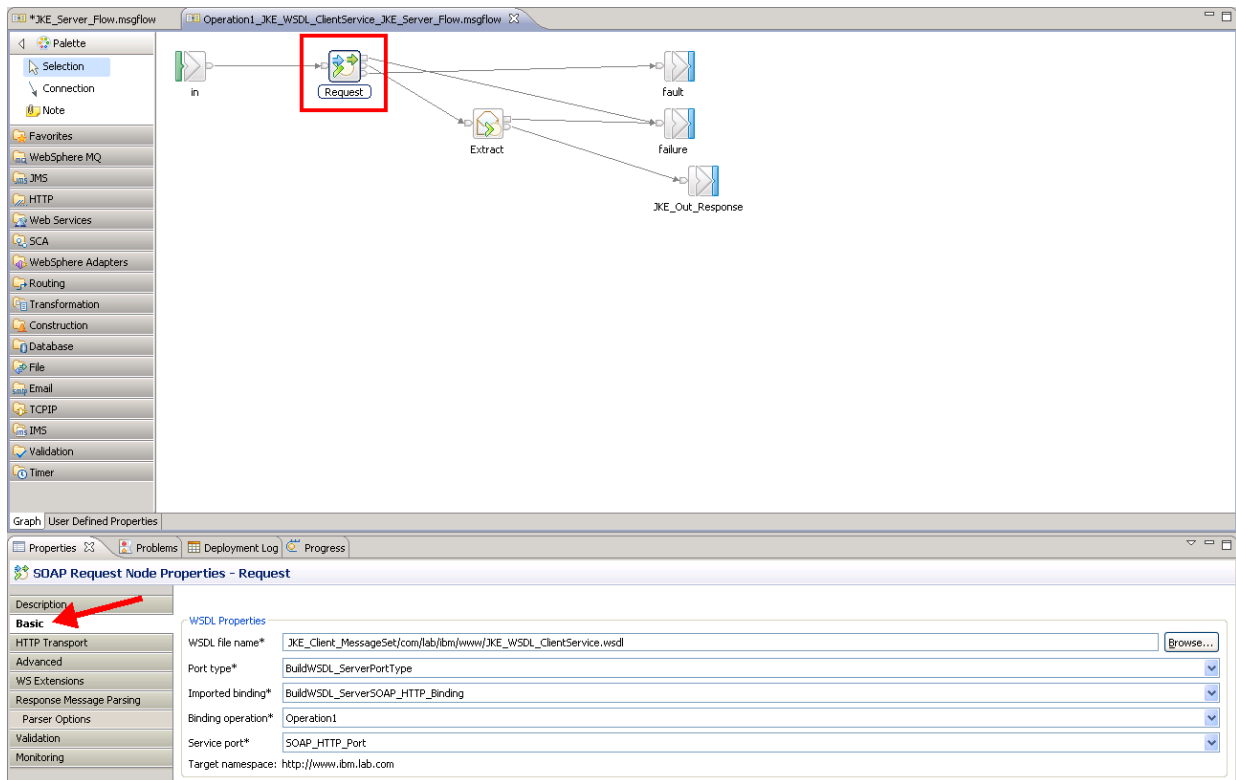


The wizard creates a sub-flow to invoke the Web service and return the response.

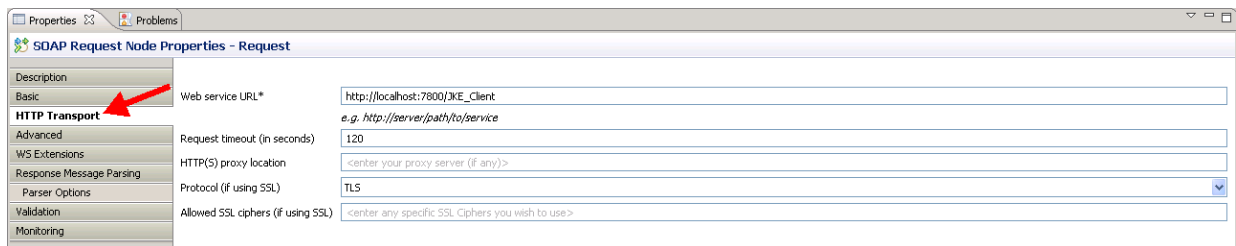
__10. **Double-click** on the sub-flow node to open the sub-flow.



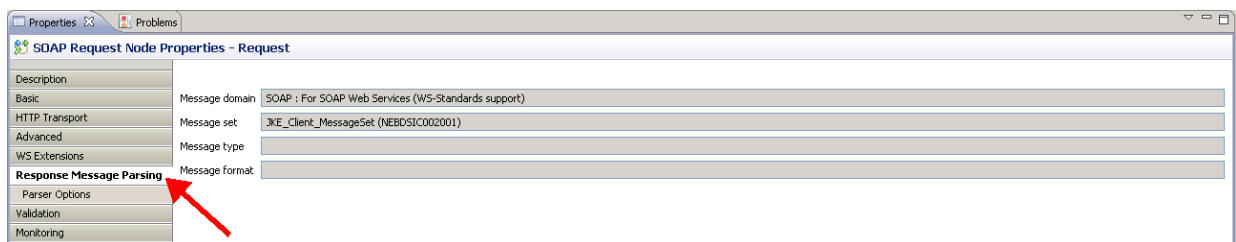
The sub-flow invokes the Web service with a SOAPRequest node. When a response is received it is passed to a SOAPExtract to remove the envelope from the SOAP message which is then passed to the message flow via the JKE_Out_Response path. SOAP Faults or other failures are also handled.



- __11. Click on the **Request** node and review the Properties. These are set automatically based on the WSDL.

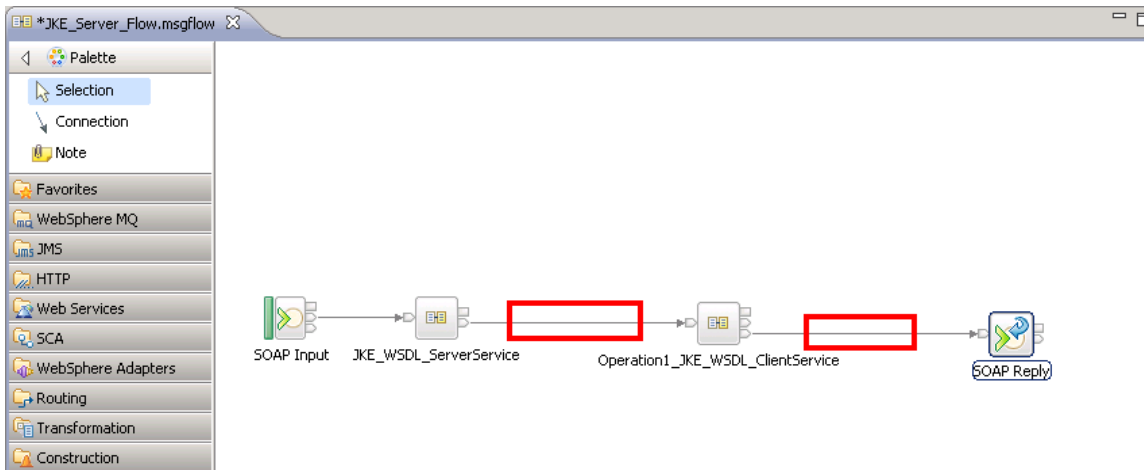



- __12. Click on the **HTTP Transport** tab. This contains the URI used to invoke the Web service

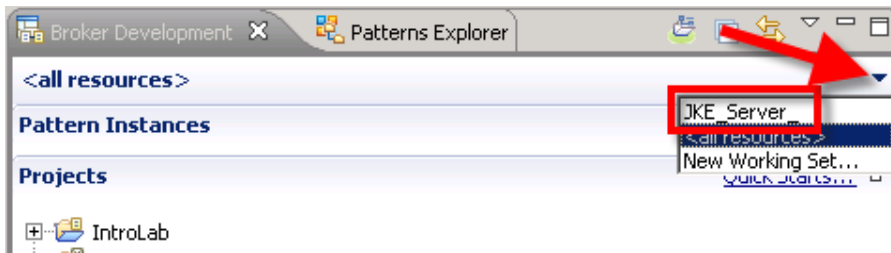


- __13. Click on the **Response Message Parsing** tab. This identifies the Parser. A specific message definition is built automatically to process the SOAP message using the XMLNSC parser.

- __14. After examining the properties **close** the sub flow editor session.

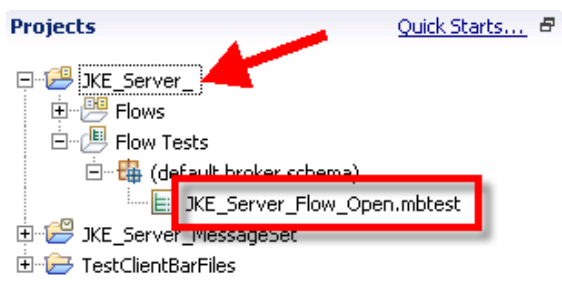


- __15. Return to the parent flow.
- __16. Wire in the subflow by connecting the **JKE_In_Request** terminal of the **JKE_WSDL_ServerService** to the **In** terminal of the subflow.
- __17. Wire the **JKE_Out_Response** terminal of the subflow to the **In** terminal of the SOAP Reply node.
- __18.  Save the message flow <Ctrl-S>.

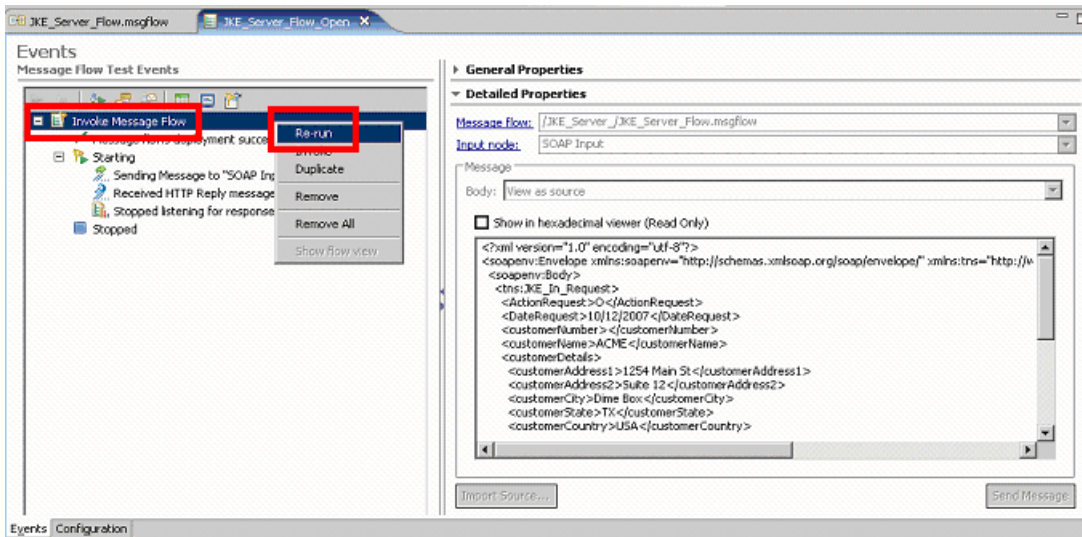


- __19. Select the small triangle opposite the Working set tab.
- __20. Select **JKE_Server_** from the menu.

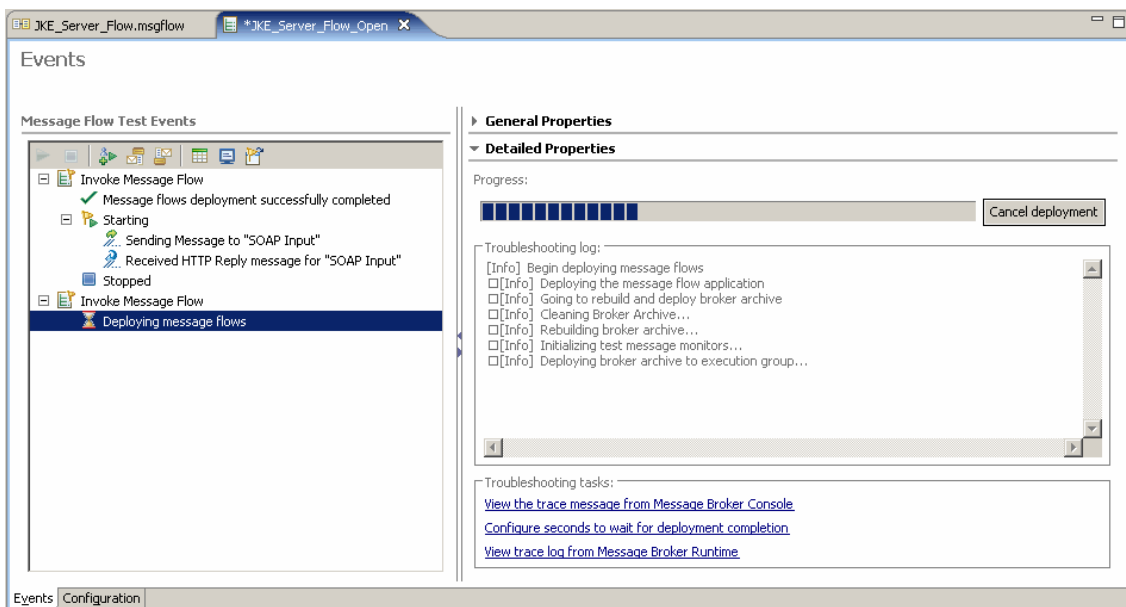
You are now ready to test your updated message flow. You will do this by reusing the Test Client configuration you saved at the end of Lab 5.



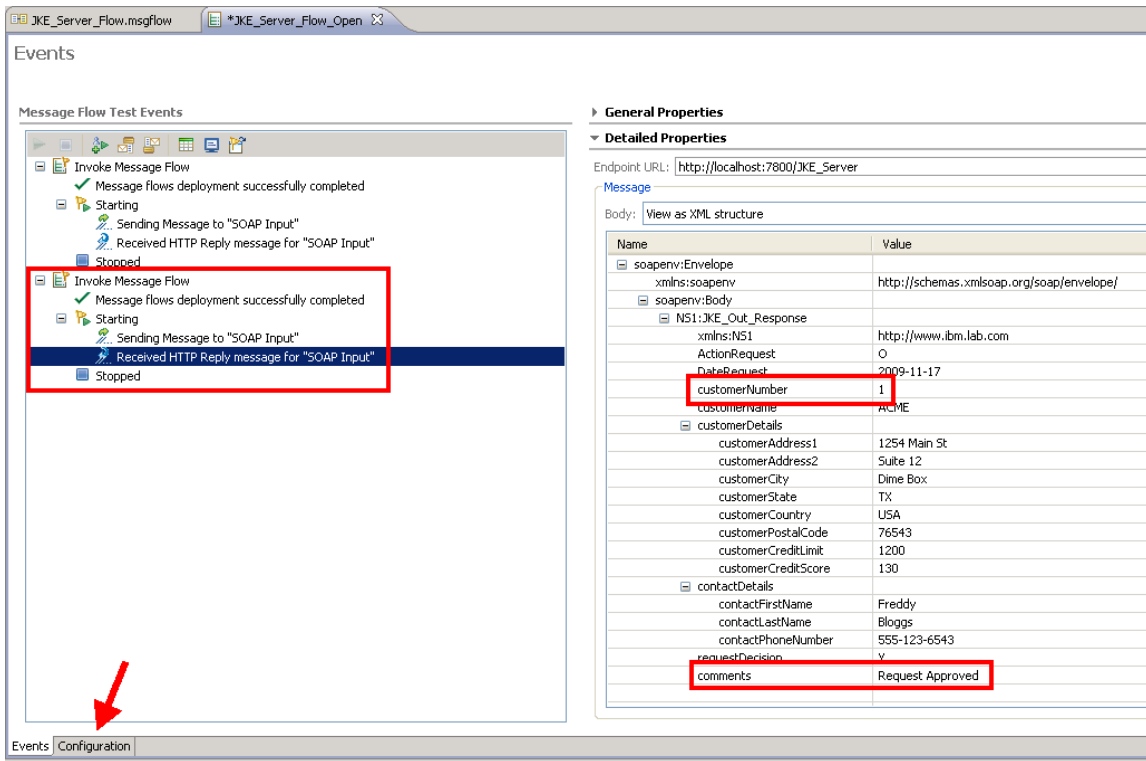
- __21. In the Broker Development pane, expand the **JKE_Server_->Flow Tests->(default broker schema)** folders.
- __22. Locate the **JKE_Server_Flow_Open.mbttest** test that was saved earlier.
- __23. Double-click on **JKE_Server_Flow_Open.mbttest** to launch the Test Client using the saved configuration.



- __24. When the Test Client has started, right-click on **Invoke Message Flow**.
- __25. Select **Re-run**. This will rerun the same test you did at the end of Lab 5.



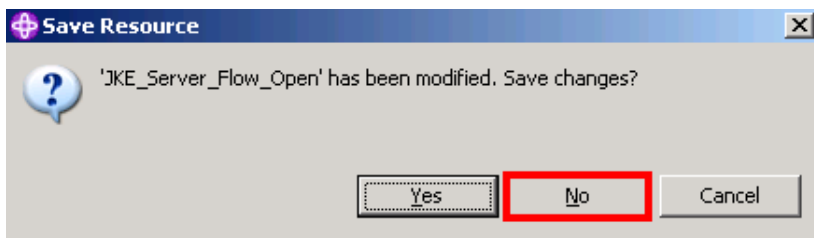
The Test Client deploys the message flow. You may observe the build and deploy process in the Progress window.



A response is received. This test has invoked both your message flow as well as the JKE_Client message flow that implements the Account Open Web service. JKE_Client also invokes the TwineBall EIS via an integrated WebSphere adapter. As you run additional tests using the Account Open message, observe the value of customerNumber. TwineBall is doing inserts to a Derby database and returning a unique sequential key that is used as the customer number. Your values may not match the pictures in the Lab Guide, but it should increment with repeated invocations.

The JKE_Client flow has changed the comments field from the input message to **Request Approved**.

The Test Client has a Configuration tab at the bottom of its panel that provides for additional flexibility. For example, if you want to manage the building and deployment of the bar file, that option is available via the configuration tab.



__26. **Close**, but do not save, the Test Client.

This is the end of Lab 6.

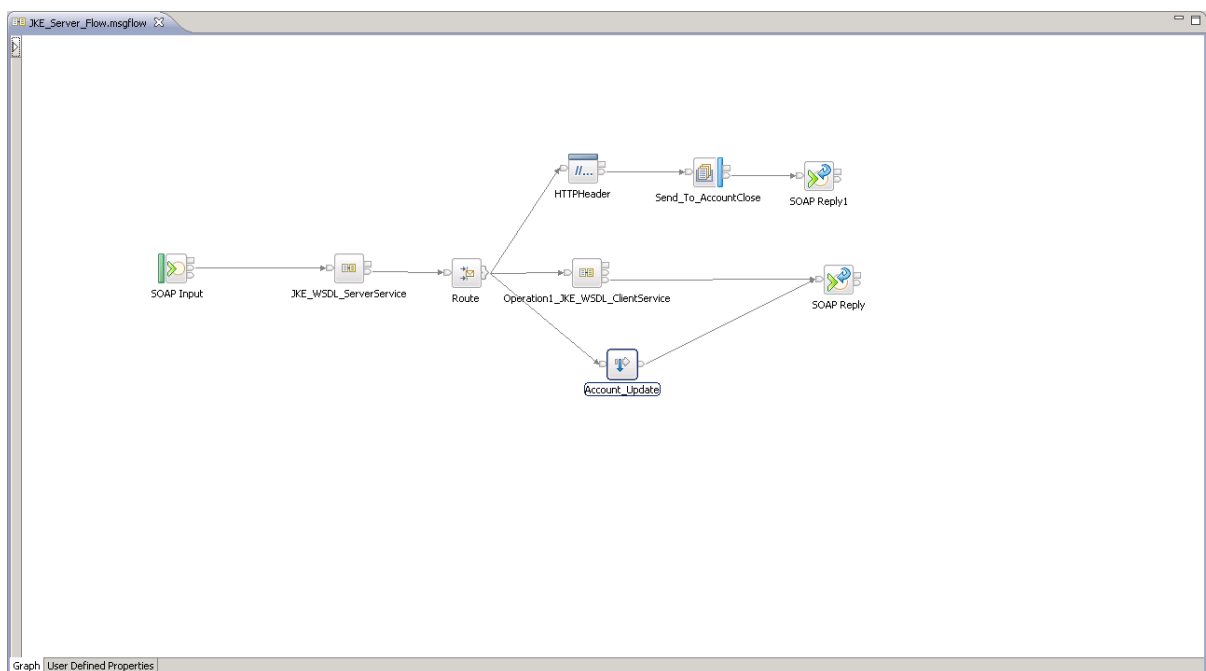
Lab 7 Building the JKE Server – Routing

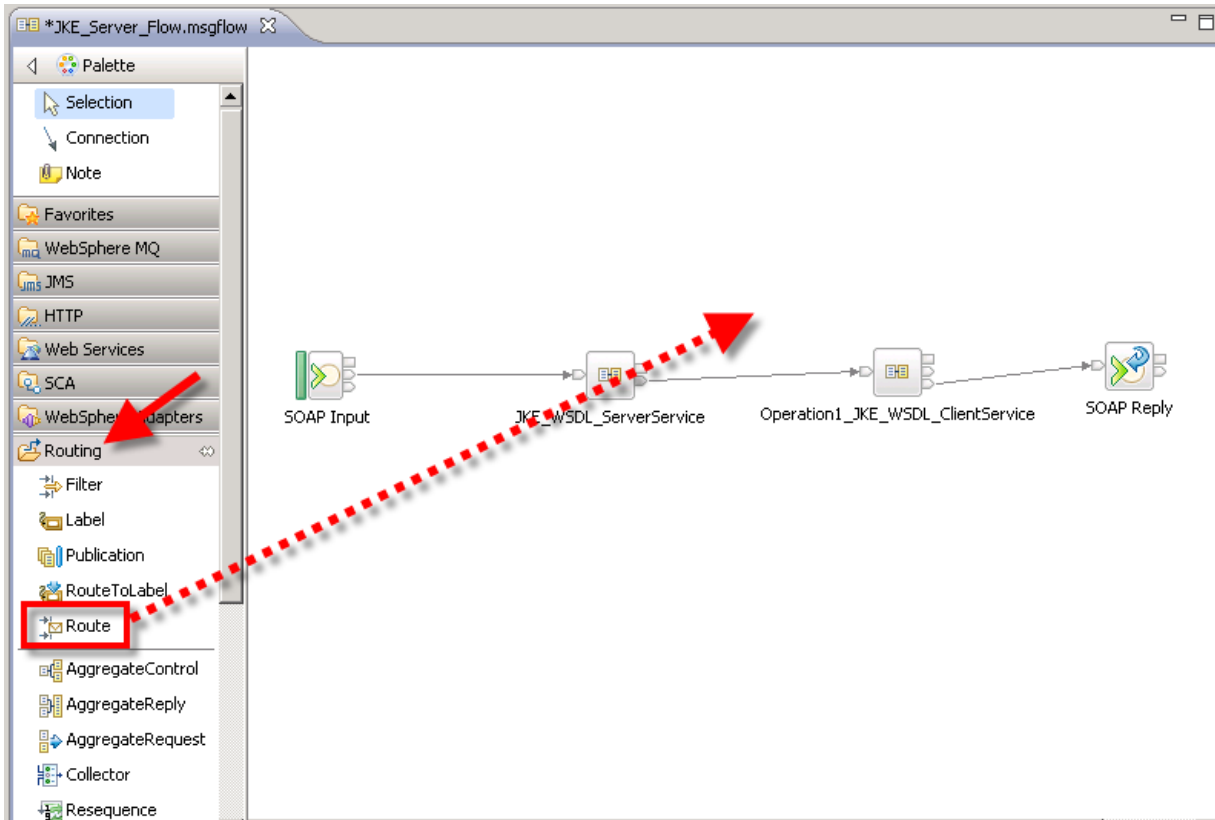
7.1 Lab Overview

The JKE Server Web service must be able to handle three types of requests: Account_Open, Account_Update and Account_Close. In this lab you will build the routing infrastructure to send the request down the appropriate path. The Account_Open path will invoke the JKE Client message flow that implements the Account Open Web service. The Account_Update path will be completed in a later lab and will invoke an IBM CICS® back-end system. The Account_Close path will invoke a WebSphere MQ application that is not part of this scenario.

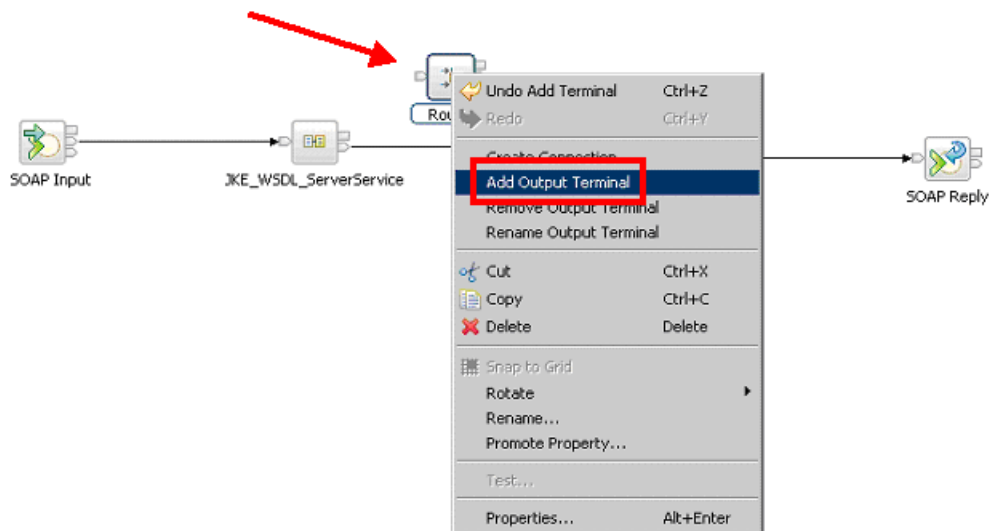
You will then test each of the three paths.

A Route node will be used in this lab. The Route node is capable of handling many routing decisions using logical tests. These tests are configured with XML Path Language (XPath) expressions that may refer to a part of the message, a part of the SOAP Envelope or another location. At this stage of flow development the Account_Update path will use a Trace node to verify a message has been routed down this path.

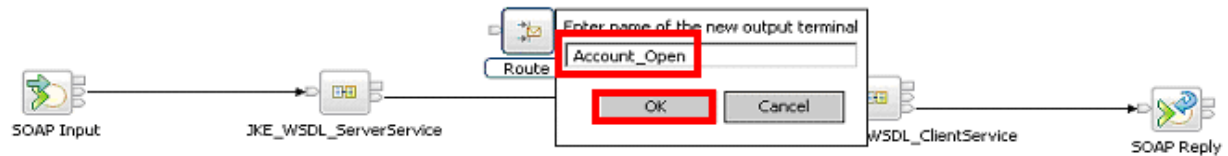




- __1. Expand the **Routing** drawer.
- __2. Drag and drop the **Route** node above the connector as shown.

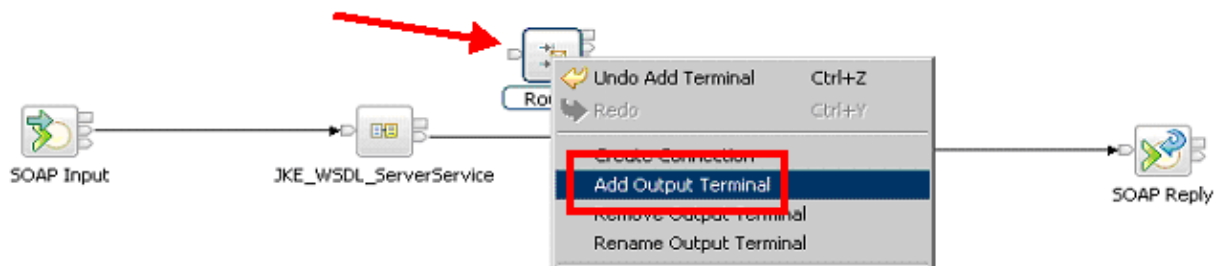


- __3. Right-click the **Route** node.
- __4. Select **Add Output Terminal**. Note: You will be adding three of these in the next series of steps.



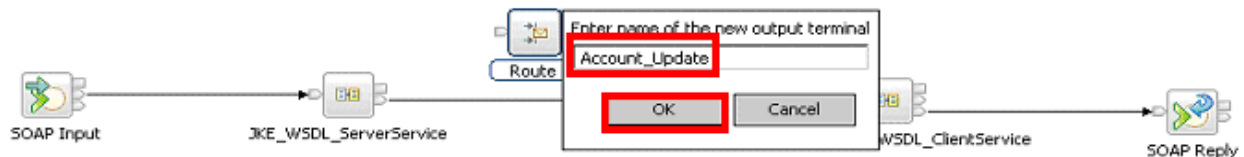
__5. Enter **Account_Open** for the name of the new output terminal.

__6. Click **OK**.



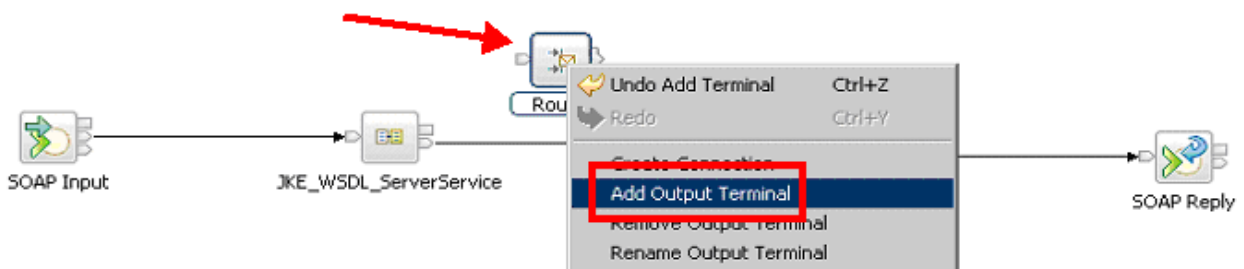
__7. Again, right-click the Route node.

__8. Select **Add Output Terminal**.



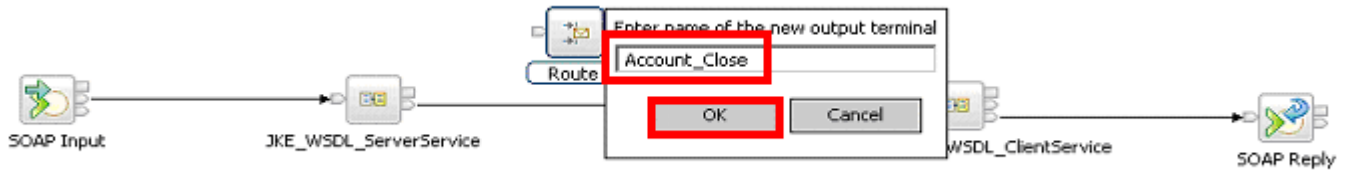
__9. Enter **Account_Update** for the name of the new output terminal.

__10. Click **OK**

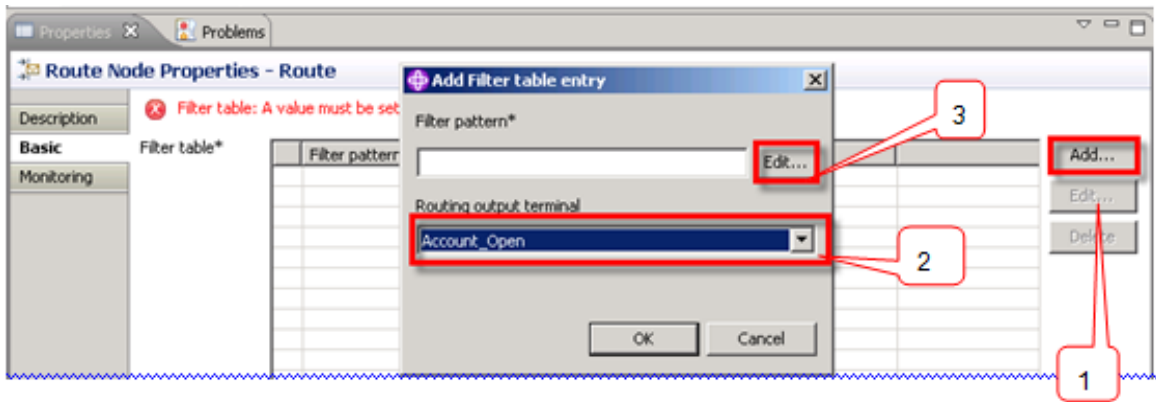


__11. Again, right-click the Route node.

__12. Select **Add Output Terminal**.



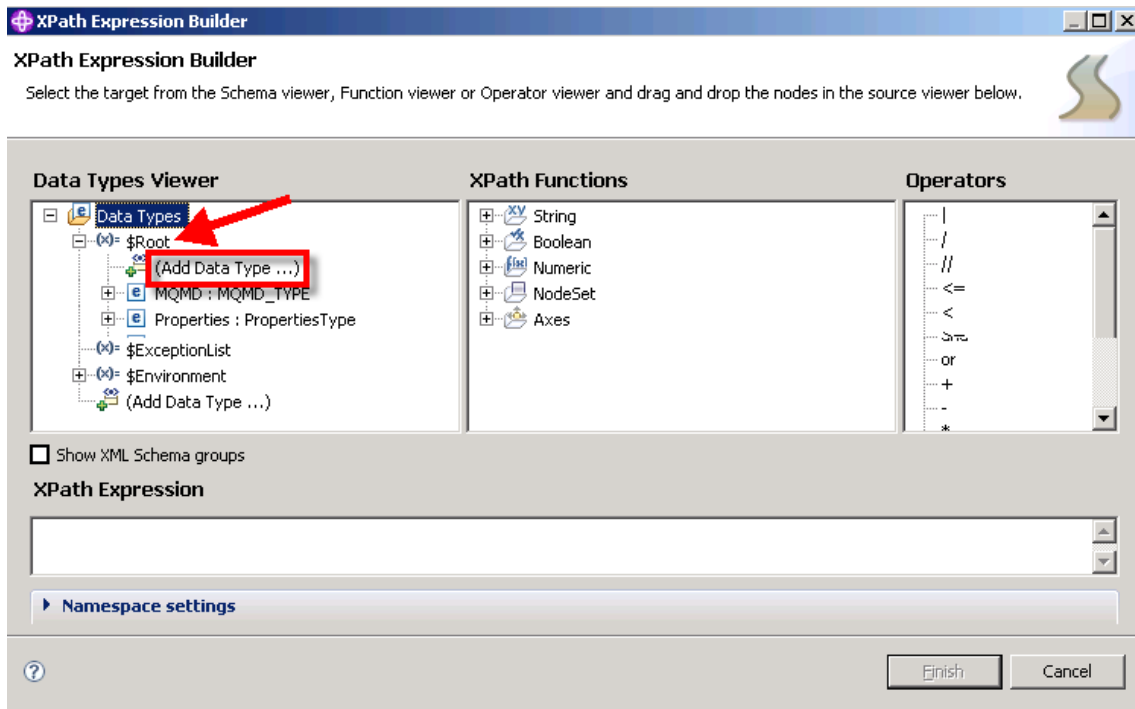
- __13. Enter **Account_Close** for the name of the new output terminal.
- __14. Click **OK**.



You have now completed the definition of the output terminals for the Route node.

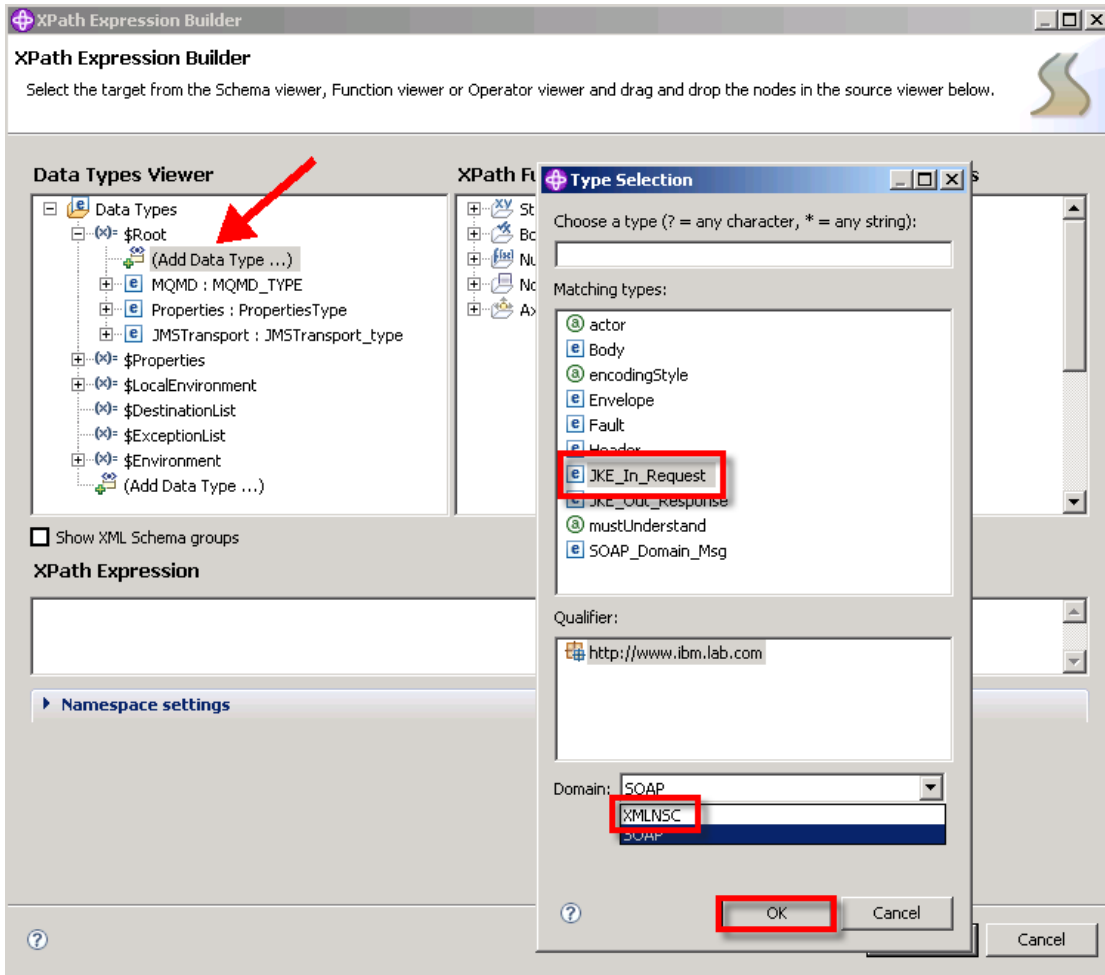
You now will define the decision criteria that will be used to route a message through the flow. This will require a logic statement that will identify a message type for each of the three output terminals. So the next several steps will be repeated three times.

- __15. In the Properties pane, click the **Add** button. It will display the Add Filter table entry panel.
- __16. Click the **Routing output terminal** pull-down.
- __17. Select **Account_Open**.
- __18. Click **Edit** to start the XPath Expression Builder.



The XPath Expression Builder is very versatile. It allows you to easily build an XPath expression for making a routing decision.

- __19. Expand the **\$Root** entry.
- __20. Select **Add Data Type**.

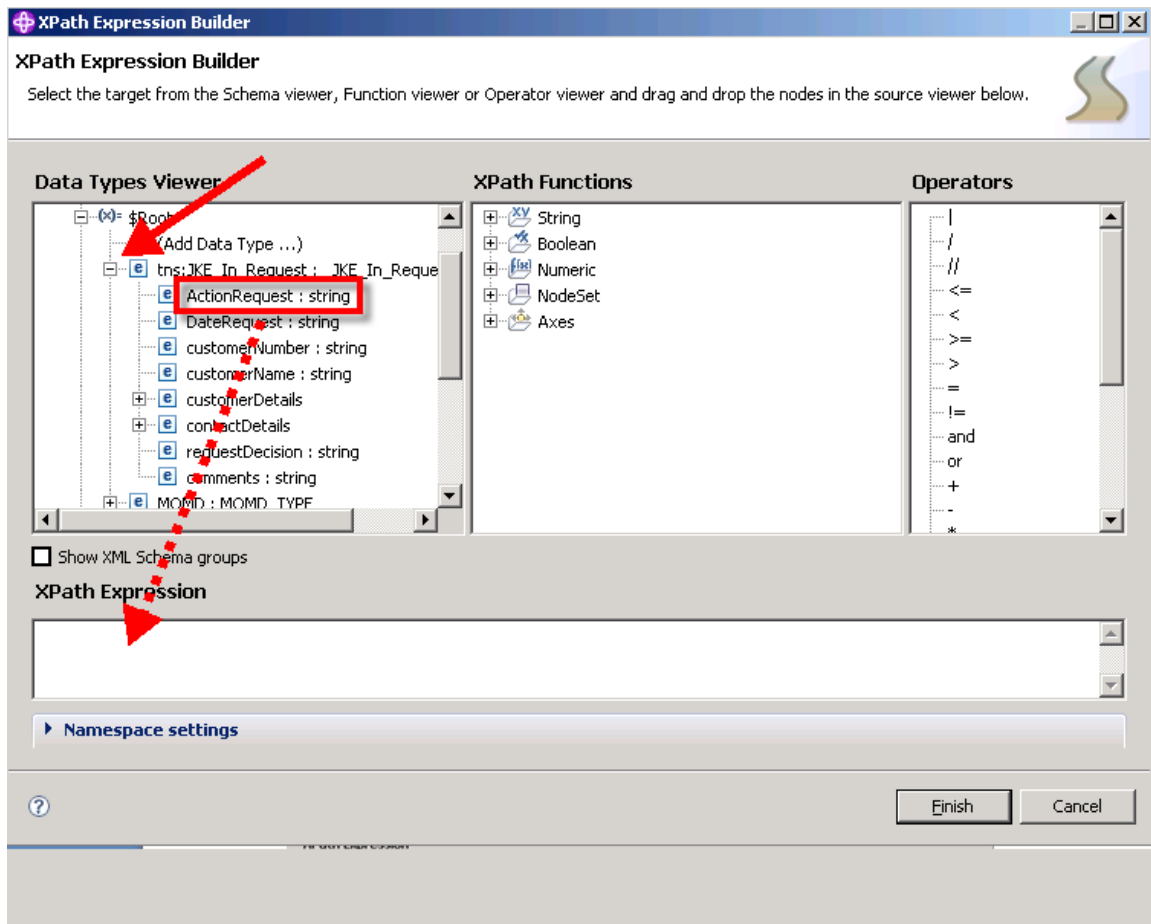


__21. Select **JKE_In_Request**.

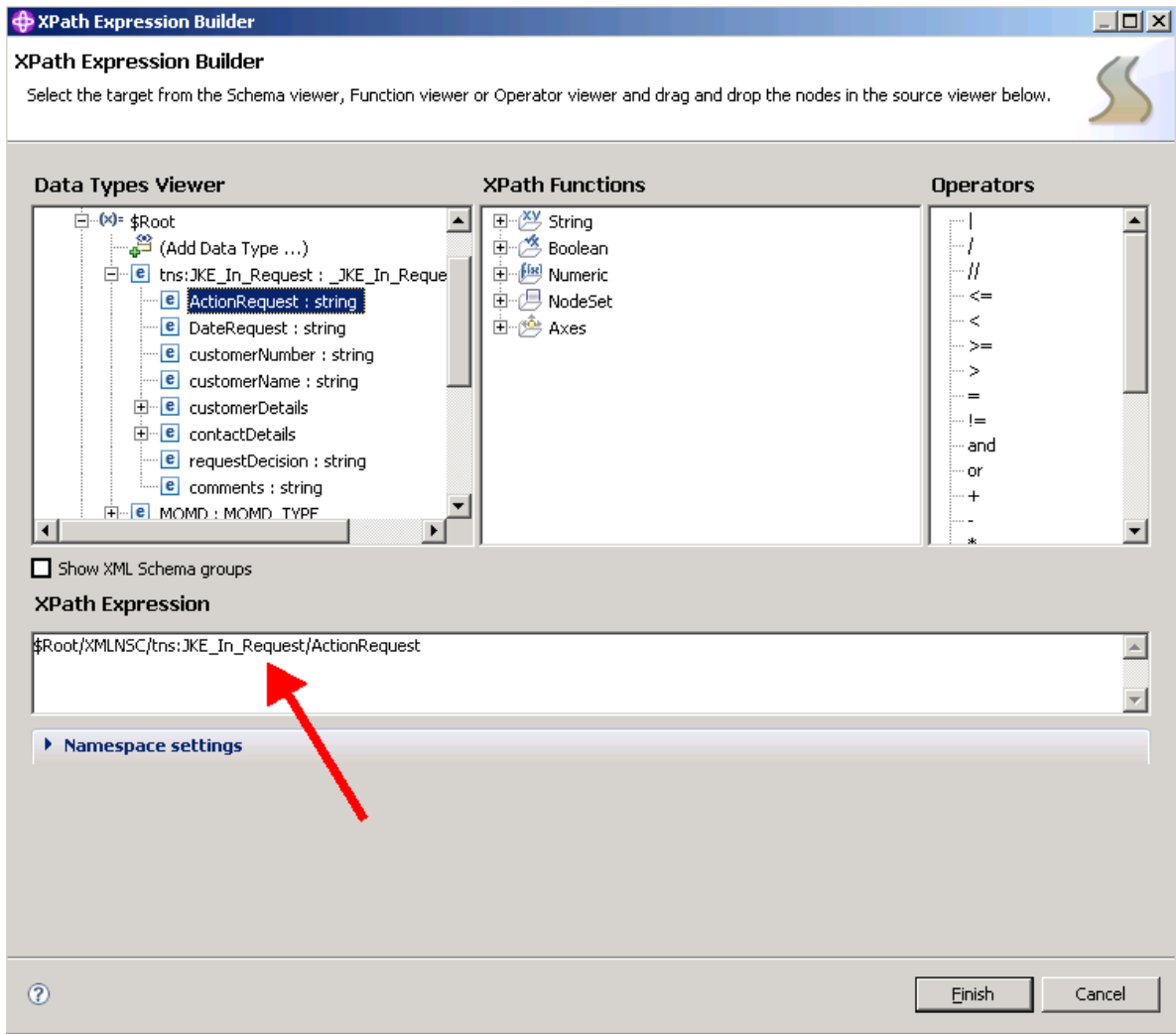
__22. Use the drop down menu to select the **XMLNSC** domain.

__23. Press the **OK** button.

Note that the list contains all of the message definitions that are known to the workspace.

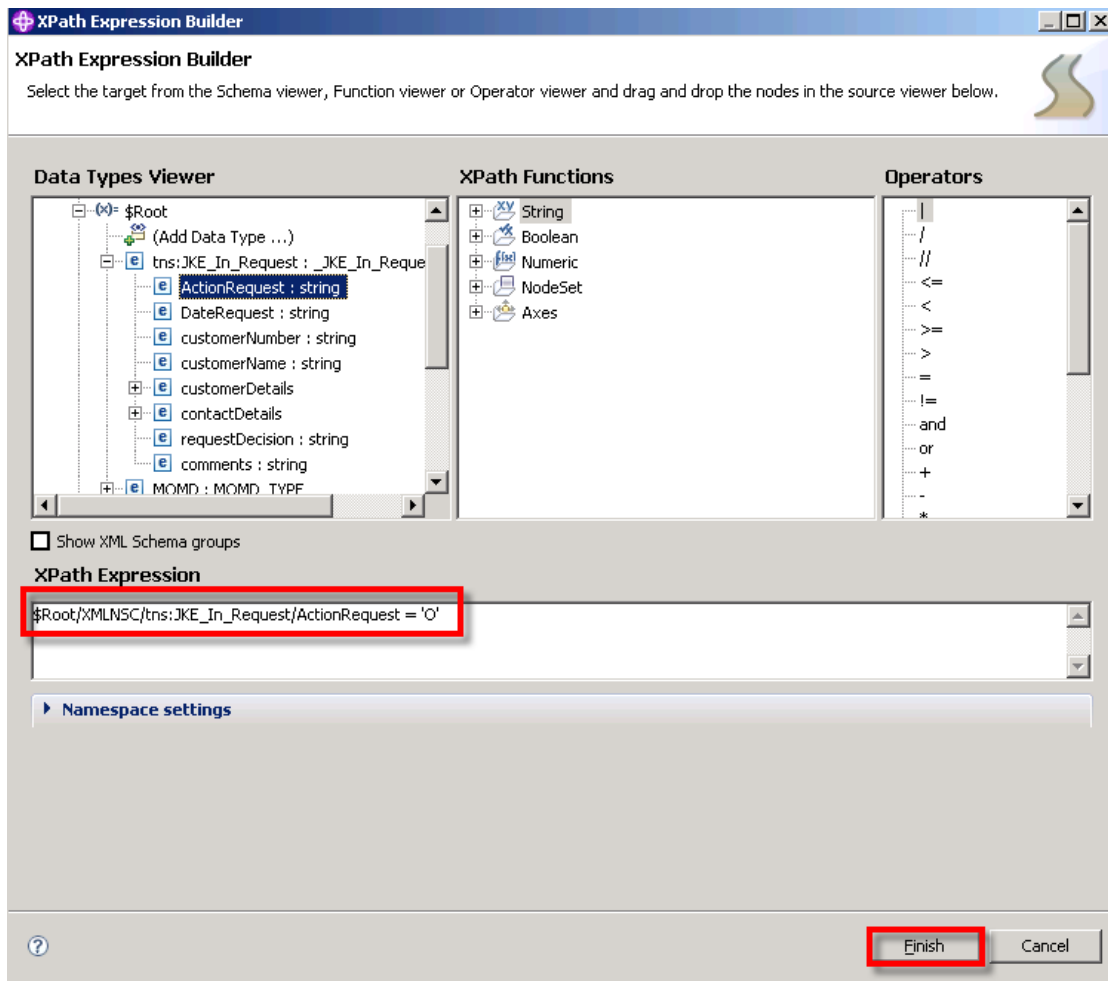


- __24. Expand the **tns:JKE_In_Request**.
- __25. Select the **ActionRequest** field.
- __26. Drag it down to the **XPath Expression** area.



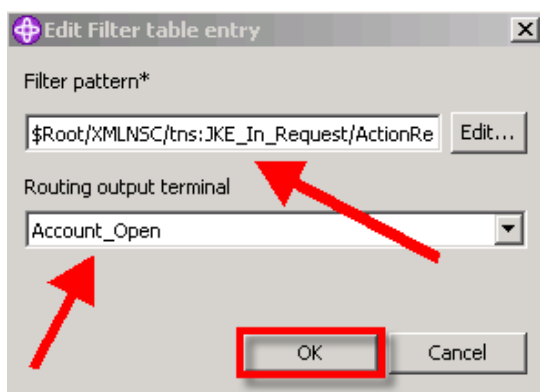
This is the result. On the next page you will be adding a comparison value. On the right side is a list of Operators which can also be used in a drag and drop fashion. In the middle section is a list of potential XPath functions that could also be used. However, the tests you will be using are very simple. The ActionRequest is a single character. An 'O' is for Account_Open, a 'U' is for Account_Update and a 'C' is for Account_Close.

Note! Be sure to enter the above ActionRequest characters in *uppercase!*

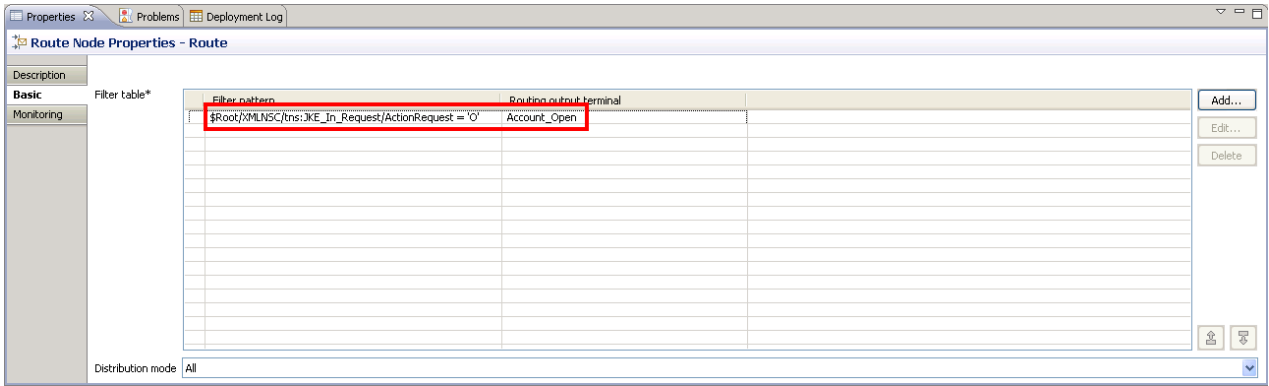


__27. Add the following: = 'O' to the XPath Expression.

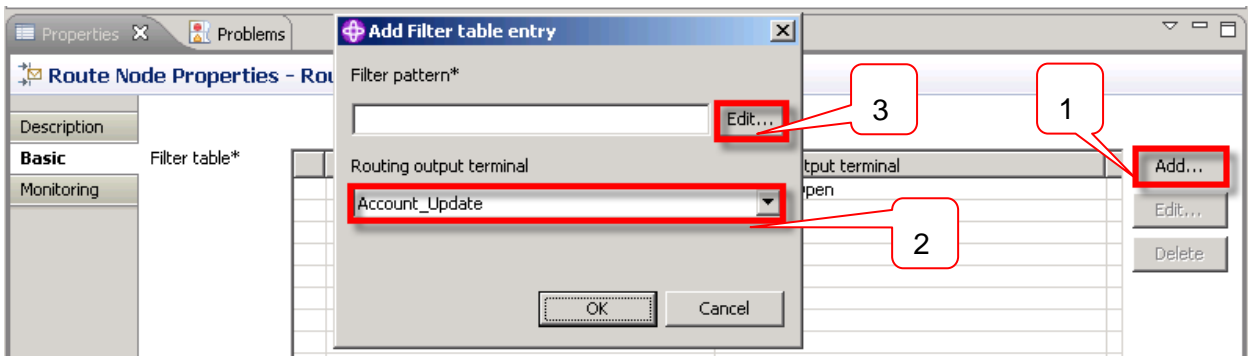
__28. Click **Finish**.



__29. Click **OK** to complete this entry.

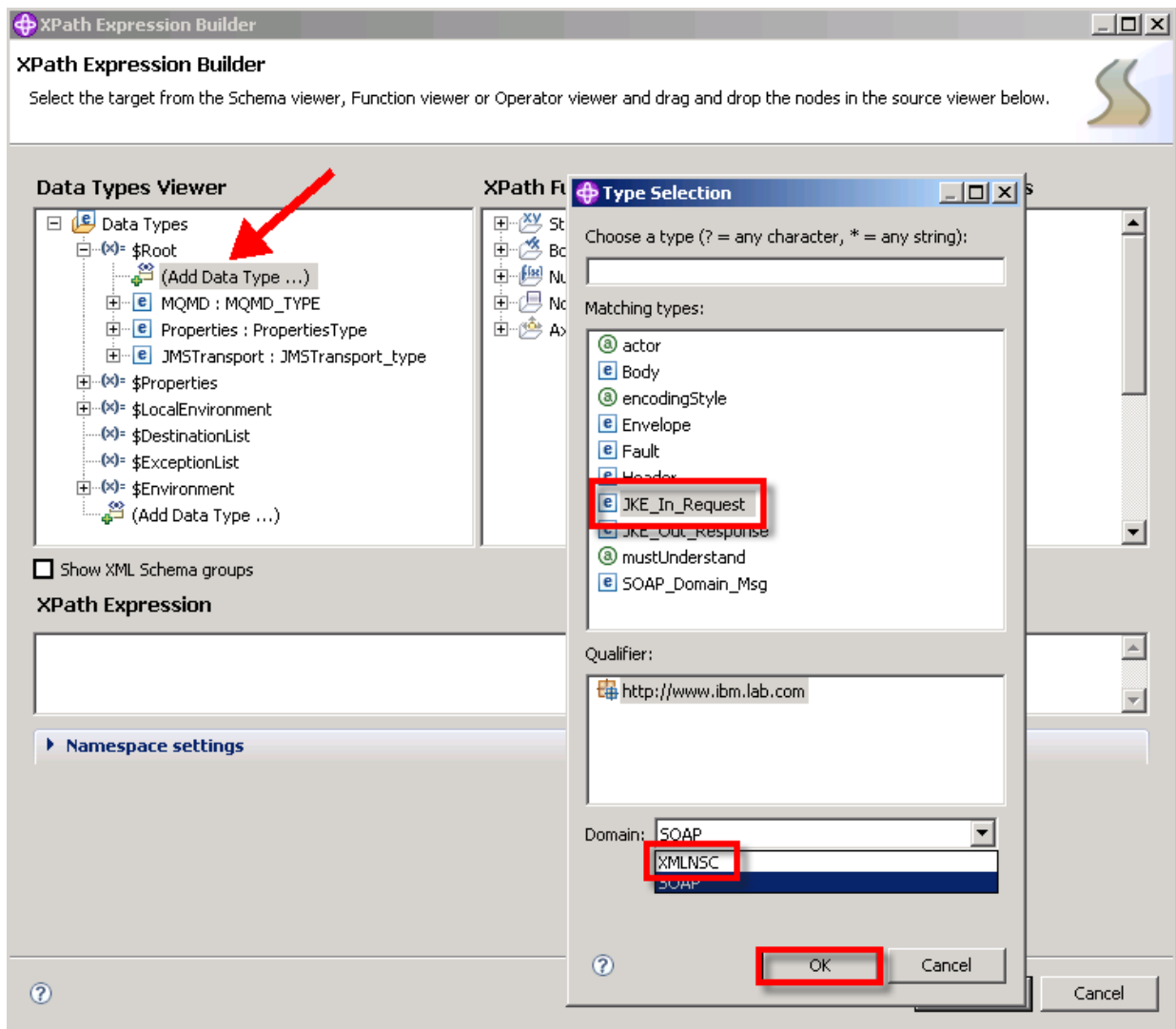


__30. As each entry is completed, it is shown in the Properties. You may make the Filter pattern area wider to see the entire expression. If you make a mistake, you can highlight the entry and use the Edit button to change it.

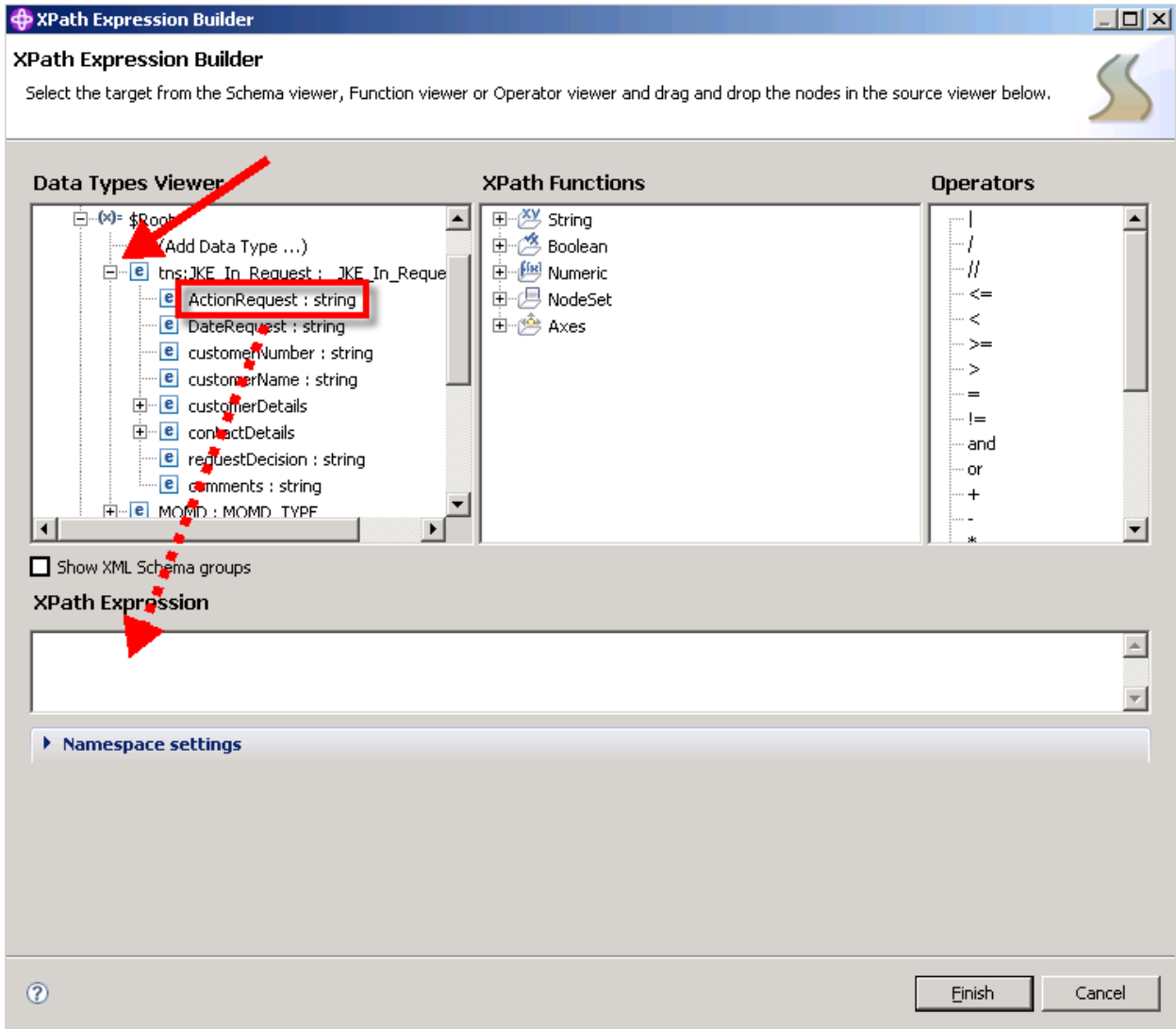


Repeat the process for Account_Update.

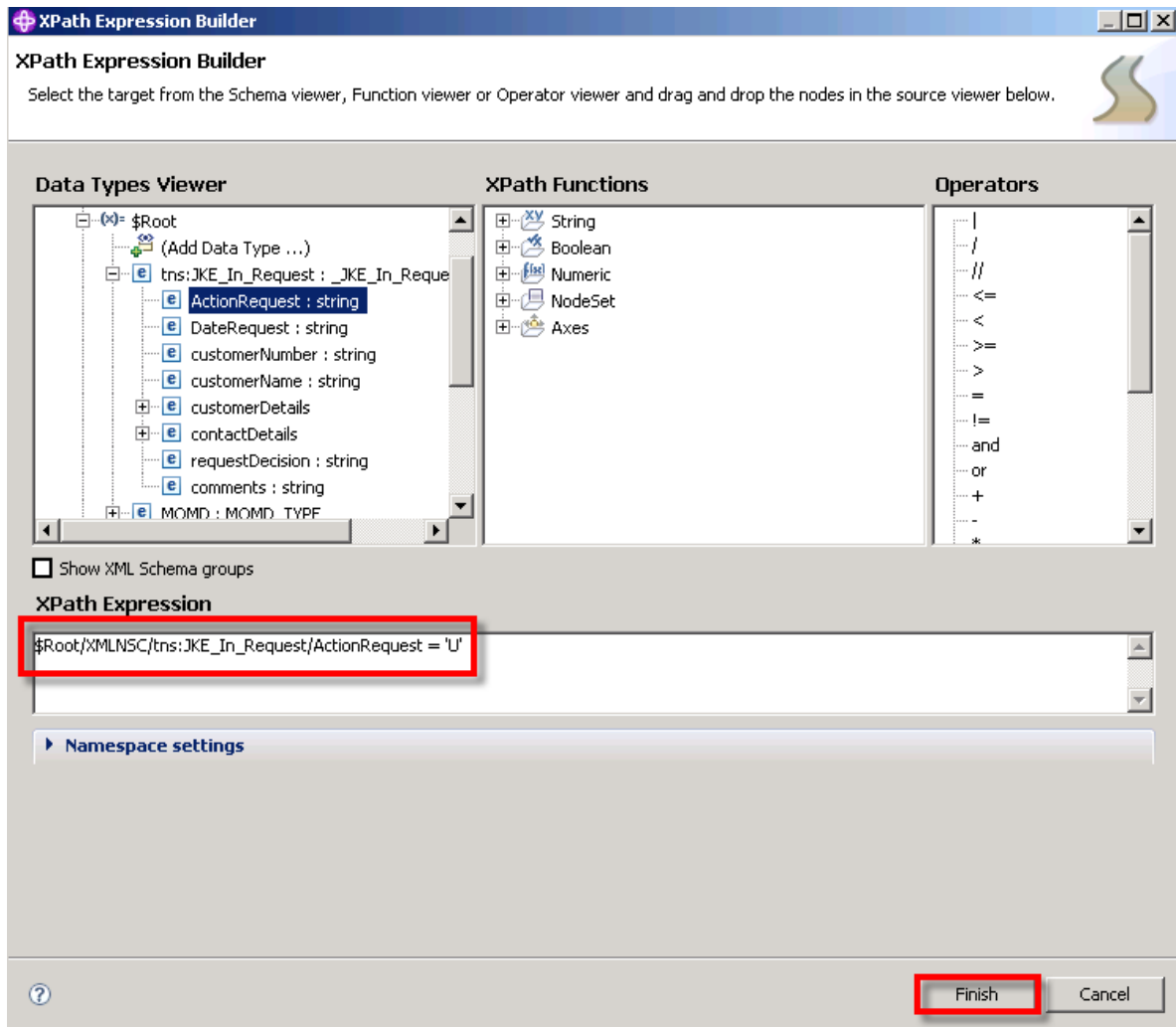
- __31. Click the **Add** button.
- __32. Use the pull-down to select **Account_Update** as the output terminal name.
- __33. Click the **Edit** button.



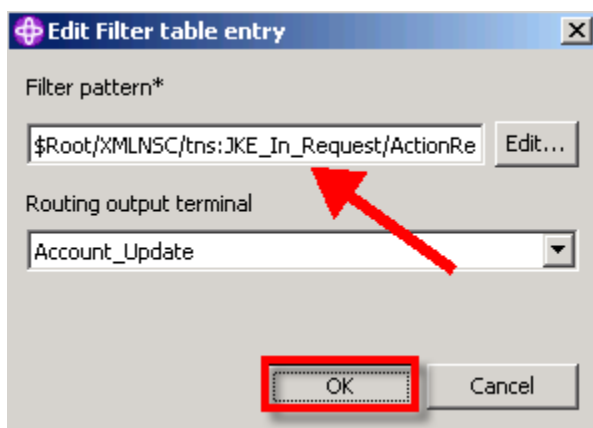
- __34. Expand **\$Root**.
- __35. Click **Add Data Type**.
- __36. Select **JKE_In_Request**.
- __37. Use the drop down menu to select the **XMLNSC** domain.
- __38. Press the **OK** button to add the data type to the viewer.



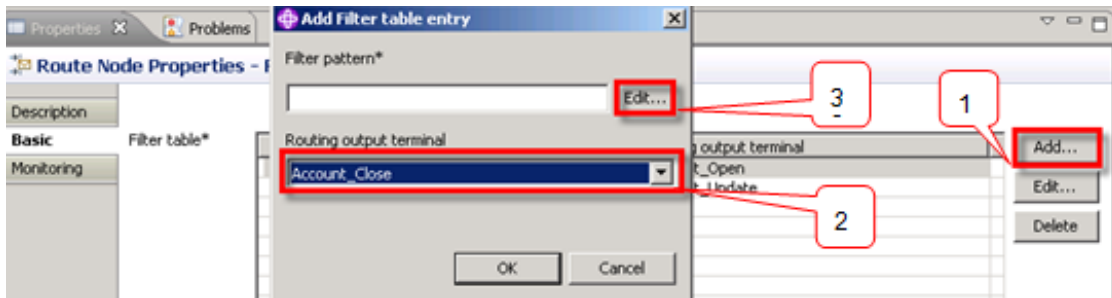
- __39. Expand **tns:JKE_In_Request**.
- __40. Drag **Action_Request** to the XPath Expression area.



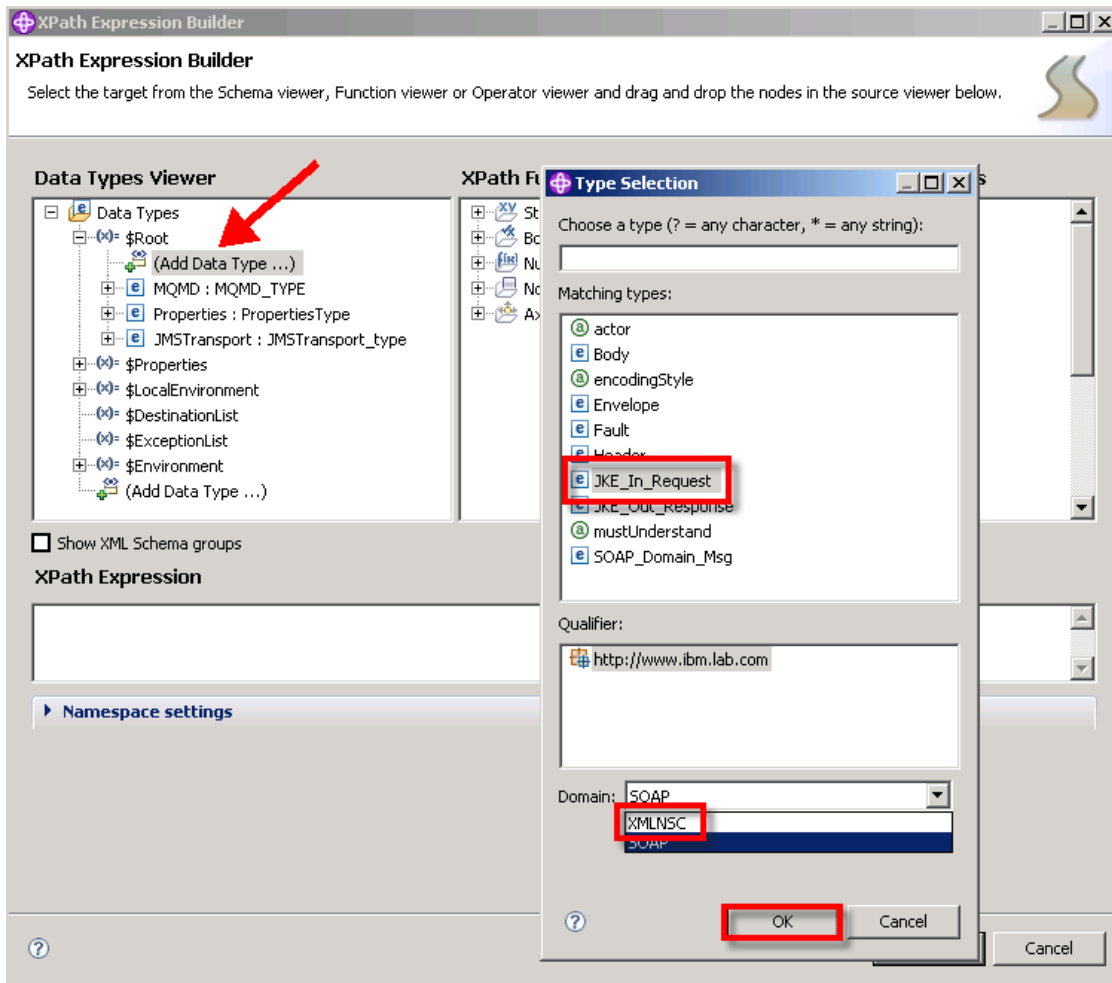
- __41. Complete the XPath statement by entering: = 'U'.
- __42. Press the **Finish** button.



- __43. Click **OK**.

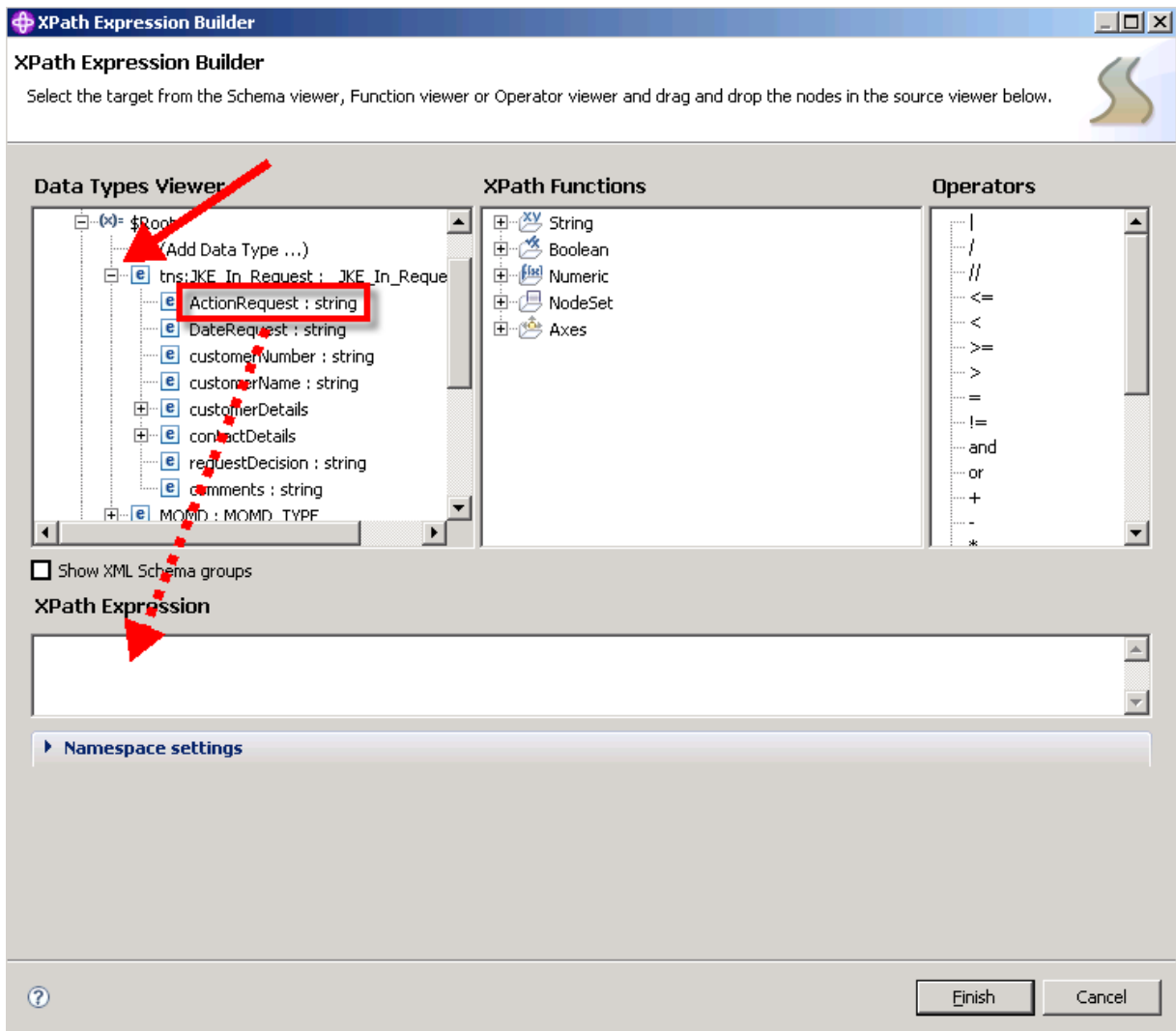


- __44. For the third filter expression, click **Add**.
- __45. Change the output terminal to **Account_Close** (using the pull-down)
- __46. Click the **Edit** button.

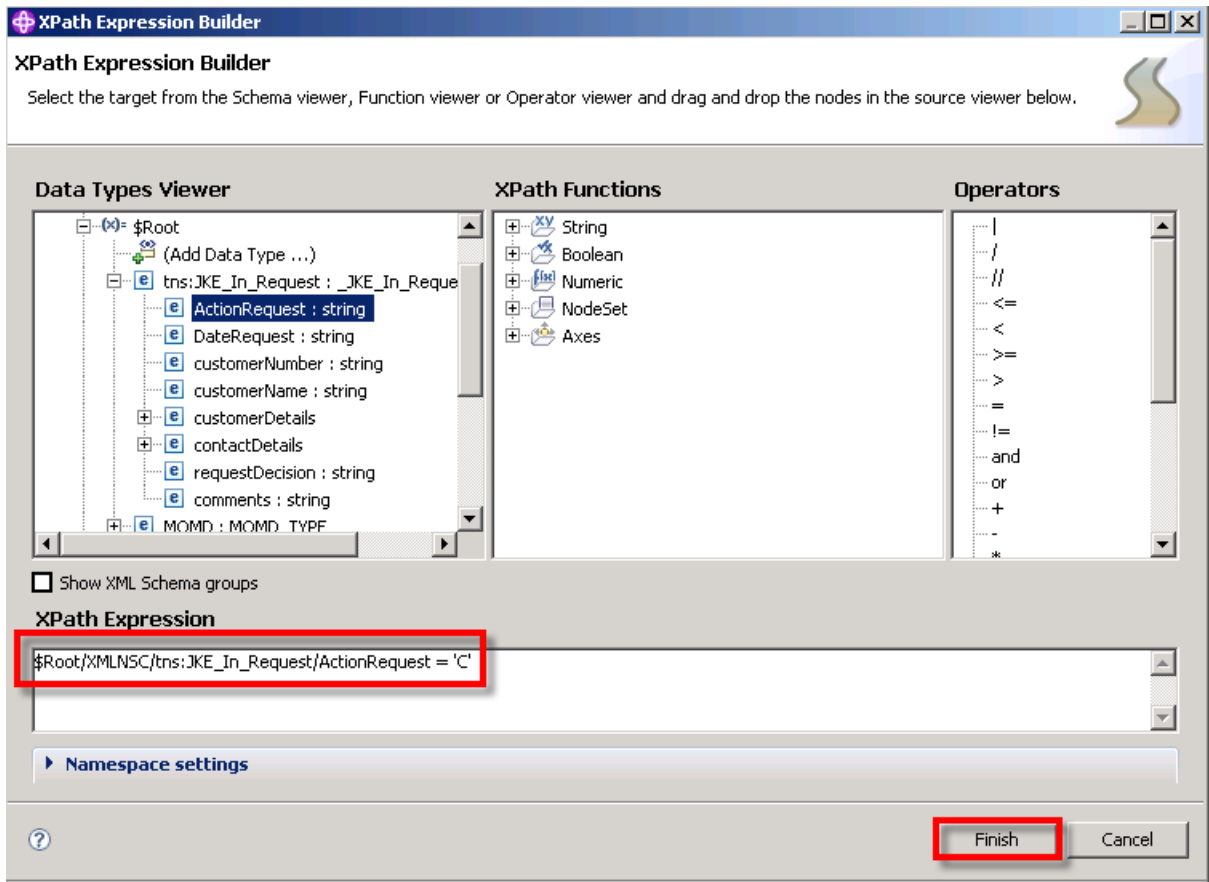


- __47. Expand **\$Root**.
- __48. Click **Add Data Type**.
- __49. Select **JKE_In_Request**.

- __50. Use the drop down menu to select the **XMLNSC** domain.
- __51. Press the **OK** button to add the data type to the viewer.

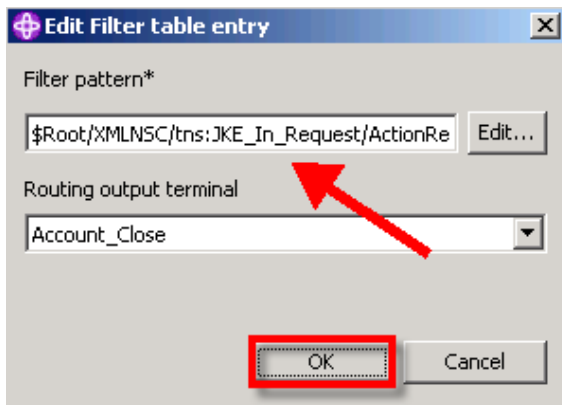


- __52. Expand **tns:JKE_In_Request**.
- __53. Drag **Action_Request** to the XPath Expression area.

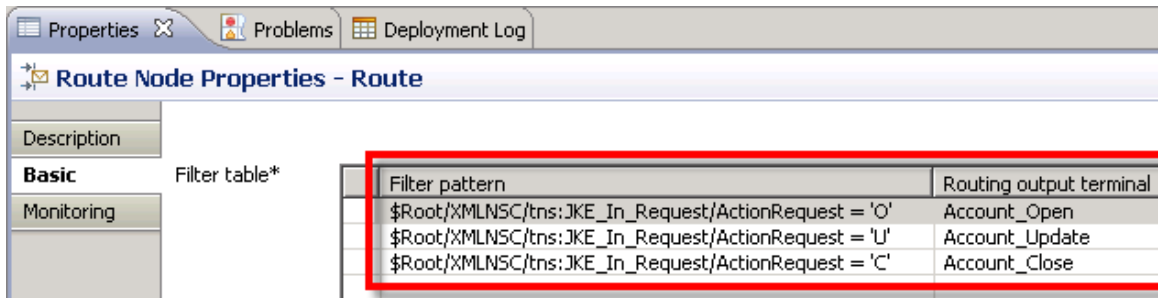


__54. Complete the entry by adding an = 'C' to the end.....make sure the C is in caps!

__55. Click the **Finish** button

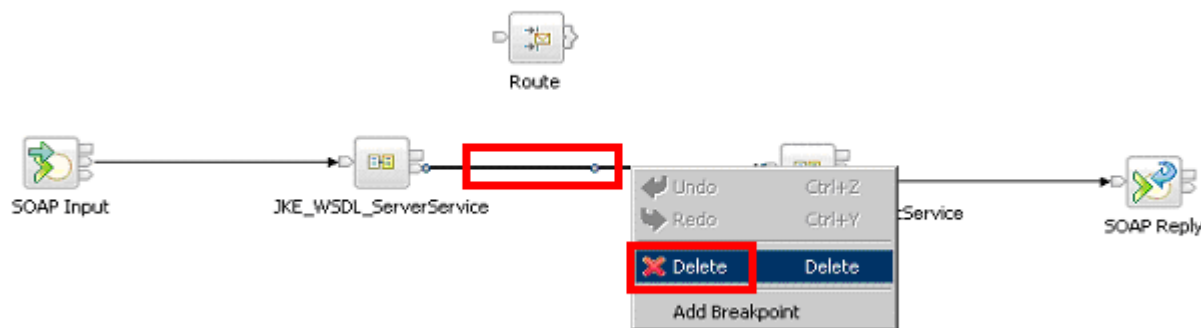


__56. Click **OK**.



__57. Verify that all three Filter expressions are correct and that they match the Output Terminals.

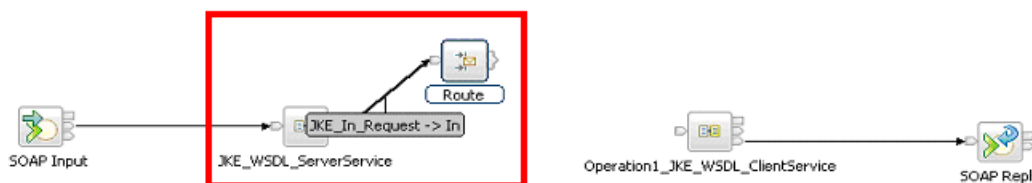
The JKE_WSDL_ServerService now must be wired to the Route node.



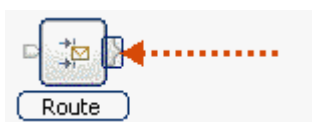
Delete the connector between JKE_WSDL_ServerService and Operation1_JKE_WSDL_ClientService

__58. Right-click on the connector.

__59. Select **Delete** from the menu.

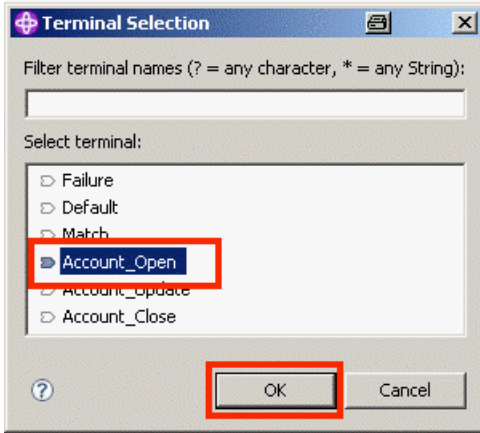


__60. Connect the JKE_In_Request terminal of the JKE_WSDL_ServerService node to the In terminal of the Route node.



The wiring for the first of the three output terminals of the Route node will now be completed.

__61. On the Route node, click on the cluster of output terminals as shown above.



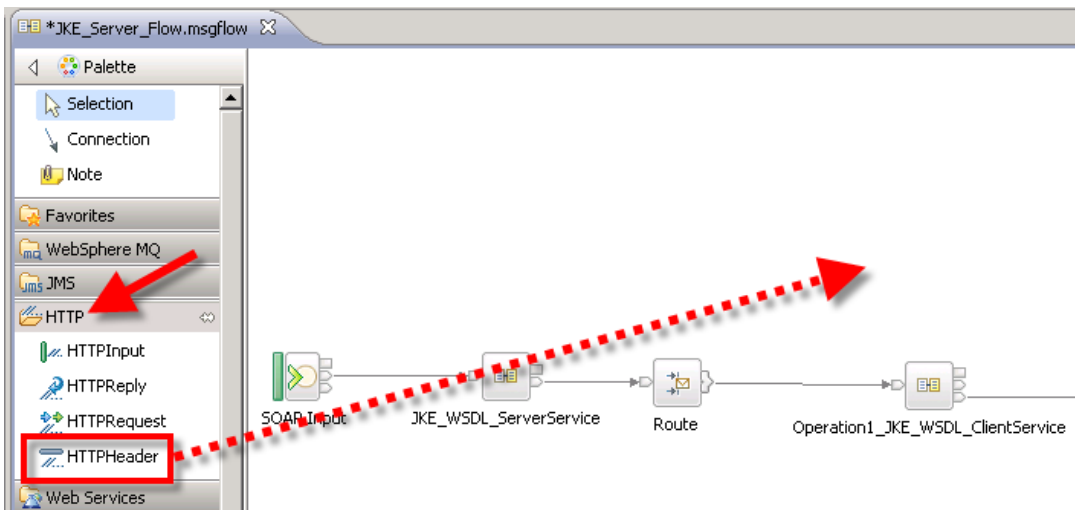
The Terminal Selection panel will be displayed.

__62. Select **Account_Open**.

__63. Click **OK**.



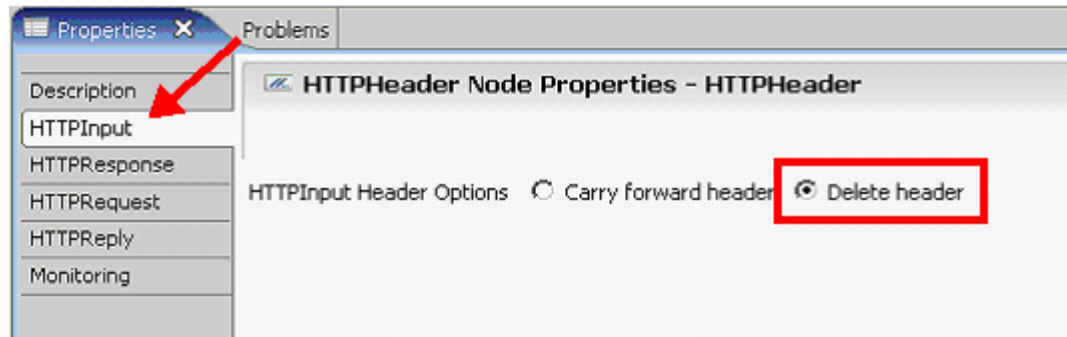
__64. The resulting connector for the Account_Open output terminal should be connected to the **In** terminal of the **Operation1_JKE_WSDL_ClientService** subflow.



The path for the Account_Close request will be built next.

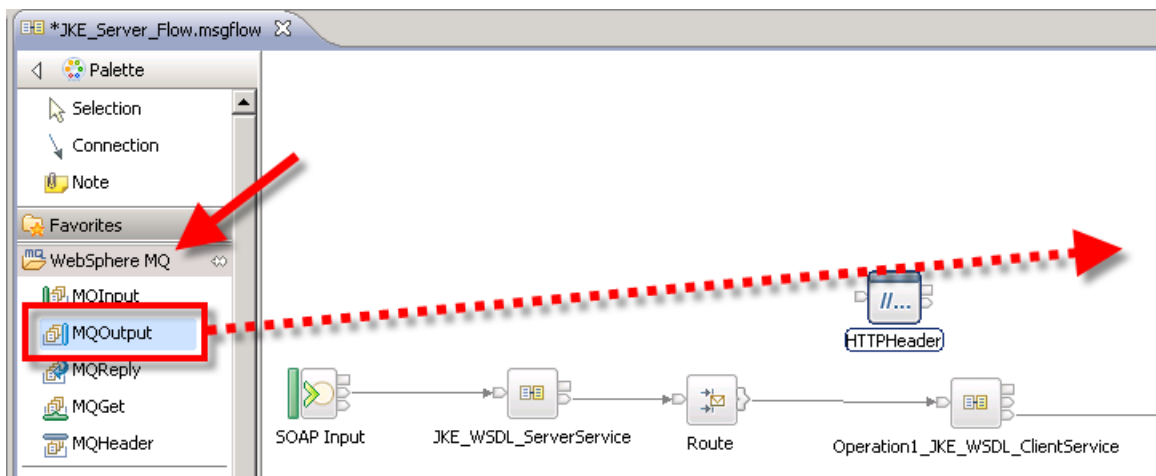
__65. Expand the **HTTP** drawer.

__66. Drag and drop an **HTTPHeader** node as shown.

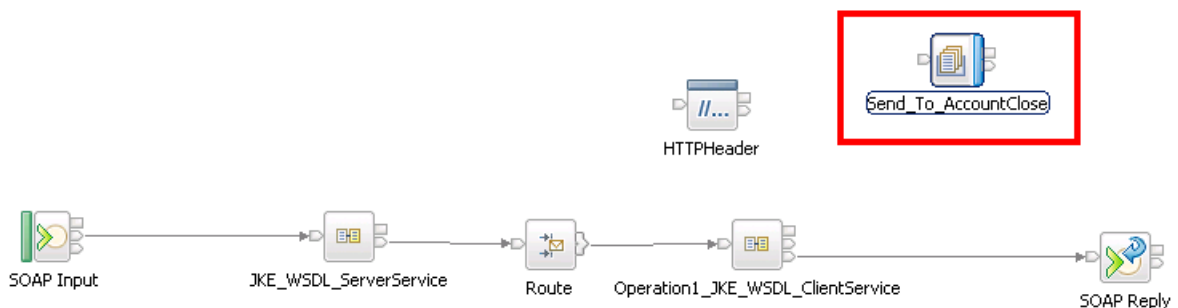


This node will be used to remove the HTTP Input header from the message tree. This will allow the JKE_In_Request message to be sent to a WebSphere MQ based application for additional processing.

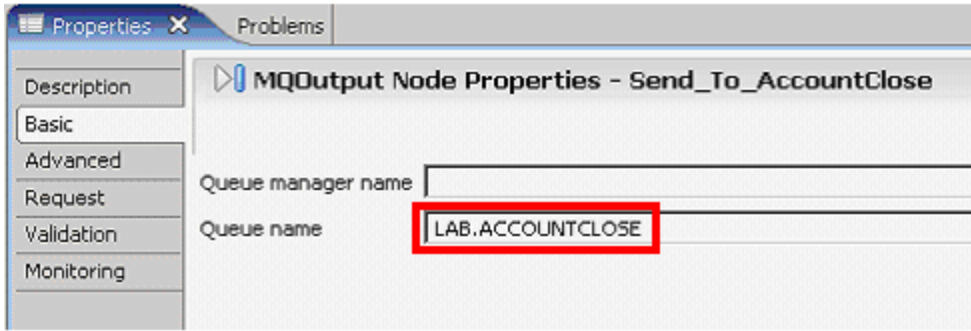
- __67. Select the **HTTPInput** tab.
- __68. Click the radio button for **Delete header**.



- __69. Expand the **WebSphere MQ** drawer.
- __70. Drag and drop an **MQOutput** node as shown.

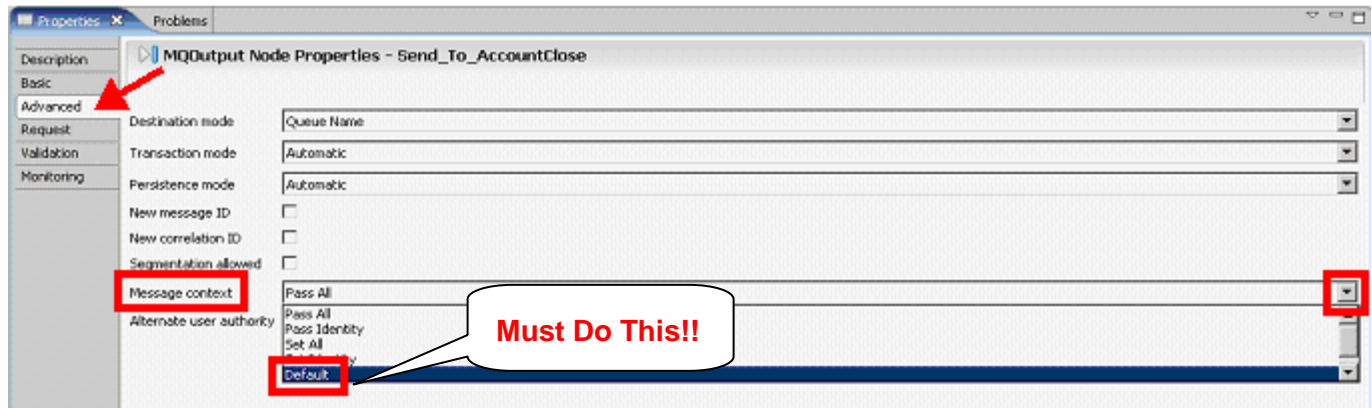


- __71. Rename the node to **Send_To_AccountClose**.

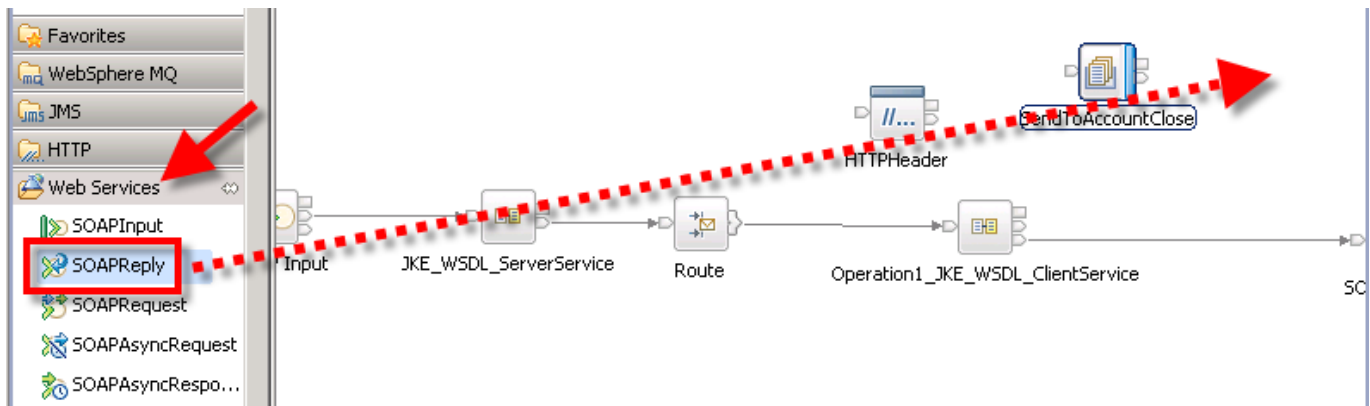


There are two Properties entries that must be set.

- __72. Select the **Basic** tab.
- __73. Enter **LAB.ACCOUNTCLOSE** as the queue name.



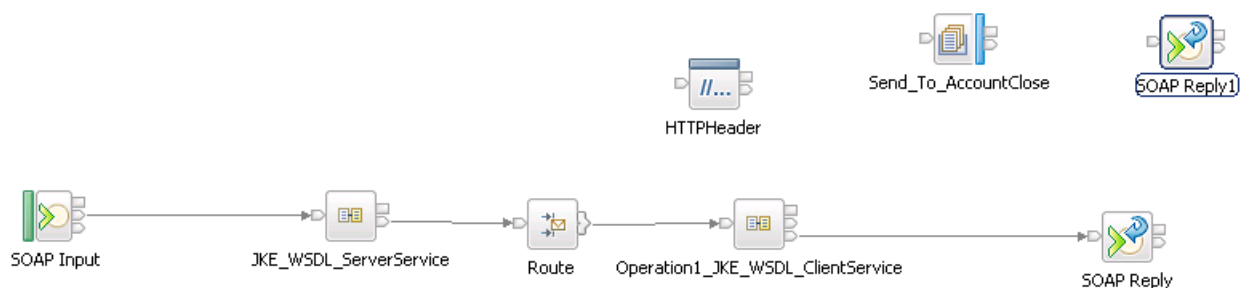
- __74. Select the **Advanced** tab.
- __75. Use the pull-down for **Message context** and select **Default**. **This option is required** because the message that initiated this flow used the HTTP protocol. Pass All tells the MQOutput node to pass the WebSphere MQ context in the message flowing into the node. However there is no context in the input message to pass since it did not originate as a WebSphere MQ message. Default will create a default MQMD in the message tree.



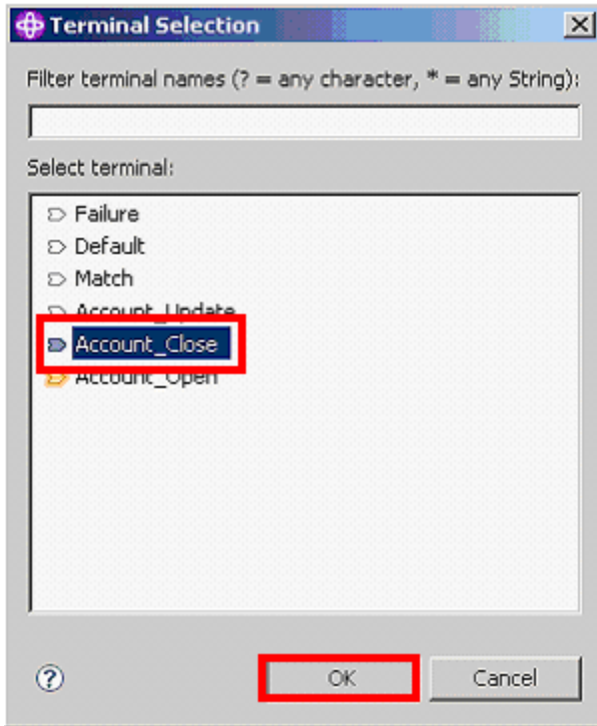
__76. For the last node on this path, Open the **Web Services** drawer.

__77. Drag and drop a **SOAPReply** node as shown.

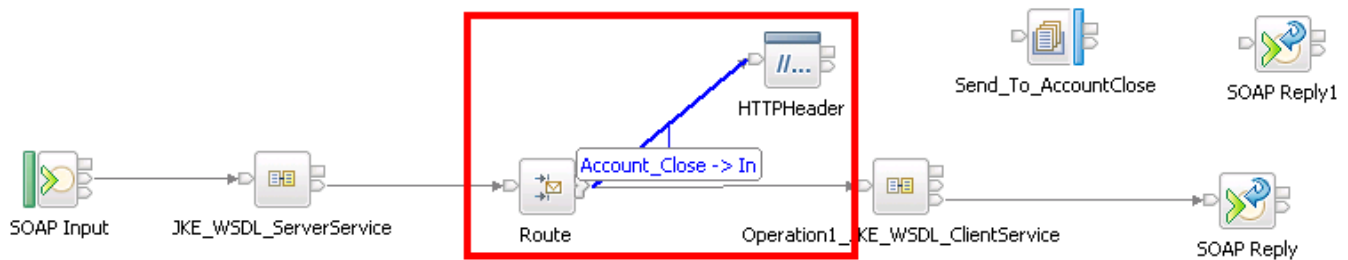
A second SOAPReply node is added here as a convenience. This will avoid connectors crossing each other in later labs. The existing SOAPReply node could have been used.



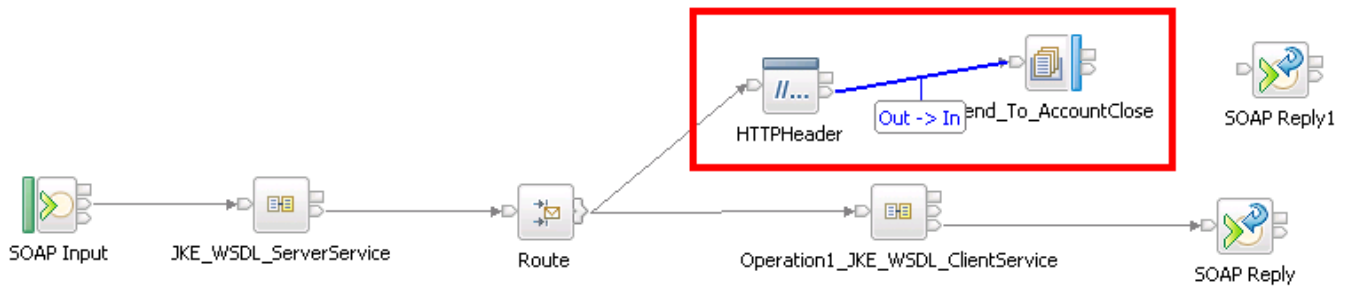
Here is the completed Account_Close path. The nodes will be wired together next.



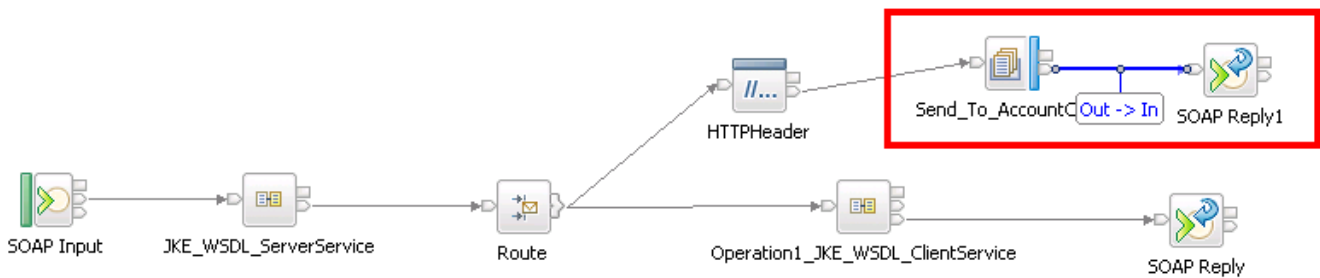
- __78. Start with the Route node. Click on the cluster of output terminals. Note that when the Terminal Selection panel is displayed, the Account_Open entry has a different color for its icon indicating that it has already been connected.
- __79. Select **Account_Close**
- __80. Click **OK**.



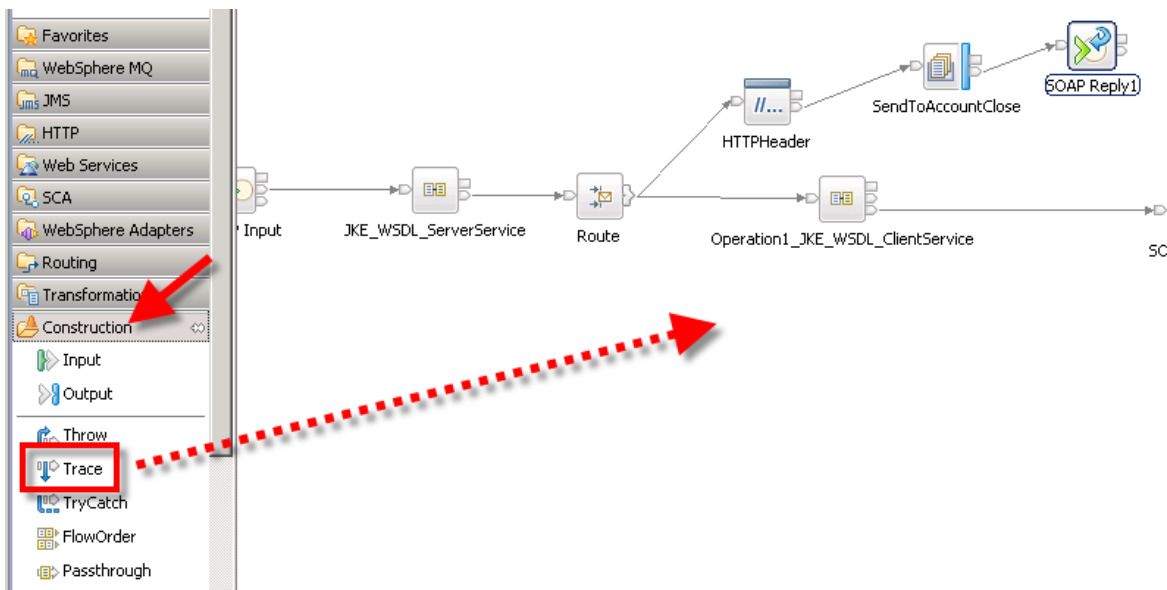
- __81. Anchor the connector to the **HTTPHeader** node, as shown above.



__82. Connect the **out** terminal of the **HTTPHeader** node to the **Send_to_AccountClose** node.



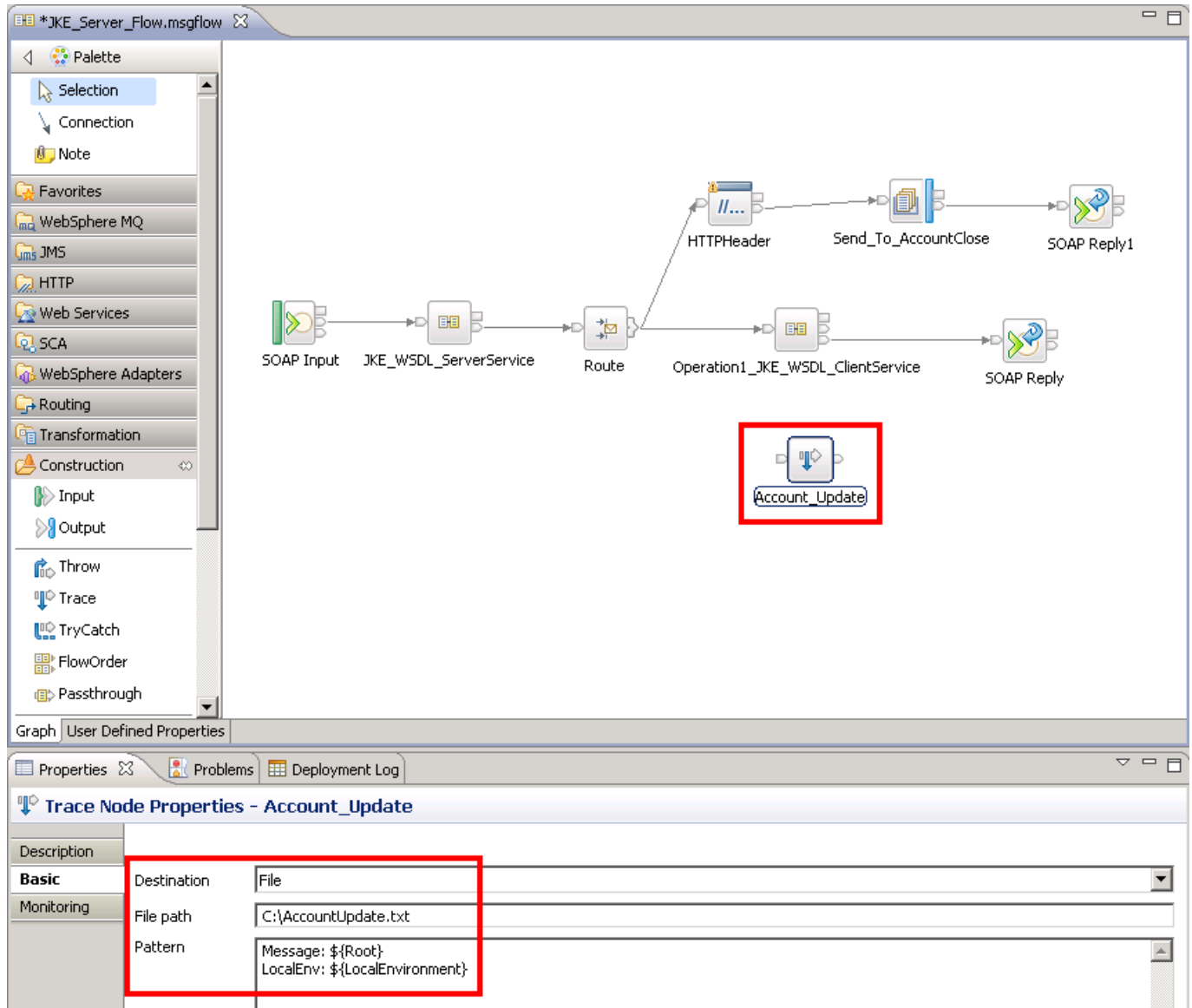
__83. Connect the **out** terminal of the **Send_to_AccountClose** node to the **SOAP Reply1** node.



A temporary path for Account_Update path will now be constructed.

__84. Open the **Construction** drawer.

__85. Drag and drop a **Trace** node as shown.

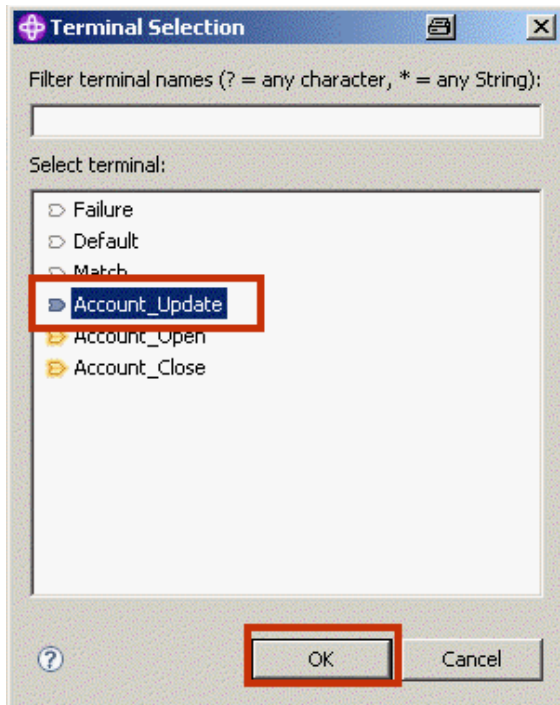


- __86. Rename the trace node to **AccountUpdate**.
- __87. In the Properties, use the Destination pull-down and select **File**.
- __88. Enter **c:\AccountUpdate.txt** for the file path.
- __89. Enter the following two lines for the Pattern:

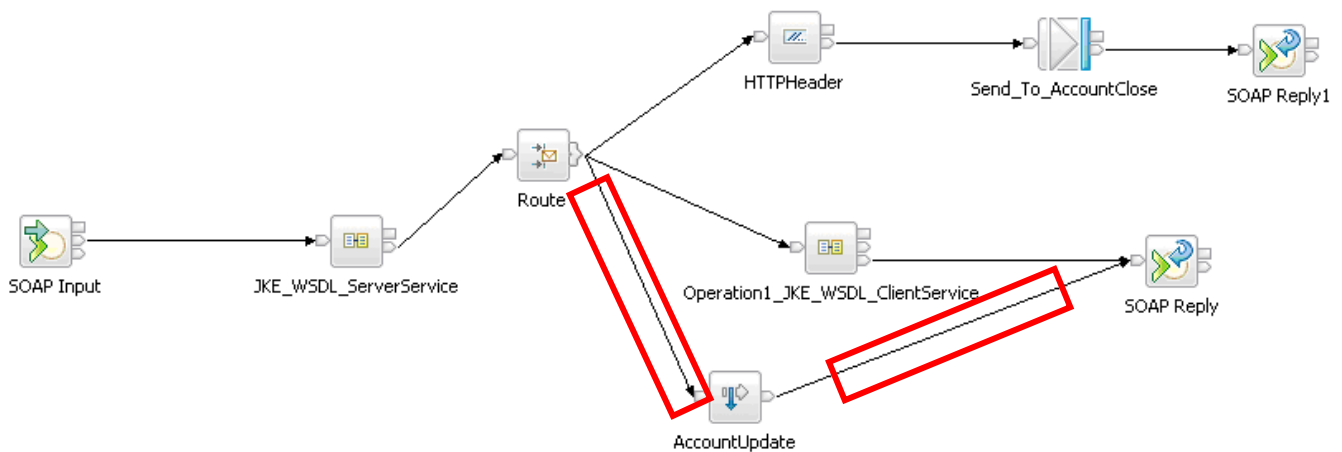
```

Message: ${Root}
LocalEnv: ${LocalEnvironment}
    
```

Be sure to type the info in the curly braces accurately including case!

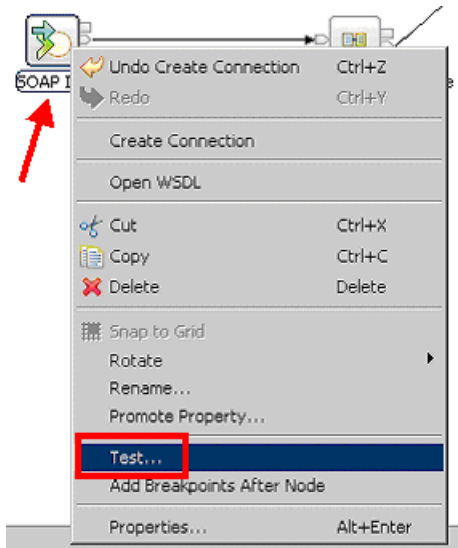


- __90. Click on the **Route** node output terminal cluster.
- __91. Select the **Account_Update** terminal.
- __92. Wire the terminal to the **In** terminal of the **AccountUpdate** trace node



- __93. Finally, connect the **AccountUpdate** node to the **SOAP Reply** node. All three paths have been completed for this lab. The three paths will now be tested.

- __94.  Save the message flow.

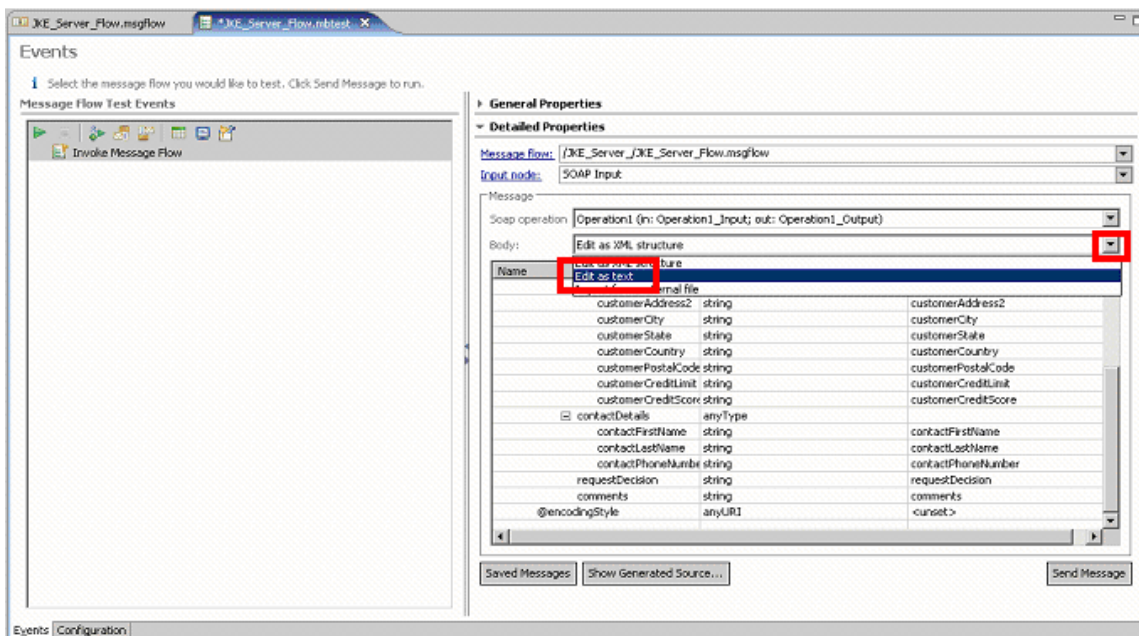


The testing process should be familiar by now.

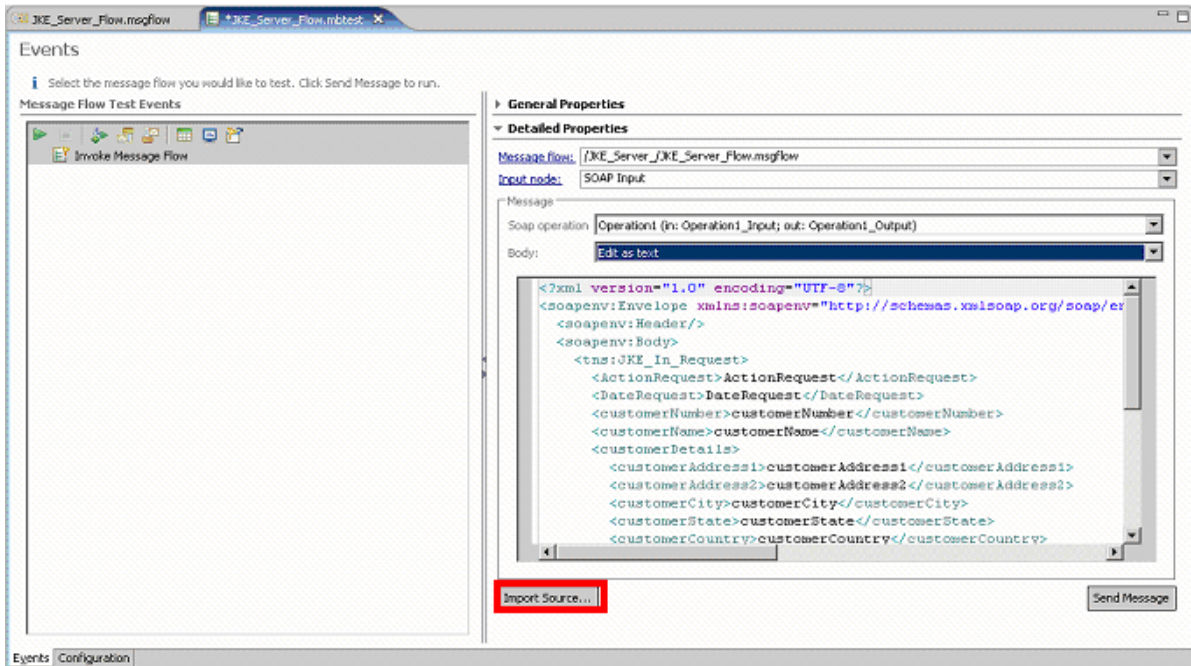
While the existing Test Client configuration could be used it cannot be saved with a different name. When this set of tests is complete the test configuration will be saved with a different name. It will contain three messages, one for each path. This configuration will be reused in the next two labs.

__95. Right-click on the **SOAP Input** node.

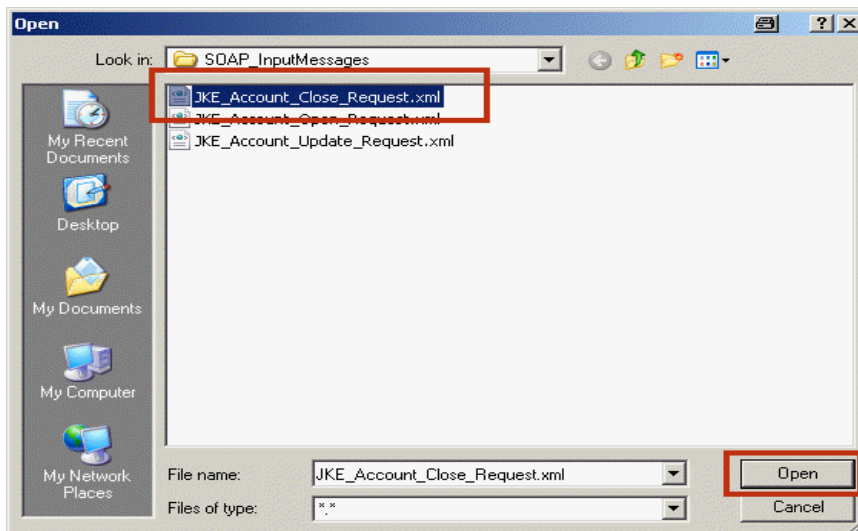
__96. Select **Test**.



__97. Use the **Body** pull-down to select **Edit as text** from the menu.

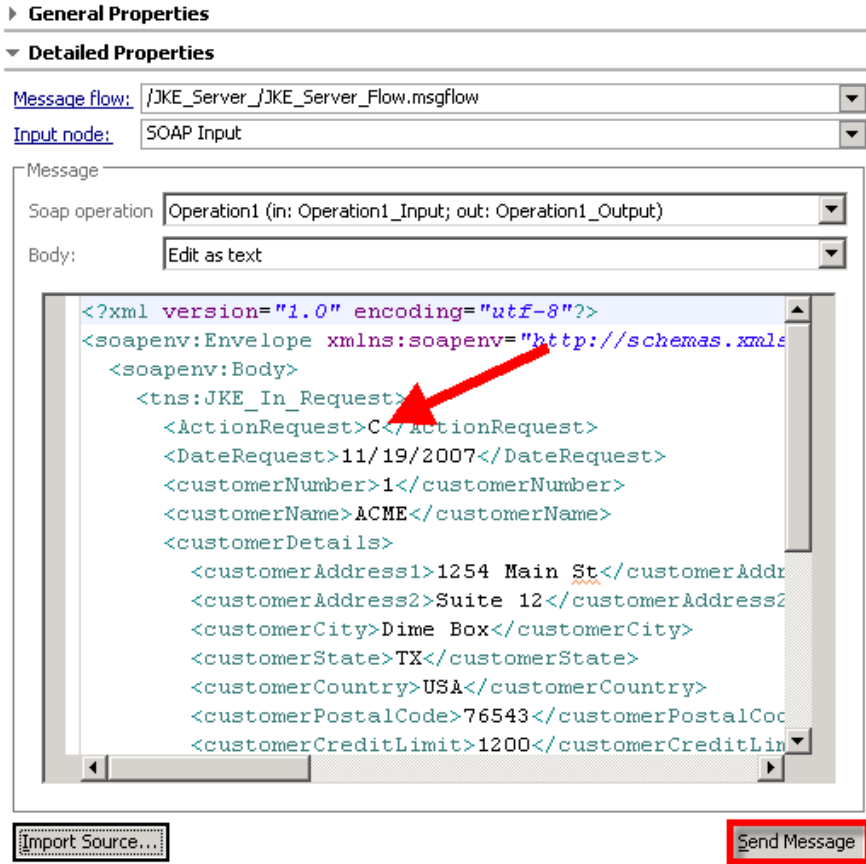


__98. Click the **Import Source** button.

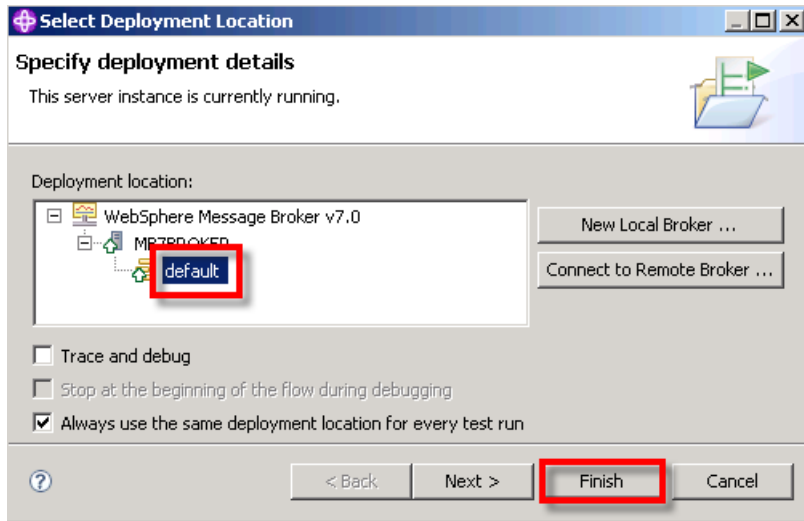


__99. Select **JKE_Account_Close_Request.xml**.

__100. Click **Open**.



__101. Click **Send Message**



The deployment information has not changed.

__102. Select the **default** execution group.

__103. Press the **Finish** button.

Events

The screenshot shows the 'Message Flow Test Events' pane on the left and the 'General Properties' pane on the right. In the events pane, the event 'MQ Queue Monitor "LAB.ACCOUNTCLOSE"' is selected and highlighted with a red box. A red arrow points from this event to the 'General Properties' pane. In the 'General Properties' pane, the 'Queue' field is also highlighted with a red box and contains the value 'LAB.ACCOUNTCLOSE'. Below the 'Queue' field, the 'Message' body is displayed as an XML structure table.

Name	Value
NS1:JKE_In_Request	
xmlns:NS1	http://www.ibm.lab.com
ActionRequest	C
DateRequest	11/19/2007
customerNumber	1
customerName	ACME
customerDetails	
customerAddress1	1254 Main St
customerAddress2	Suite 12
customerCity	Dime Box
customerState	TX
customerCountry	USA
customerPostalCode	76543
customerCreditLimit	1200
customerCreditScore	123
contactDetails	
contactFirstName	Freddy
contactLastName	Bloggs
contactPhoneNumber	555-123-6543

Note that there are two output events. One is from the SOAP reply and the other is from the LAB.ACCOUNTCLOSE queue, which is displayed using the Queue name as a label.

The screenshot shows the 'Message Flow Test Events' pane on the left with a context menu open over the 'MQ Queue Monitor "LAB.ACCOUNTCLOSE"' event. The 'Duplicate' option in the menu is highlighted with a red box. A red arrow points from the event to the menu. On the right, the 'General Properties' pane shows the 'Message' body as XML text.

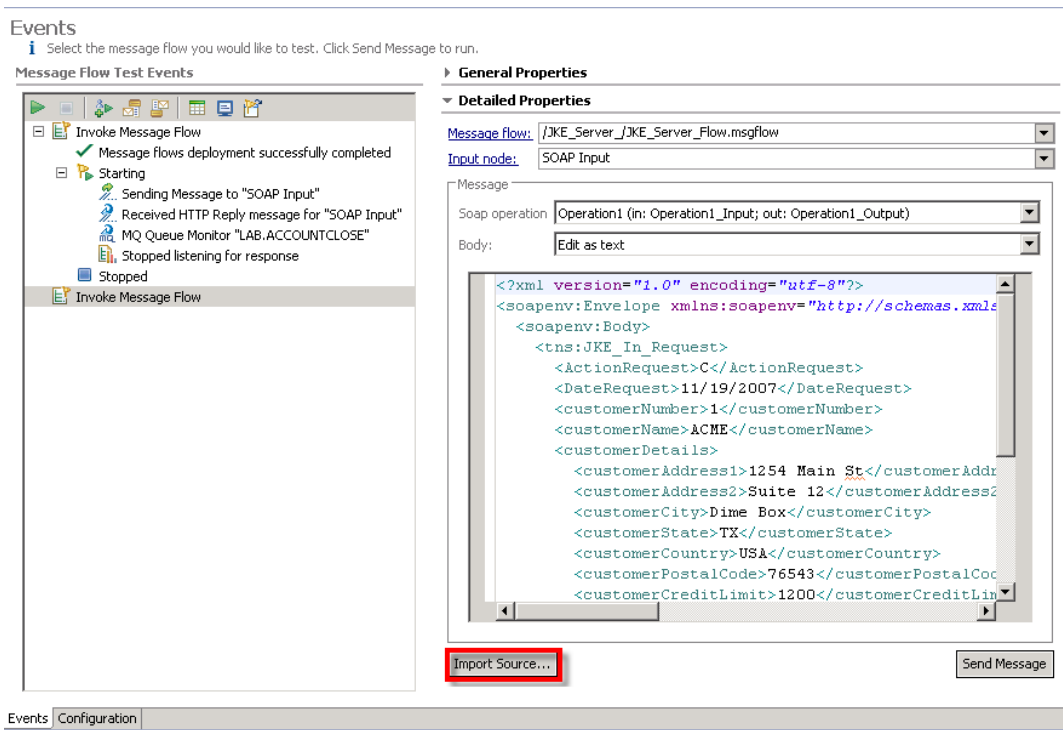
```
<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns1:JKE_In_Request>
      <ActionRequest>C</ActionRequest>
      <DateRequest>11/19/2007</DateRequest>
      <customerNumber>1</customerNumber>
      <customerName>ACME</customerName>
      <customerDetails>
        <customerAddress1>1254 Main St</customerAddress1>
        <customerAddress2>Suite 12</customerAddress2>
        <customerCity>Dime Box</customerCity>
        <customerState>TX</customerState>
        <customerCountry>USA</customerCountry>
        <customerPostalCode>76543</customerPostalCode>
      </customerDetails>
    </ns1:JKE_In_Request>
  </soapenv:Body>
</soapenv:Envelope>
```

__104. You now need to test the Account_Open path.

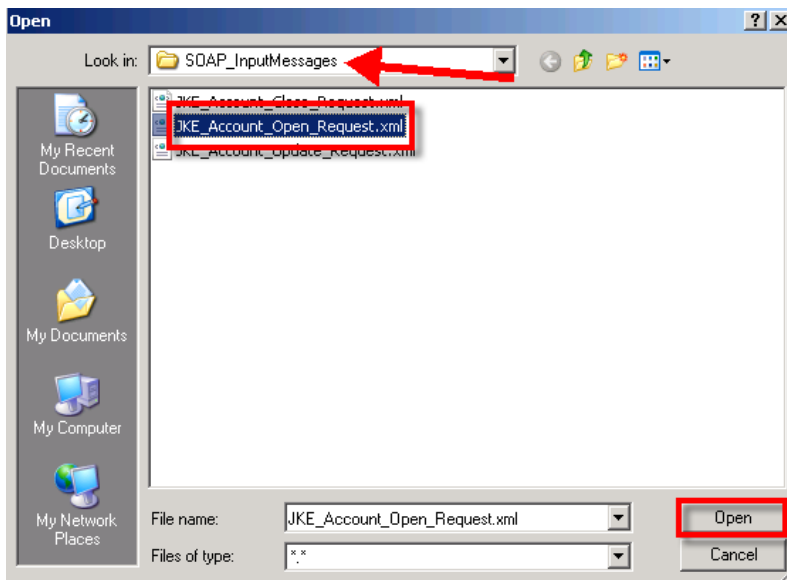
__105. Select **Invoke Message Flow**.

__106. Press the right mouse button.

__107. Select **Duplicate**.



__108. Click **Import Source**.



__109. Select **JKE_Account_Open_Request.xml**.

__110. Click **Open**.

Events

Select the message flow you would like to test. Click Send Message to run.

Message Flow Test Events

- Invoke Message Flow
 - Message flows deployment successfully completed
 - Starting
 - Sending Message to "SOAP Input"
 - Received HTTP Reply message for "SOAP Input"
 - MQ Queue Monitor "LAB.ACCOUNTCLOSE"
 - Stopped listening for response
 - Stopped
- Invoke Message Flow

General Properties

Detailed Properties

Message flow: /JKE_Server/JKE_Server_Flow.msgflow

Input node: SOAP Input

Message

Soap operation: Operation1 (in: Operation1_Input; out: Operation1_Output)

Body: Edit as text

```
<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <tns:JKE_In_Request>
      <ActionRequest>0</ActionRequest>
      <DateRequest>10/12/2007</DateRequest>
      <customerNumber></customerNumber>
      <customerName>ACME</customerName>
      <customerDetails>
        <customerAddress1>1254 Main St</customerAddress1>
        <customerAddress2>Suite 12</customerAddress2>
        <customerCity>Dime Box</customerCity>
        <customerState>TX</customerState>
        <customerCountry>USA</customerCountry>
        <customerPostalCode>76543</customerPostalCode>
        <customerCreditLimit>1200</customerCreditLimit>
      </customerDetails>
    </tns:JKE_In_Request>
  </soapenv:Body>
</soapenv:Envelope>
```

Import Source... Send Message

Events Configuration

111. Click **Send Message**.

Events

Message Flow Test Events

- Invoke Message Flow
 - Message flows deployment successfully completed
 - Starting
 - Sending Message to "SOAP Input"
 - Received HTTP Reply message for "SOAP Input"
 - MQ Queue Monitor "LAB.ACCOUNTCLOSE"
 - Stopped listening for response
 - Stopped
- Invoke Message Flow
 - Message flows deployment successfully completed
 - Starting
 - Sending Message to "SOAP Input"
 - Received HTTP Reply message for "SOAP Input"
 - Listening for response

General Properties

Detailed Properties

Endpoint URL: http://localhost:7800/JKE_Server

Message

Body: View as XML structure

Name	Value
soapenv:Envelope	http://schemas.xmlsoap.org/soap/envelope/
soapenv:Body	
N51:JKE_Out_Resp	
xmlns:N51	http://www.ibm.lab.com
ActionRequest	0
DateRequest	2009-10-23
customerNumber	4
customerName	ACME
customerDetails	
customerAd	1254 Main St
customerAd	Suite 12
customerCit	Dime Box
customerSt	TX
customerCo	USA
customerPo	76543
customerCr	1200
customerCr	130
contactDetails	
contactFirst	Freddy
contactLast	Bloggs
contactPhor	555-123-6543
requestDecision	Y
comments	Request Approved

Events Configuration

Since the message flow now has an MQOutput node, which is monitored by the Test Client for output, a Stopped line will not appear since the Account Open path does not write to this queue. Note that the Test Events pane indicates that the Test Client is "Listening for response". We configured the Test Client to time out in 40 seconds so if that period of time expires a Timeout message will appear and the Test will stop. The Stop icon in the upper left could also be used.

The screenshot shows the 'Events' pane on the left and the 'General Properties' pane on the right. The 'Message Flow Test Events' pane shows a sequence of events for two 'Invoke Message Flow' operations. The second operation is currently in the 'Stopped' state, indicated by a red arrow pointing to the 'Stopped' icon. The 'General Properties' pane shows the 'Detailed Properties' for the selected message flow, including the endpoint URL 'http://localhost:7800/JKE_Server' and the message body in XML format.

Name	Value
soapenv:Envelope	http://schemas.xmlsoap.org/soap/envelope/
soapenv:Body	
N51:JKE_Out_Respi	
xmlns:N51	http://www.ibm.lab.com
ActionRequest	O
DateRequest	2009-10-23
customerNumber	4
customerName	ACME
customerDetails	
customerAd	1254 Main St
customerAd	Suite 12
customerCit	Dime Box
customerSts	TX
customerCo	USA
customerPo	76543
customerCr	1200
customerCr	130
contactDetails	
contactFirst	Freddy
contactLast	Bloggs
contactPhor	555-123-6543
requestDecision	Y
comments	Request Approved

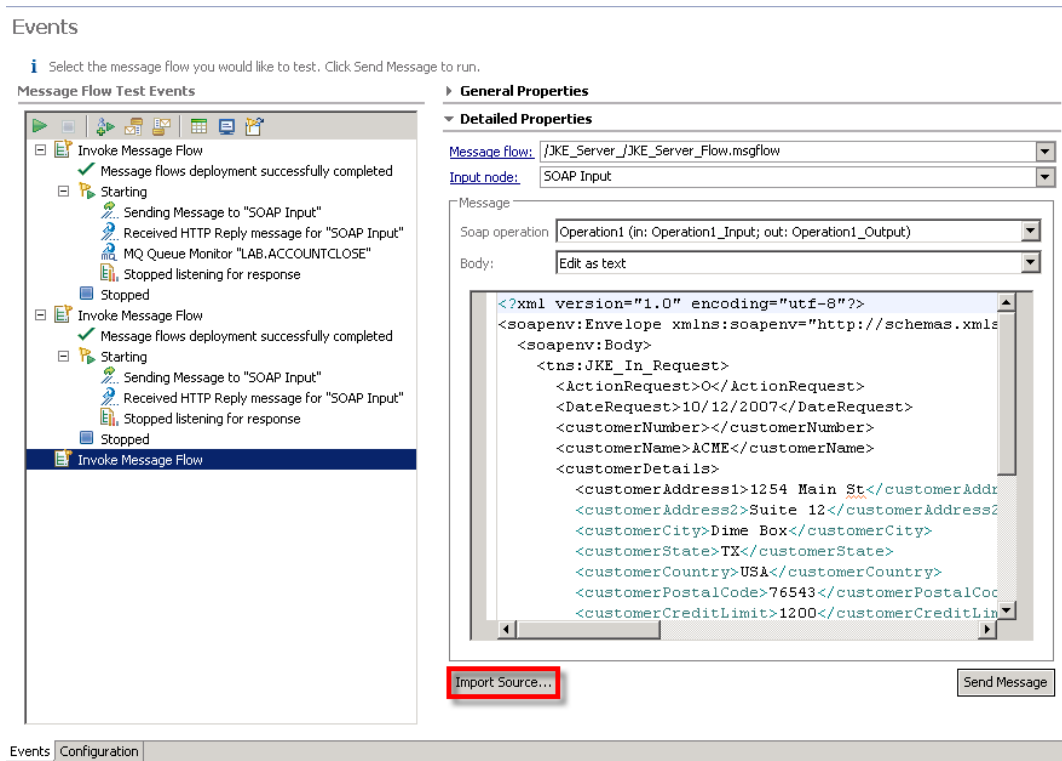
__112. Make sure the test has stopped either via a timeout or by using the Stop icon.

This screenshot is similar to the previous one, but with a context menu open over the 'Stopped' icon in the 'Message Flow Test Events' pane. The 'Duplicate' option is highlighted with a red box. The 'General Properties' pane on the right shows the 'Detailed Properties' for the selected message flow, including the message flow name 'JKE_Server_JKE_Server_Flow.msgflow' and the input node 'SOAP Input'. The message body is shown in XML format.

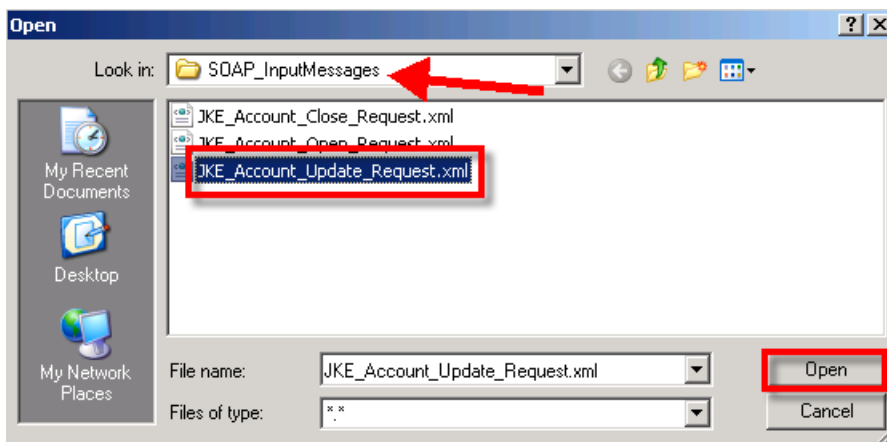
```
<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <tns:JKE_In_Request>
      <ActionRequest>O</ActionRequest>
      <DateRequest>10/12/2007</DateRequest>
      <customerNumber></customerNumber>
      <customerName>ACME</customerName>
      <customerDetails>
        <customerAddress1>1254 Main St</customerAddress1>
        <customerAddress2>Suite 12</customerAddress2>
        <customerCity>Dime Box</customerCity>
        <customerState>TX</customerState>
        <customerCountry>USA</customerCountry>
        <customerPostalCode>76543</customerPostalCode>
        <customerCreditLimit>1200</customerCreditLimit>
      </customerDetails>
    </tns:JKE_In_Request>
  </soapenv:Body>
</soapenv:Envelope>
```

You now need to test the Account_Update path.

- ___113. Select **Invoke Message Flow**
- ___114. Press the right mouse button.
- ___115. Select **Duplicate** from the menu.



- ___116. Click **Import Source**.



- ___117. Select the **JKE_Account_Update_Request.xml** file.
- ___118. Click **Open**.

Events

Select the message flow you would like to test. Click Send Message to run.

Message Flow Test Events

- Invoke Message Flow
 - Message flows deployment successfully completed
 - Starting
 - Sending Message to "SOAP Input"
 - Received HTTP Reply message for "SOAP Input"
 - MQ Queue Monitor "LAB.ACCOUNTCLOSE"
 - Stopped listening for response
 - Stopped
- Invoke Message Flow
 - Message flows deployment successfully completed
 - Starting
 - Sending Message to "SOAP Input"
 - Received HTTP Reply message for "SOAP Input"
 - Stopped listening for response
 - Stopped
- Invoke Message Flow
 - Message flows deployment successfully completed
 - Starting
 - Sending Message to "SOAP Input"
 - Received HTTP Reply message for "SOAP Input"
 - Stopped listening for response
 - Stopped
- Invoke Message Flow
 - Message flows deployment successfully completed
 - Starting
 - Sending Message to "SOAP Input"
 - Received HTTP Reply message for "SOAP Input"
 - Stopped listening for response
 - Stopped

General Properties

Detailed Properties

Message flow: /JKE_Server/JKE_Server_Flow.msgflow

Input node: SOAP Input

Message

Soap operation: Operation1 (in: Operation1_Input; out: Operation1_Output)

Body: Edt as text

```
<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <tns:JKE_In_Request>
      <ActionRequest>U</ActionRequest>
      <DateRequest>10/12/2007</DateRequest>
      <customerNumber>1</customerNumber>
      <customerName>ACME</customerName>
      <customerDetails>
        <customerAddress1>1254 Main St</customerAddress1>
        <customerAddress2>Suite 12</customerAddress2>
        <customerCity>Dime Box</customerCity>
        <customerState>TX</customerState>
        <customerCountry>USA</customerCountry>
        <customerPostalCode>76543</customerPostalCode>
        <customerCreditLimit>1200</customerCreditLimit>
      </customerDetails>
    </tns:JKE_In_Request>
  </soapenv:Body>
</soapenv:Envelope>
```

Import Source... Send Message

119. Click **Send Message**

Events

Message Flow Test Events

- Invoke Message Flow
 - Message flows deployment successfully completed
 - Starting
 - Sending Message to "SOAP Input"
 - Received HTTP Reply message for "SOAP Input"
 - MQ Queue Monitor "LAB.ACCOUNTCLOSE"
 - Stopped listening for response
 - Stopped
- Invoke Message Flow
 - Message flows deployment successfully completed
 - Starting
 - Sending Message to "SOAP Input"
 - Received HTTP Reply message for "SOAP Input"
 - Stopped listening for response
 - Stopped
- Invoke Message Flow
 - Message flows deployment successfully completed
 - Starting
 - Sending Message to "SOAP Input"
 - Received HTTP Reply message for "SOAP Input"
 - Stopped listening for response
 - Stopped
- Invoke Message Flow
 - Message flows deployment successfully completed
 - Starting
 - Sending Message to "SOAP Input"
 - Received HTTP Reply message for "SOAP Input"
 - Listening for response
 - Stopped

General Properties

Detailed Properties

Endpoint URL: http://localhost:7800/JKE_Server

Message

Body: View as XML structure

Name	Value
soapenv:Envelope	
xmlns:soapenv	http://schemas.xmlsoap.org/soap/envelope/
soapenv:Body	
NS1:JKE_In_Request	
xmlns:NS1	http://www.ibm.lab.com
ActionRequest	U
DateRequest	10/12/2007
customerNumber	1
customerName	ACME
customerDetails	
customerAd	1254 Main St
customerAd	Suite 12
customerCity	Dime Box
customerState	TX
customerCountry	USA
customerPostalCode	76543
customerCreditLimit	1200
customerCreditLimit	123
contactDetails	
contactFirst	Freddy
contactLast	Bloggs
contactPhone	555-123-6543
requestDecision	Y
comments	Just a Comment

Again, a Stopped line is not shown. The Test Client is monitoring both the WebSphere MQ queue and the Web service response. Since the Account_Update path does not use the queue, the Test Client will continue to listen for a response until the 40 second timeout occurs or the test is manually halted using the Stop icon in the upper left.

Events

Message Flow Test Events

Invoke Message Flow

- Message flows deployment successfully completed
- Starting
 - Sending Message to "SOAP Input"
 - Received HTTP Reply message for "SOAP Input"
 - MQ Queue Monitor "LAB.ACCOUNTCLOSE"
 - Stopped listening for response
- Stopped

Invoke Message Flow

- Message flows deployment successfully completed
- Starting
 - Sending Message to "SOAP Input"
 - Received HTTP Reply message for "SOAP Input"
 - Stopped listening for response
- Stopped

Invoke Message Flow

- Message flows deployment successfully completed
- Starting
 - Sending Message to "SOAP Input"
 - Received HTTP Reply message for "SOAP Input"
 - Stopped listening for response
- Stopped

General Properties

Detailed Properties

Endpoint URL: `http://localhost:7800/JKE_Server`

Message

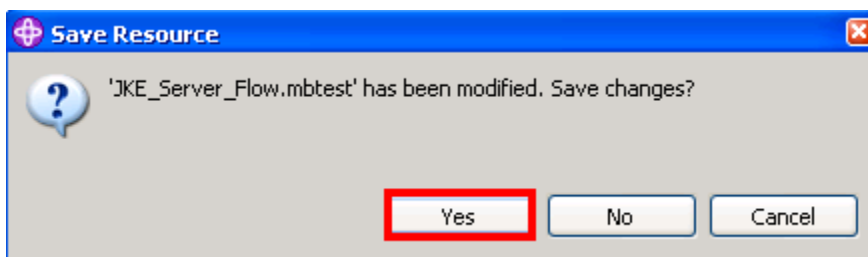
Body: `View as XML structure`

Name	Value
soapenv:Envelope	
xmlns:soapenv	http://schemas.xmlsoap.org/soap/envelope/
soapenv:Body	
NS1:JKE_In_Request	
xmlns:NS1	http://www.ibm.lab.com
ActionRequest	U
DateRequest	10/12/2007
customerNumber	1
customerName	ACME
customerDetails	
customerAd	1254 Main St
customerAd	Suite 12
customerCit	Dime Box
customerSt	TX
customerCo	USA
customerPo	76543
customerCr	1200
customerCr	123
contactDetails	
contactFirst	Freddy
contactLast	Bloggs
contactPhor	555-123-6543
requestDecision	Y
comments	Just a Comment

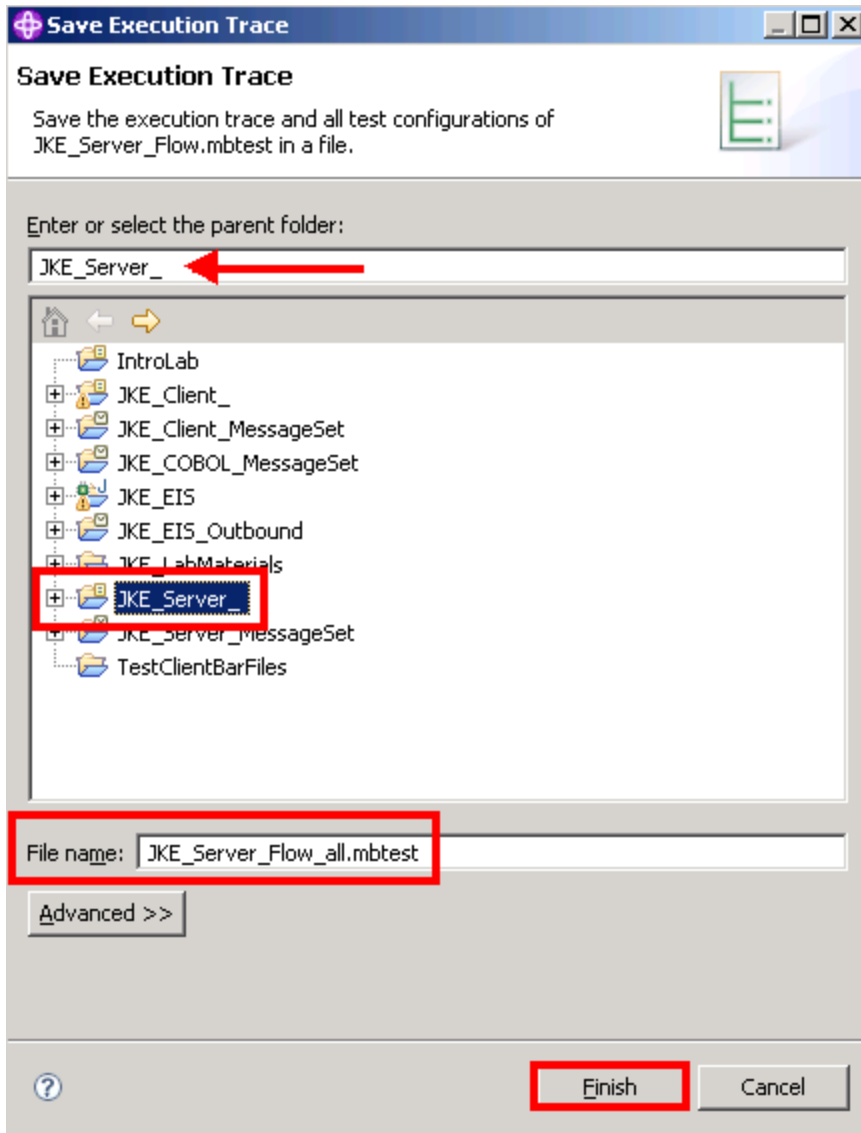
Events Configuration

__120. Make sure that the test has stopped, either via a timeout or from using the Stop icon. This test will be used again in later labs so the configuration will be saved under a new name.

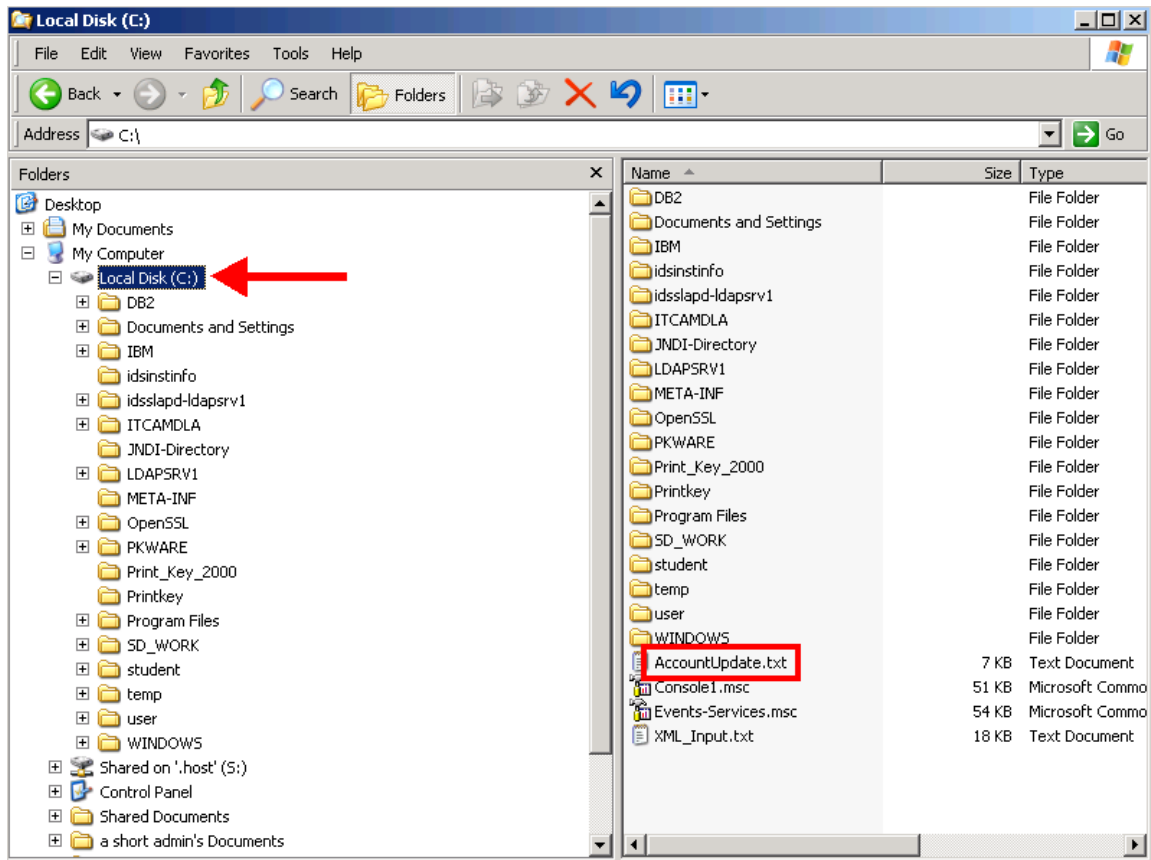
__121. Close the Test Client.



__122. Click **Yes** to save the test client configuration and use cases.



- __123. Highlight the **JKE_Server_** entry.
- __124. Add “**_All**” to the File name as shown above.
- __125. Click **Finish** to save the test.



__126. Open the **Windows Explorer** using the icon in the Quick Launch area.

__127. Double click the **AccountUpdate.txt** file to open it.

```

AccountUpdate.txt - Notepad
File Edit Format View Help
Message: ( ['SOAPROOT' : 0x77388c0]
(0x01000000:Name ):Properties = ( ['SOAPPROPERTYPARSER' : 0x32477d0]
(0x03000000:Namevalue):MessageSet = 'JKE_Server_MessageSet' (CHARACTER)
(0x03000000:Namevalue):MessageType = '' (CHARACTER)
(0x03000000:Namevalue):MessageFormat = '' (CHARACTER)
(0x03000000:Namevalue):Encoding = 546 (INTEGER)
(0x03000000:Namevalue):CodedCharSetId = 1208 (INTEGER)
(0x03000000:Namevalue):Transactional = FALSE (BOOLEAN)
(0x03000000:Namevalue):Persistence = FALSE (BOOLEAN)
(0x03000000:Namevalue):creationTime = GMTTIMESTAMP '2009-01-20 15:41:00.486' (GMTTIMESTAMP)
(0x03000000:Namevalue):ExpirationTime = -1 (INTEGER)
(0x03000000:Namevalue):Priority = 0 (INTEGER)
(0x03000000:Namevalue):ReplyIdentifier = x'0000000000000000000000000000000000000000000000000000000000000000' (BLOB)
(0x03000000:Namevalue):ReplyProtocol = 'SOAP-Axis2' (CHARACTER)
(0x03000000:Namevalue):Topic = NULL
(0x03000000:Namevalue):contentType = 'text/xml; charset=utf-8' (CHARACTER)
(0x03000000:Namevalue):IdentitySourceType = '' (CHARACTER)
(0x03000000:Namevalue):IdentitySourceToken = '' (CHARACTER)
(0x03000000:Namevalue):IdentitySourcePassword = '' (CHARACTER)
(0x03000000:Namevalue):IdentitySourceIssuedBy = '' (CHARACTER)
(0x03000000:Namevalue):IdentityMappedType = '' (CHARACTER)
(0x03000000:Namevalue):IdentityMappedToken = '' (CHARACTER)
(0x03000000:Namevalue):IdentityMappedPassword = '' (CHARACTER)
(0x03000000:Namevalue):IdentityMappedIssuedBy = '' (CHARACTER)
(0x01000000:Name ):HTTPInputHeader = ( ['WSINPHDR' : 0x77386b8]
(0x03000000:Namevalue):X-Original-HTTP-Command = 'POST http://localhost:7800/JKE_Server HTTP/1.1' (CHARACTER)
(0x03000000:Namevalue):SOAPAction = 'AccountOpen' (CHARACTER)
(0x03000000:Namevalue):Content-Type = 'text/xml; charset=utf-8' (CHARACTER)
(0x03000000:Namevalue):User-Agent = 'Java/1.5.0' (CHARACTER)
(0x03000000:Namevalue):Host = 'localhost:7800' (CHARACTER)
(0x03000000:Namevalue):Accept = 'text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2' (CHARACTER)
(0x03000000:Namevalue):Connection = 'keep-alive' (CHARACTER)
(0x03000000:Namevalue):Content-Length = 1262 (CHARACTER)
(0x03000000:Namevalue):X-Remote-Addr = '127.0.0.1' (CHARACTER)
(0x03000000:Namevalue):X-Remote-Host = '127.0.0.1' (CHARACTER)
(0x03000000:Namevalue):X-Server-Name = 'localhost' (CHARACTER)
(0x03000000:Namevalue):X-Server-Port = '7800' (CHARACTER)
(0x03000000:Namevalue):X-Query-String = '' (CHARACTER)
(0x01000000:Folder):XMLNSC = ( ['xmlnsc' : 0x77249b8]
(0x01000000:Folder)http://www.ibm.lab.com:JKE_In_Request = ( ['xmlnsc' : 0x7724790]
(0x03000000:PCdataField):ActionRequest = 'u' (CHARACTER)
(0x03000000:PCdataField):DateRequest = '10/12/2007' (CHARACTER)
(0x03000000:PCdataField):customerNumber = '1' (CHARACTER)
(0x03000000:PCdataField):customerName = 'ACME' (CHARACTER)
(0x01000000:Folder ):customerDetails = (
(0x03000000:PCdataField):customerAddress1 = '1254 Main St' (CHARACTER)
(0x03000000:PCdataField):customerAddress2 = 'Suite 12' (CHARACTER)
(0x03000000:PCdataField):customerCity = 'Dime Box' (CHARACTER)
(0x03000000:PCdataField):customerState = 'TX' (CHARACTER)
(0x03000000:PCdataField):customerCountry = 'USA' (CHARACTER)
(0x03000000:PCdataField):customerPostalCode = '76543' (CHARACTER)
(0x03000000:PCdataField):customerCreditLimit = '1200' (CHARACTER)
(0x03000000:PCdataField):customerCreditScore = '123' (CHARACTER)
)
(0x01000000:Folder ):contactDetails = (
(0x03000000:PCdataField):contactFirstName = 'Freddy' (CHARACTER)
(0x03000000:PCdataField):contactLastName = 'Bloggs' (CHARACTER)
(0x03000000:PCdataField):contactPhoneNumber = '555-123-6543' (CHARACTER)
)
(0x03000000:PCdataField):requestDecision = 'Y' (CHARACTER)
(0x03000000:PCdataField):comments = 'Just a Comment' (CHARACTER)
)
)
)
LocalEnv:( ['MQROOT' : 0x7723650]
(0x01000000:Name):destination = (
(0x01000000:Name):SOAP = (
(0x01000000:Name):Reply = (
(0x03000000:Namevalue):ReplyIdentifier = x'534f415000000000030000000000000000a80e000000000000' (BLOB)
)
)
)
(0x01000000:Name):RouterList = (

```

__128. Scroll down to the HTTPInput header entry. This is the information that is available to the message flow when the SOAP Input node is used.

__129. Close the Notepad window.

__130. Minimize the Windows Explorer session.

You have completed Lab 7.

Lab 8 Building the JKE Server - Mapping

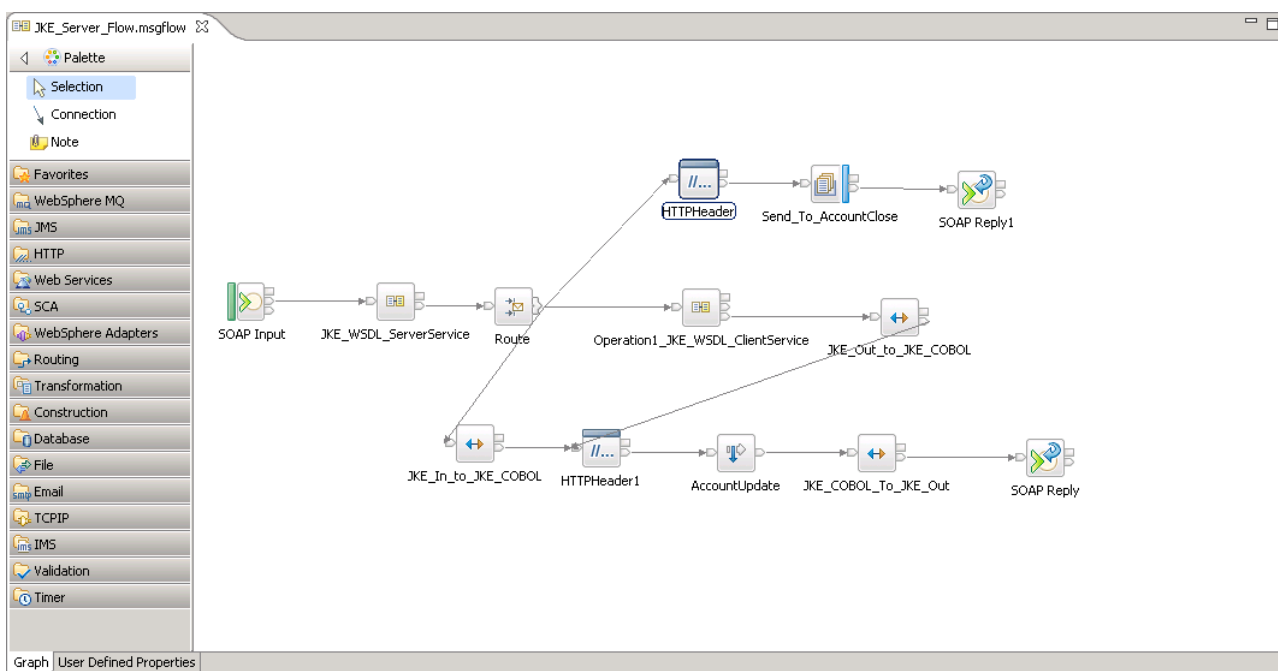
8.1 Overview

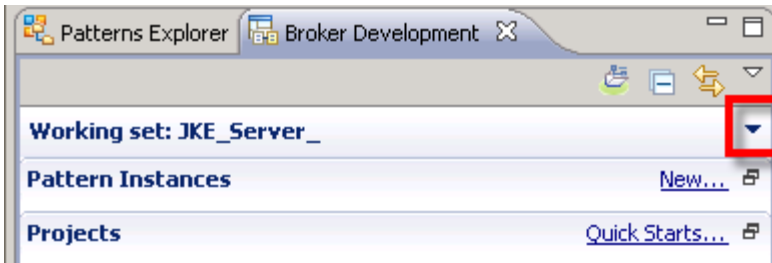
Before the message flow can send a request to CICS it must be transformed from XML to COBOL. Once a response has been received from CICS, it must then be transformed back into XML and placed in a SOAP Envelope and returned to the requesting client application.

The Account_Open Web service will return a Web service response, defined as JKE_Out_Response. This will also need to be transformed to COBOL. The TwineBall EIS application holds the master account information. In addition the CICS system also has a copy of the Account information as well as additional information used for other purposes. This is why an Account_Open request takes both paths.

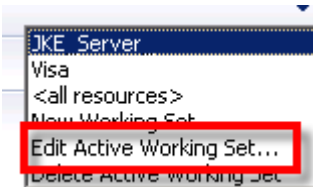
The Account_Update path has been designed to support changes to other systems that do not involve the EIS system.

In this lab you will be building the transformations using the graphical Mapping node. You will also use another HTTPHeader node to remove two types of HTTP headers.

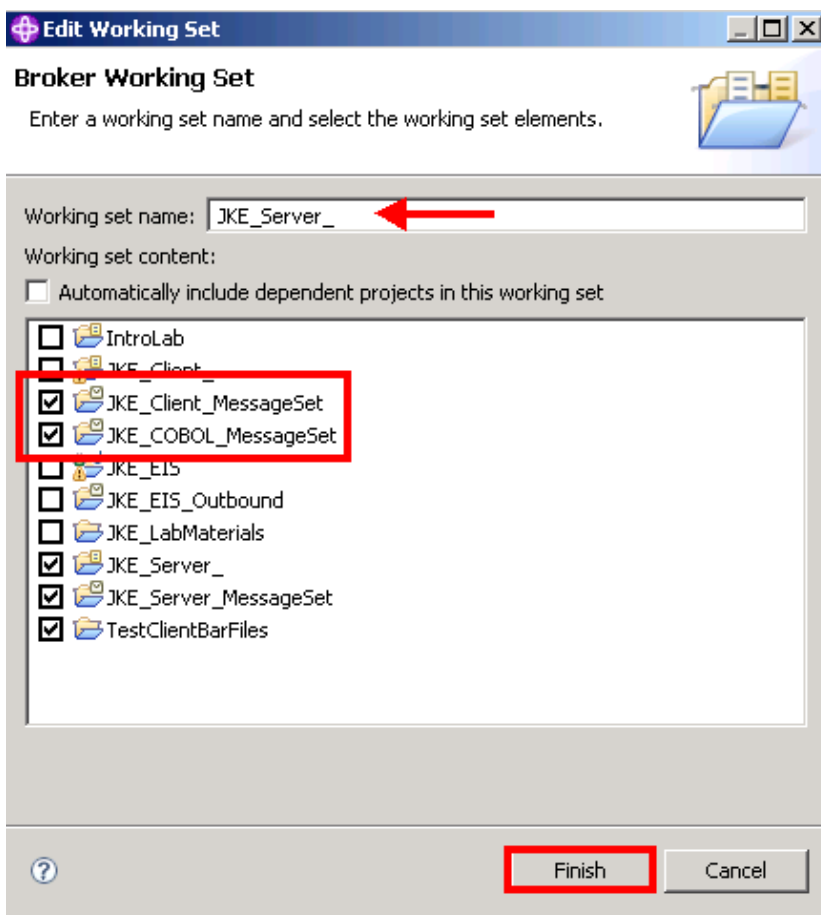




__1. Select the small triangle in the upper right corner of the **JKE_Server_** working set.

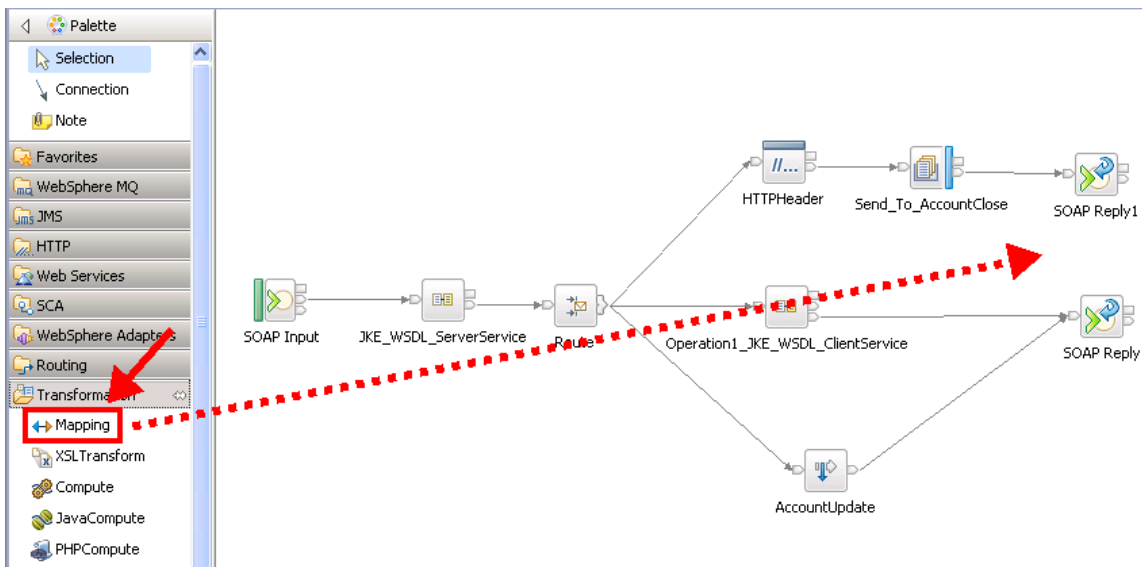


__2. Select the **Edit Active Working Set** option from the menu.

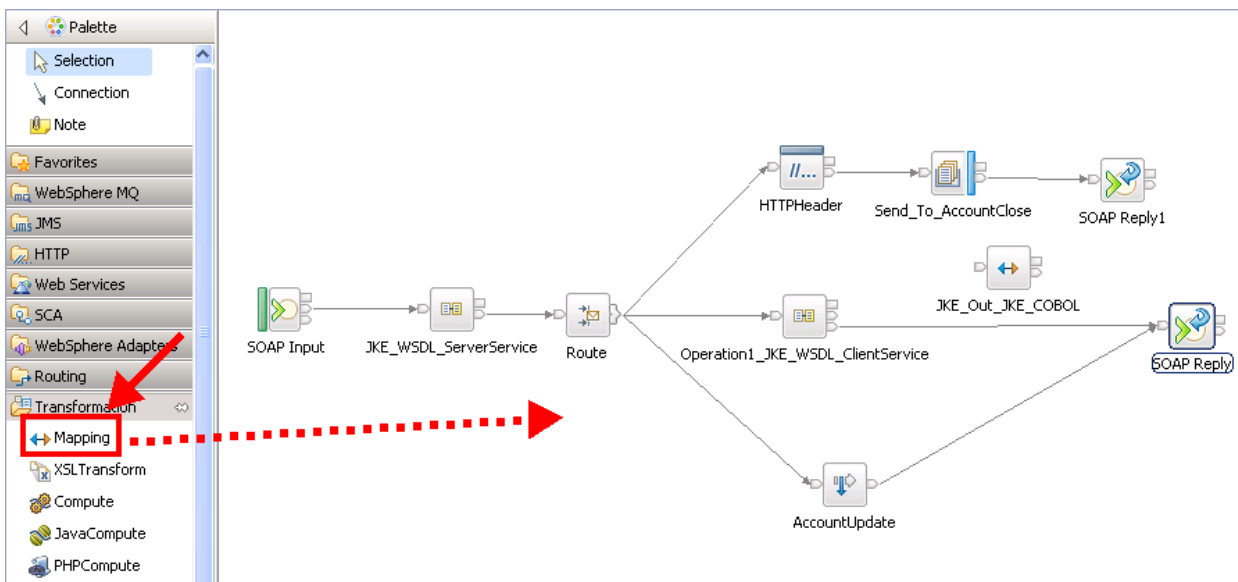


__3. Select the **JKE_COBOL_MessageSet** and **JKE_Client_MessageSet** check boxes.

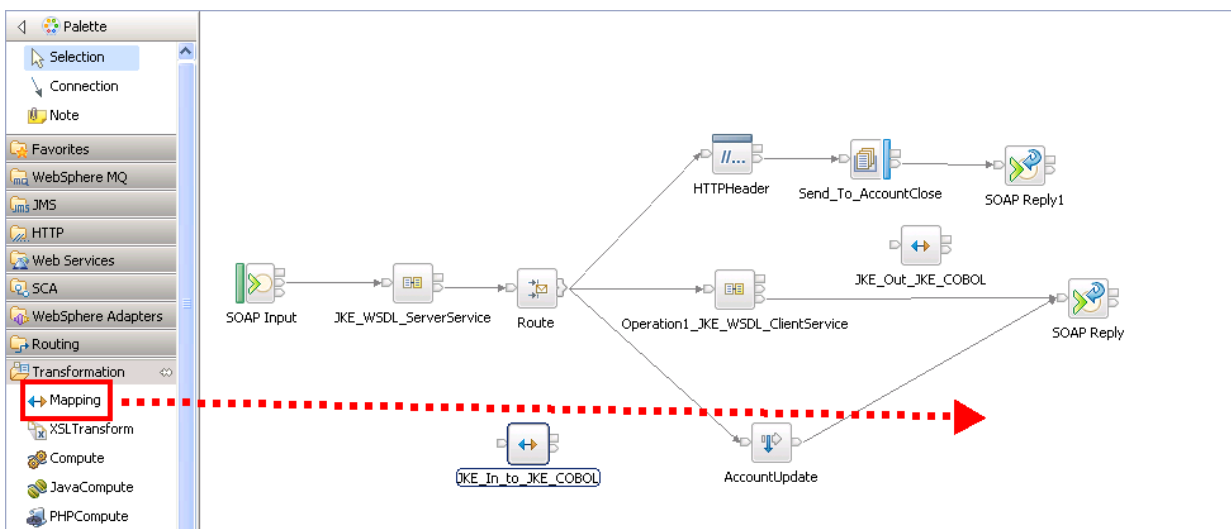
__4. Press the **Finish** button to add the projects to the working set.



- __5. Open the **Transformation** drawer.
- __6. Drag a **Mapping** node to the upper right side, just above the connector line.
- __7. Rename the Mapping node to **JKE_Out_to_JKE_COBOL**. The purpose of this node is to transform the SOAP response received from the JKE_Client Web service to a COBOL structure that will be sent to CICS in Lab 9.

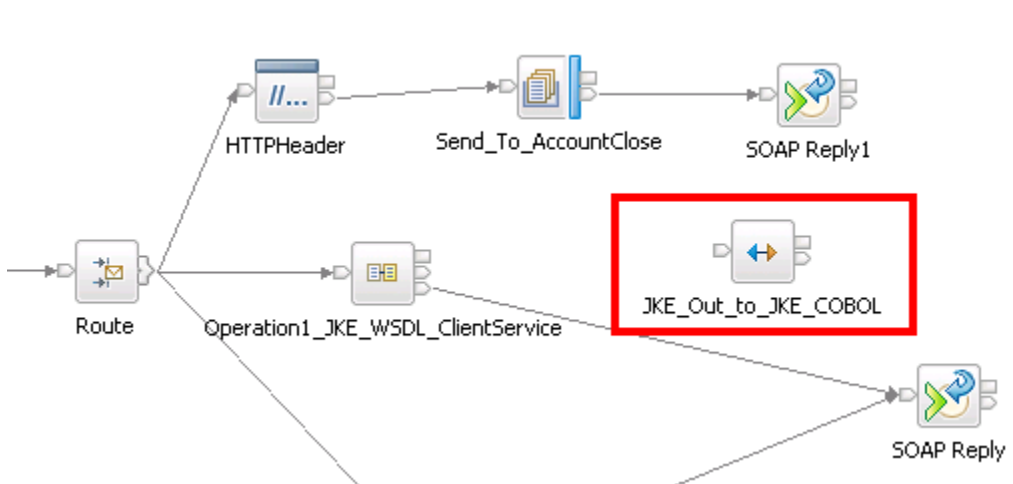


- __8. Drag another **Mapping** node to the lower left side as shown.
- __9. Rename the Mapping node to **JKE_In_to_JKE_COBOL**. The purpose of this node is to transform the JKE_In_Request received from the Web service client that invokes this flow to a COBOL structure that will be sent to CICS in Lab 9.

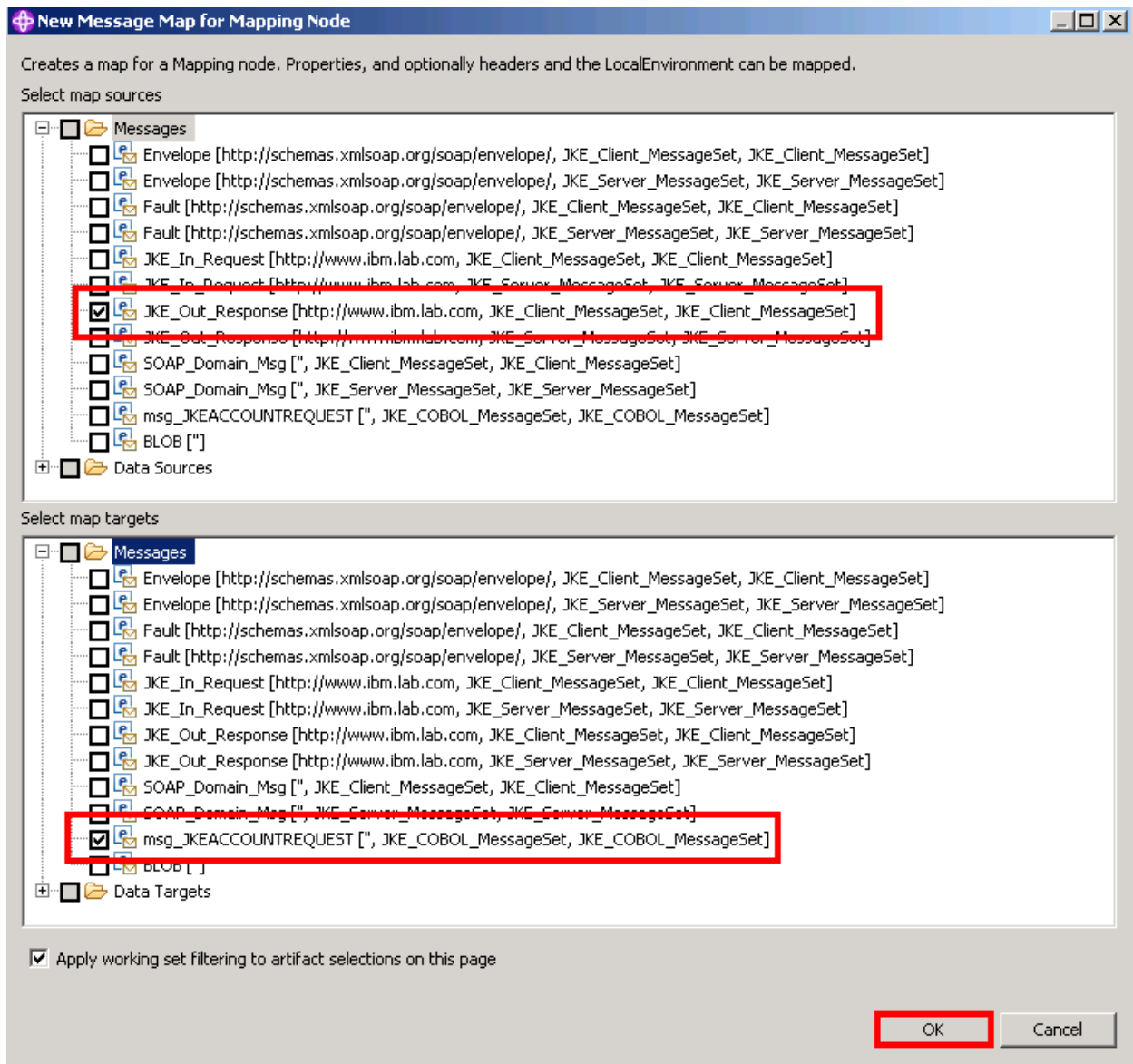


- __10. Drag a third **Mapping** node to the lower right as shown.
- __11. Rename the node to **JKE_COBOL_to_JKE_Out**. This node will transform the response from CICS from COBOL to XML so it can be returned to the Web service caller.

In the next series of steps, you will create maps for the three mapping nodes. The steps are the same, but the results will be different based on the source and target structures. You will define these maps in the same order as the nodes were added to the message flow. The Mapping wizard is started by double-clicking on a Mapping node.

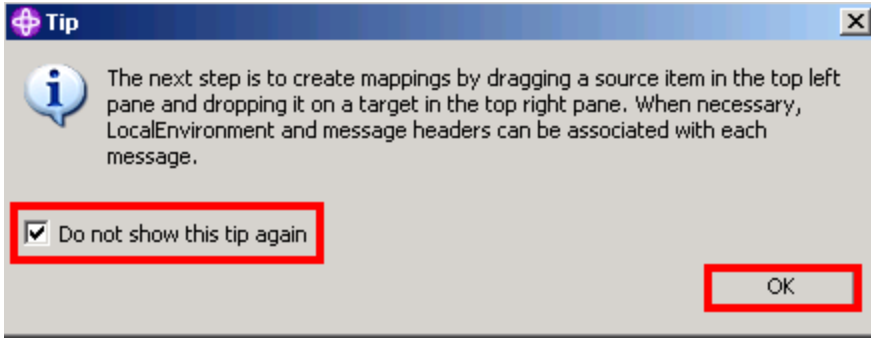


- __12. **Double-click** the **JKE_Out_to_JKE_COBOL** node to start the wizard.

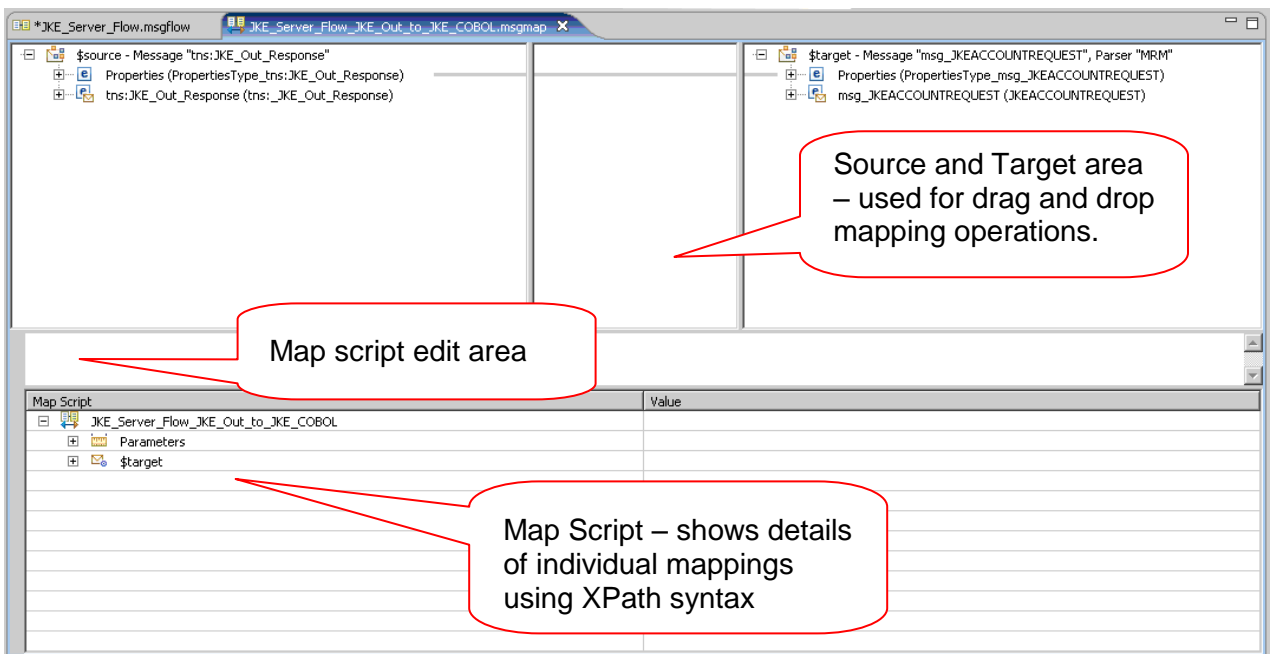


A panel is displayed where you identify the source and targets for the mapping operation. It can be expanded until all items are shown or scrolling can be used to see the message definitions.

- ___13. Click the check box for **JKE_Out_Response[JKE_Client_MessageSet]** for the map **source**.
- ___14. Click the check box for **msg_JKEACCOUNTREQUEST[JKE_COBOL_MessageSet]** for the map **target**.
- ___15. Make sure you have selected the correct items. There are **JKE_Out_Response** items for both the Client and Server message sets. The **Client** version should be used in this map.
- ___16. Click **OK**.



- __17. A tip box is displayed. You may want to click the check box for the **Do not show this tip again**.
- __18. Click **OK**.



The Mapping Panel is divided into three areas. The top area is used for drag and drop mapping. The middle area is used when editing a particular entry. The lower area contains the XPath statements for each mapping of source to target where the target is on the left side. The source information can also be edited in place.

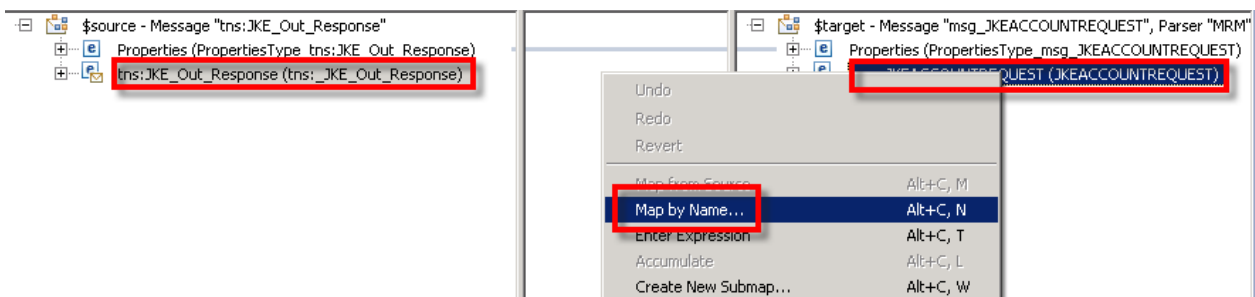
Note that the Properties folder has already been mapped. You can tell this because of the line connecting the Source and Target representations of that folder.

Map Script	value
Properties	
MessageSet	"JKE_COBOL_MessageSet"
MessageType	"{}:msg_JKEACCOUNTREQUEST"
MessageFormat	"Binary1"
Encoding	\$source/Properties/Encoding
CodedCharSetId	\$source/Properties/CodedCharSetId
Transactional	\$source/Properties/Transactional
Persistence	\$source/Properties/Persistence

__19. Expand the **\$target** and **Properties** folders in the Map Script area.

The individual mappings that have been done for the Properties folder can now be viewed. Note that the Mapping node has filled in the information about the target message set for you....everything has been set up to serialize this message into a COBOL bit stream when an output node is encountered. It is possible at this point to also make other changes in the Properties folder. For example, if you were sending this message to a real CICS running on IBM z/OS®, you could set the Encoding and CodedCharSetId fields to match what is needed by CICS and remove any requirement for code set conversions on the message by the transport or the receiver.

__20. Collapse the **Properties** folder.

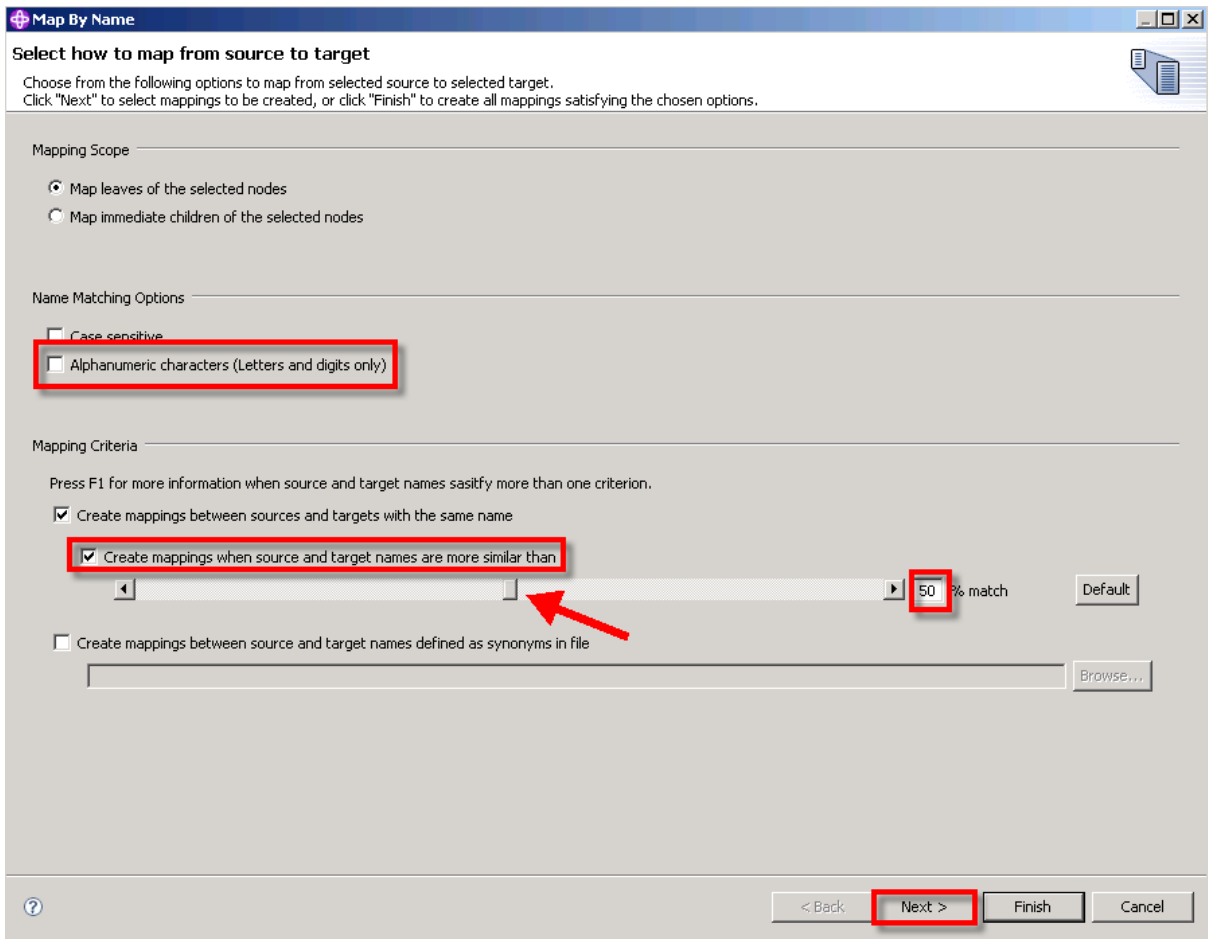


__21. Highlight the **tns:JKE_Out_Response** in the **source**.

__22. Right-click on the **msg_JKEACCOUNTREQUEST** for the **target**.

__23. Select **Map by Name** from the menu.

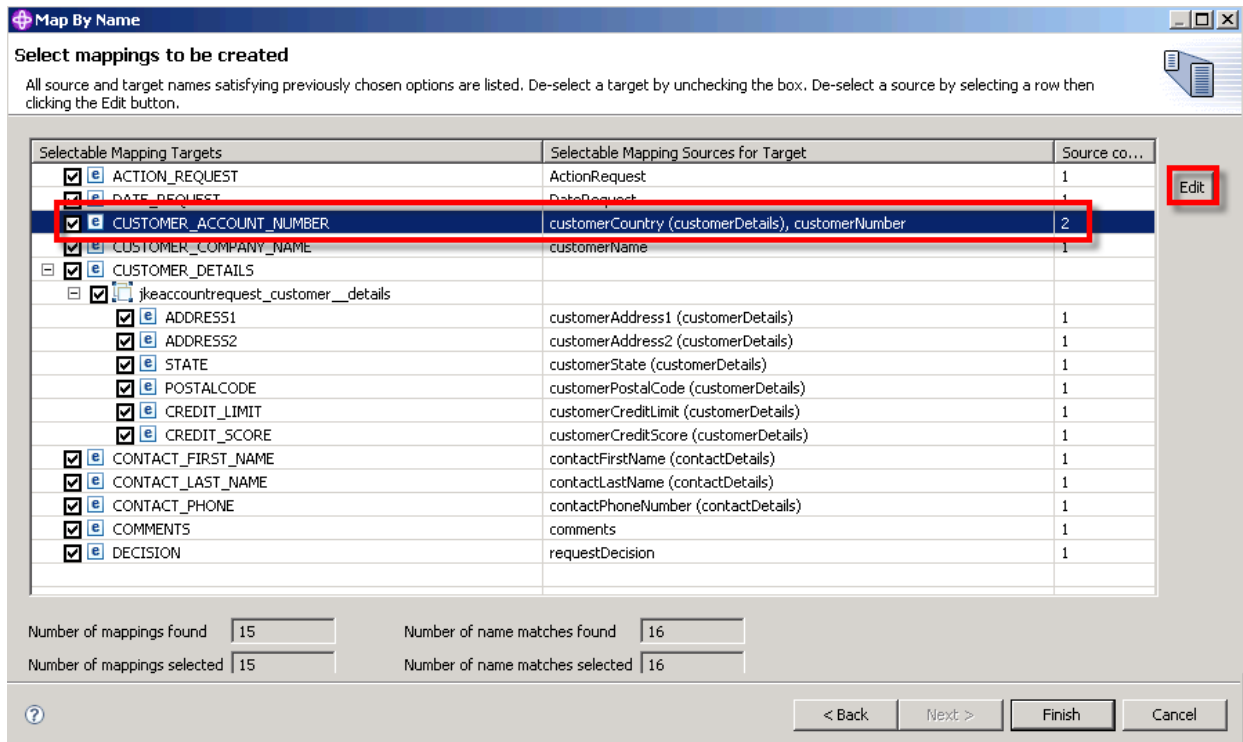
As an alternative you can also do a drag and drop of the source element to the target element in which case the same option panel will appear. In addition to the **Map by Name** option there are other options such as **Create a Submap**, **Map from Source** and **Call Existing Submap**. This capability is very useful when there is a structure or part of a structure that is used multiple times.



This mapping operation is between messages of different formats – XML for the source, COBOL for the target. The field names are different, the order of fields is different in some cases and in two cases data that is a string in the source becomes Packed Decimal (for one field) or Binary (for another field).

- __24. Select the **check** the box for **Create mappings when source and target names are more similar than**.
- __25. Set the **% match** to **50** using the slider bar or by overtyping the value.
- __26. **Uncheck** the box for Alphanumeric characters.
- __27. Click **Next**.

A comparison of the names will be performed and the results shown on the next screen.

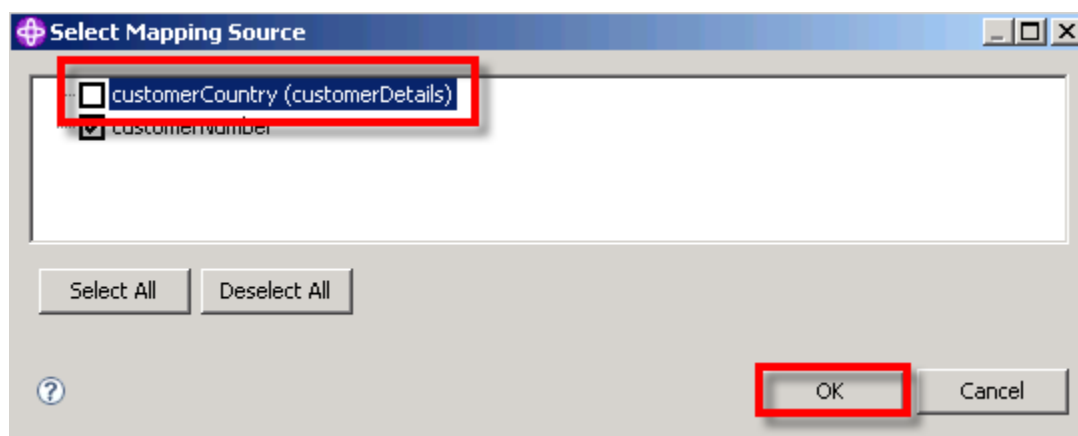


A preview of the results of the Map by Name operation using the parameters supplied on the previous panel is displayed. If the results are not satisfactory then the **Back** button can be used to return to the previous panel. This panel allows for incorrect or duplicate mappings to be removed.

In this case the **CUSTOMER_ACCOUNT_NUMBER** target has two sources. This is not correct so it will now be corrected.

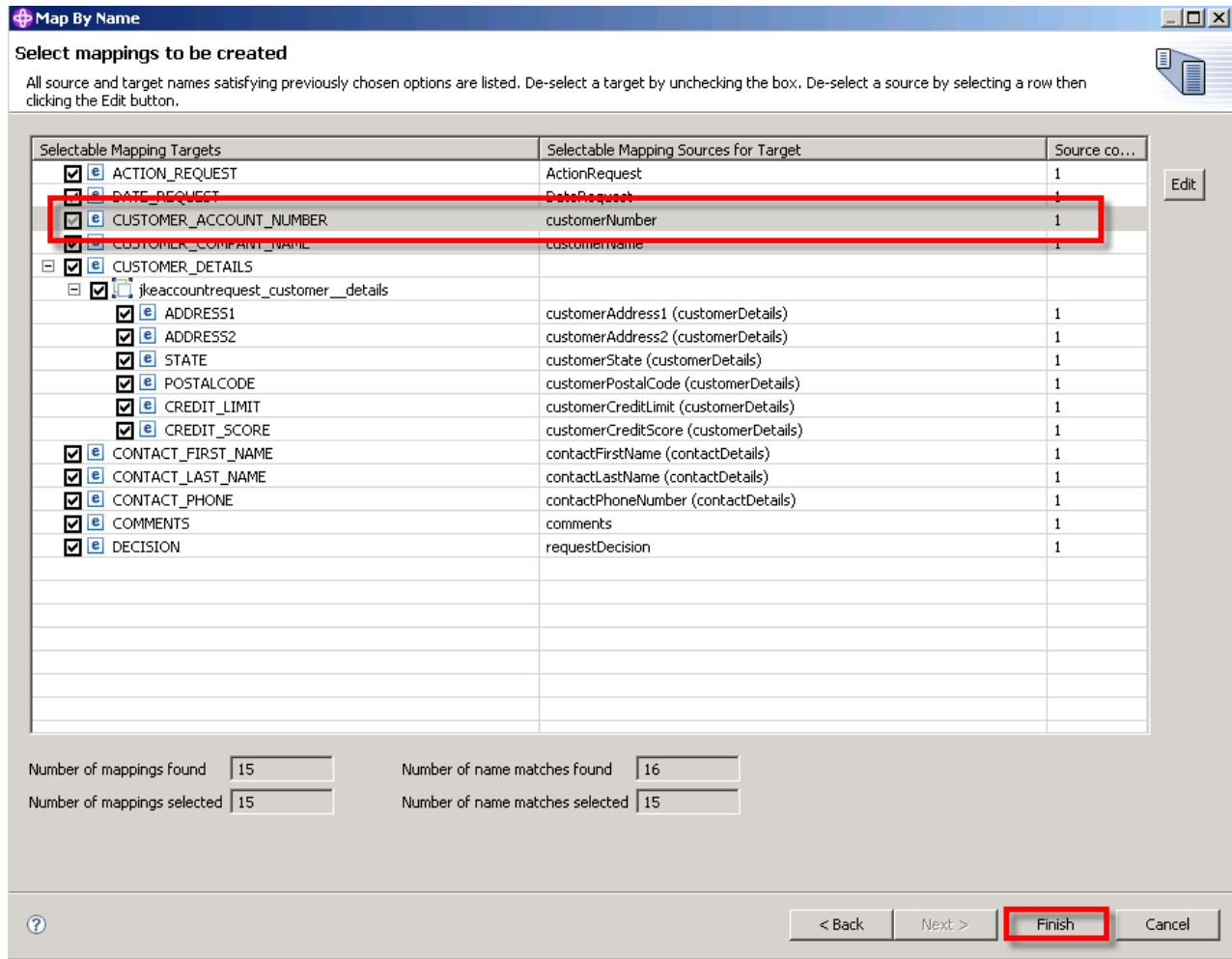
__28. **Select** the line with two sources.

__29. Click on the **Edit** button.



__30. **Uncheck** the box for **customerCountry** as this is not the proper source for this target.

__31. Click **OK** to remove the incorrect mapping.



CUSTOMER_ACCOUNT_NUMBER now has a single correct source.

__32. Click the **Finish** button to perform the mapping.

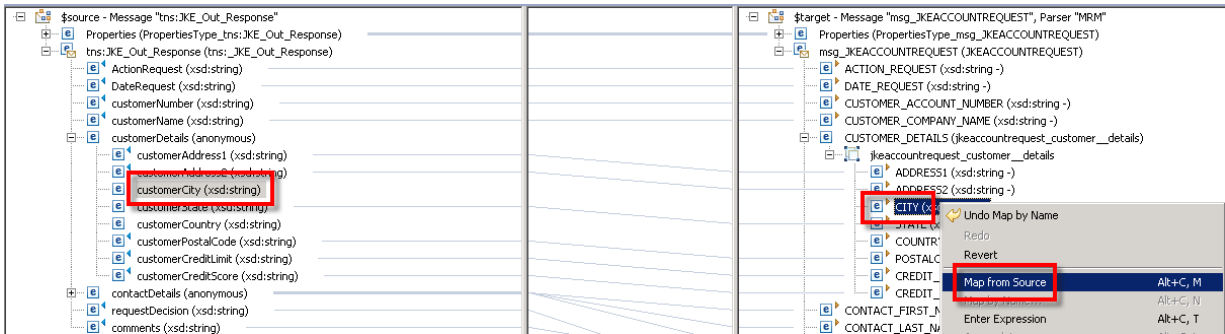


Take a few minutes and look at the results. In the **Source** and **Target** area blue lines show the connections between the items. If there is a mapping that is incorrect, you can click on the blue line in the middle section and remove the incorrect mapping.

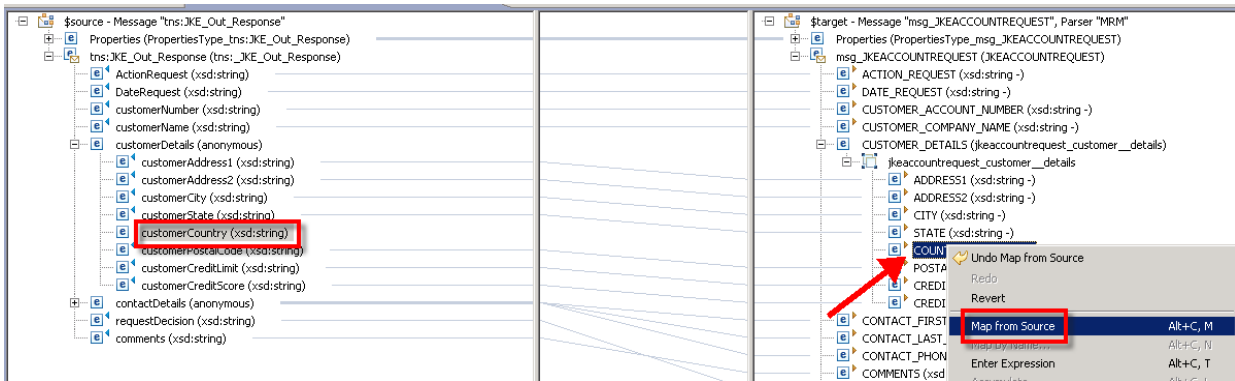
In this case we have the opposite condition. If you look closely you will see that there are two target fields, **CITY** and **COUNTRY**, that the Map by name algorithm was not able to establish a match for. At a 50% matching criteria the shorter names missed a match because of the “customer” prefix in front of those fields in the source names. These two elements will be mapped manually.

Map Script	Value
JKE_Server_Flow_JKE_In_to_JKE_COBOL	
Parameters	
\$target	
Properties	
msg_JKEACCOUNTREQUEST	
ACTION_REQUEST	\$source/tns:JKE_Out_Response/ActionRequest
DATE_REQUEST	\$source/tns:JKE_Out_Response/DateRequest
CUSTOMER_ACCOUNT_NUMBER	\$source/tns:JKE_Out_Response/customerNumber
CUSTOMER_COMPANY_NAME	\$source/tns:JKE_Out_Response/customerName
CUSTOMER_DETAILS	
ADDRESS1	\$source/tns:JKE_Out_Response/customerDetails/customerAddress1
ADDRESS2	\$source/tns:JKE_Out_Response/customerDetails/customerAddress2
CITY	" "
STATE	\$source/tns:JKE_Out_Response/customerDetails/customerState
COUNTRY	" "
POSTALCODE	\$source/tns:JKE_Out_Response/customerDetails/customerPostalCode
CREDIT_LIMIT	\$source/tns:JKE_Out_Response/customerDetails/customerCreditLimit
CREDIT_SCORE	\$source/tns:JKE_Out_Response/customerDetails/customerCreditScore
CONTACT_FIRST_NAME	\$source/tns:JKE_Out_Response/contactDetails/contactFirstName
CONTACT_LAST_NAME	\$source/tns:JKE_Out_Response/contactDetails/contactLastName
CONTACT_PHONE	\$source/tns:JKE_Out_Response/contactDetails/contactPhoneNumber

Here is a view from the Map Script area. Here we can spot the unmapped elements by looking for a lack of a **\$source** XPath statement in the Value cell for that particular target.




- __33. Click on **customerCity** in the Source pane.
- __34. Right click on **CITY** in the Target pane.
- __35. Select **Map from Source** (drag and drop also works).

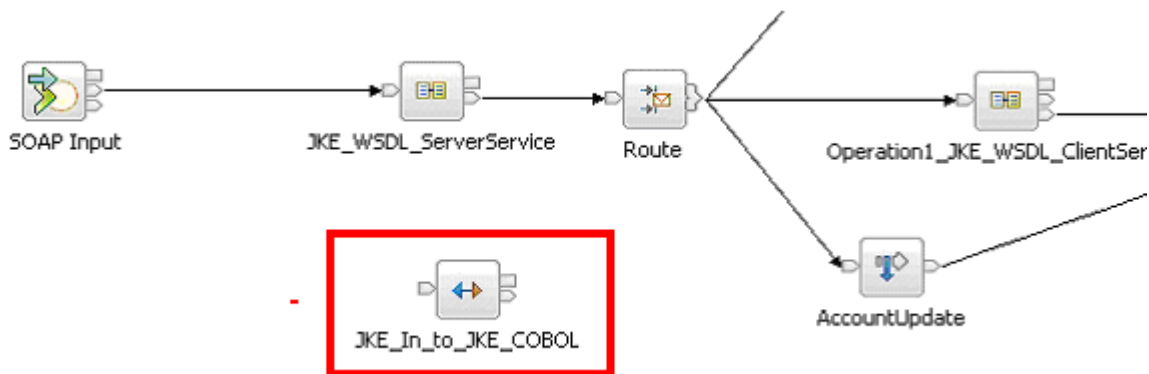


- __36. Click on **customerCountry** in the Source pane.
- __37. Right click on **COUNTRY** in the Target pane.
- __38. Select **Map from Source** (drag and drop also works).

Map Script	Value
msg_JKEACCOUNTREQUEST	
ACTION_REQUEST	<code>\$/source/tns:JKE_Out_Response/ActionRequest</code>
DATE_REQUEST	<code>\$/source/tns:JKE_Out_Response/DateRequest</code>
CUSTOMER_ACCOUNT_NUMBER	<code>\$/source/tns:JKE_Out_Response/customerNumber</code>
CUSTOMER_COMPANY_NAME	<code>\$/source/tns:JKE_Out_Response/customerName</code>
CUSTOMER_DETAILS	
ADDRESS1	<code>\$/source/tns:JKE_Out_Response/customerDetails/customerAddress1</code>
ADDRESS2	<code>\$/source/tns:JKE_Out_Response/customerDetails/customerAddress2</code>
CITY	<code>\$/source/tns:JKE_Out_Response/customerDetails/customerCity</code>
STATE	<code>\$/source/tns:JKE_Out_Response/customerDetails/customerState</code>
COUNTRY	
POSTAL_CODE	<code>\$/source/tns:JKE_Out_Response/customerDetails/customerPostalCode</code>
CREDIT_LIMIT	<code>\$/source/tns:JKE_Out_Response/customerDetails/customerCreditLimit</code>
CREDIT_SCORE	<code>\$/source/tns:JKE_Out_Response/customerDetails/customerCreditScore</code>
CONTACT_FIRST_NAME	<code>\$/source/tns:JKE_Out_Response/contactDetails/contactFirstName</code>
CONTACT_LAST_NAME	<code>\$/source/tns:JKE_Out_Response/contactDetails/contactLastName</code>
CONTACT_PHONE	<code>\$/source/tns:JKE_Out_Response/contactDetails/contactPhoneNumber</code>
COMMENTS	<code>\$/source/tns:JKE_Out_Response/comments</code>
DECISION	<code>\$/source/tns:JKE_Out_Response/requestDecision</code>

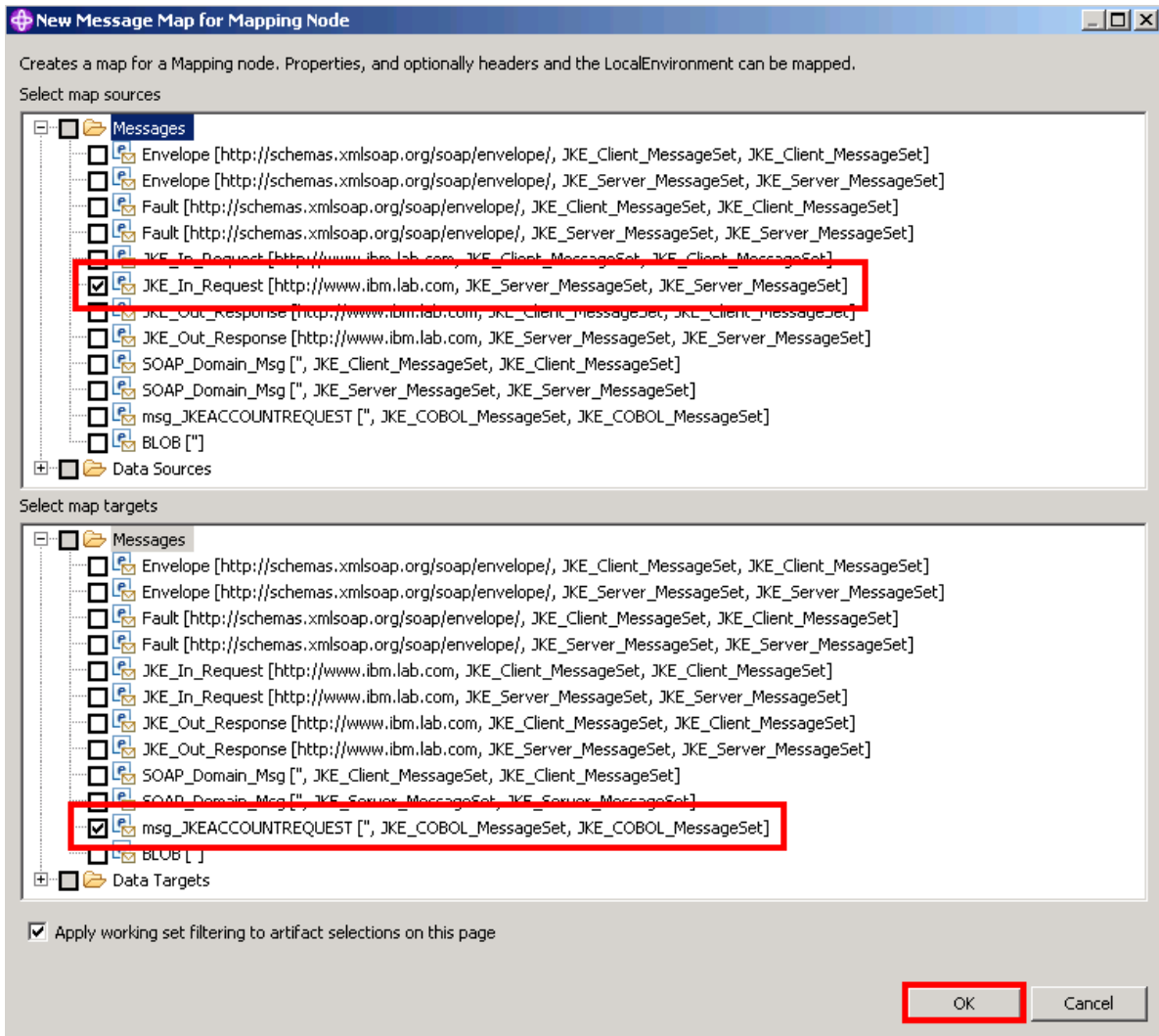
Here is the end result with all fields properly mapped.

- ___39. When you are finished examining the results,  save the map.
- ___40. Close the mapping editor.



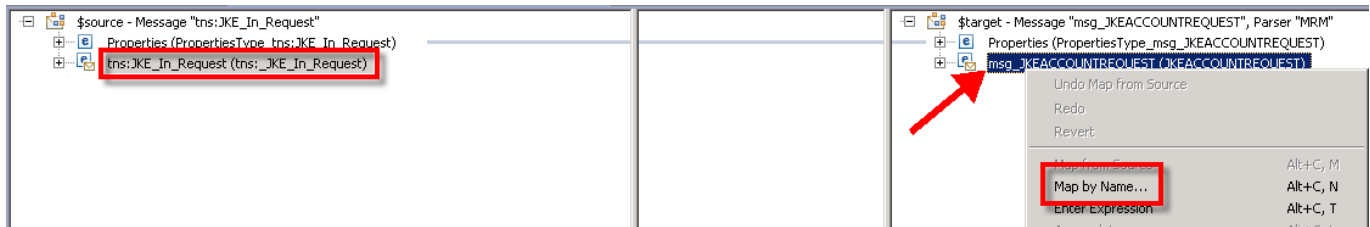
The next map to be built is on the left side of the message flow.

- ___41. Double-click on the **JKE_In_to_JKE_Cobol** node.

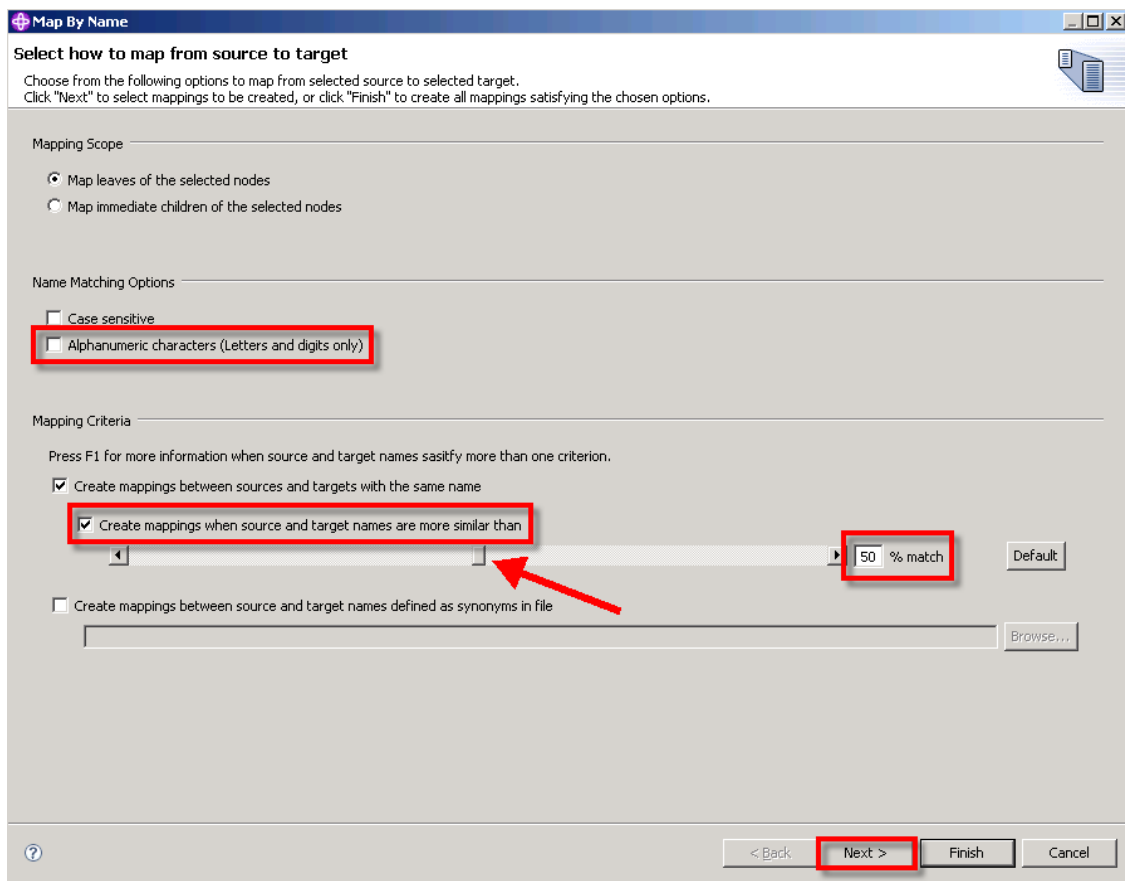


The mapping wizard presents the source and target options. You may be wondering about where all of this information comes from. Message definitions from all message sets in the Workspace are presented. (Note that BLOB is supported.)

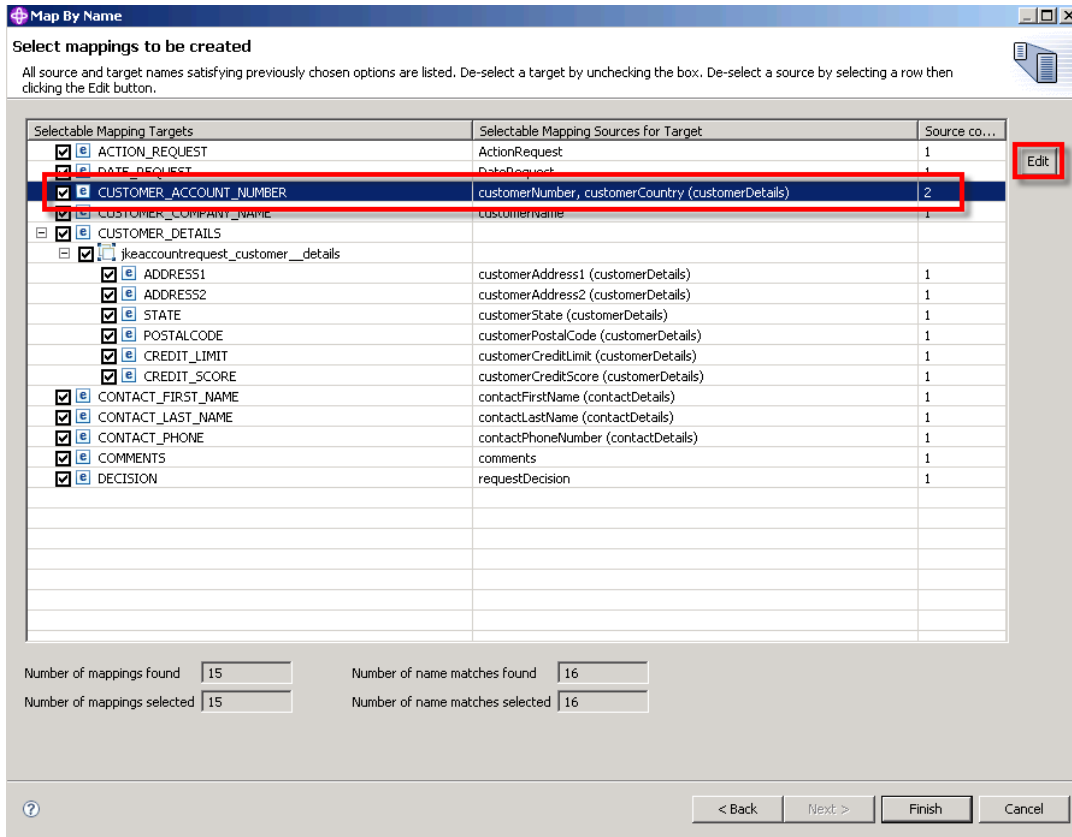
- __42. Click the check box for **JKE_In_Request[JKE_Server_MessageSet]** for the map **source**.
- __43. Click the check box for **msg_JKEACCOUNTREQUEST[JKE_COBOL_MessageSet]** for the map **target**.
- __44. **Make sure you have selected the correct items.** There are **JKE_In_Request** items for both the Client and Server message sets (the **Server** one is correct).
- __45. Click **OK**.



- __46. Click on **tns:JKE_In_Request** in the source pane.
- __47. Right-click on **msg_JKEACCOUNTREQUEST** in the target pane.
- __48. Select **Map by Name** from the menu.



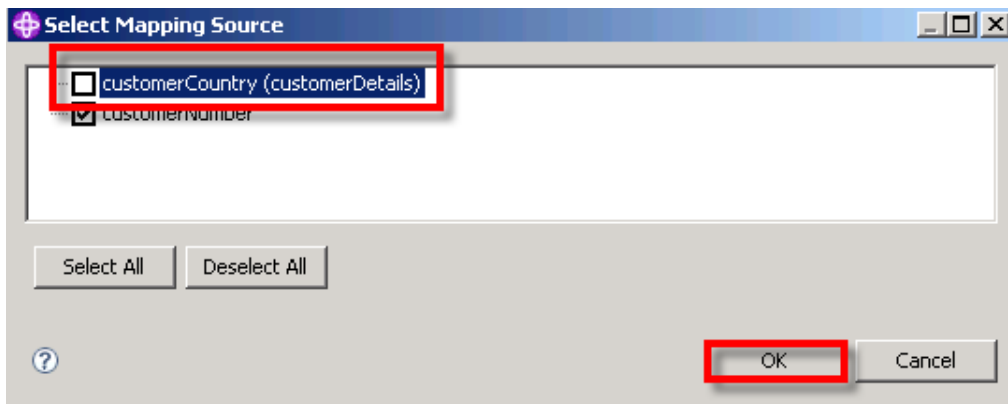
- __49. **Check** the box for **Create mappings when source and target names are more similar than**.
- __50. Set the % match to **50** using the slider bar or by overtyping.
- __51. **Uncheck** the box for Alphanumeric characters.
- __52. Click **Next**.



Note that the **CUSTOMER_ACCOUNT_NUMBER** target has two sources.

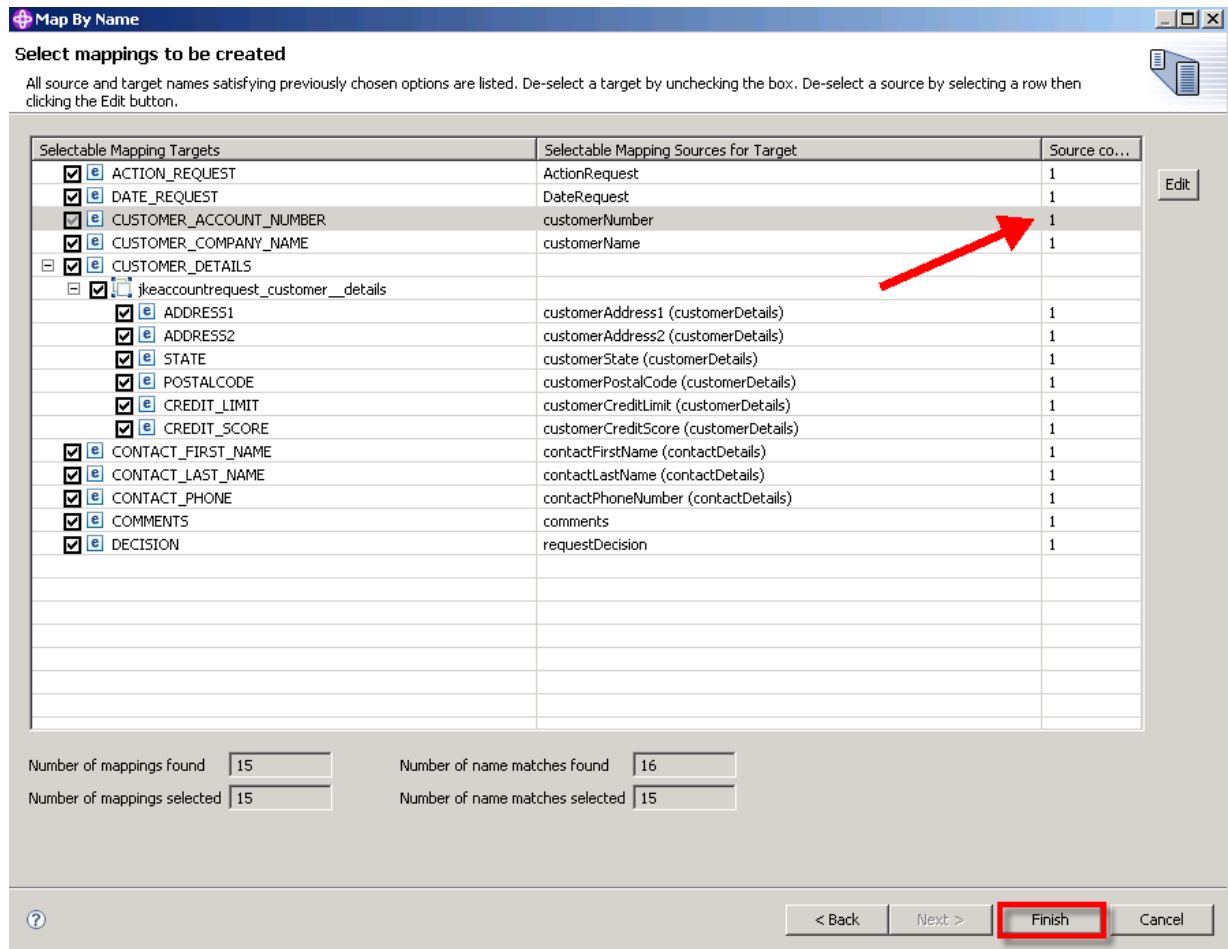
__53. **Select** the line with two sources.

__54. Click on the **Edit** button.



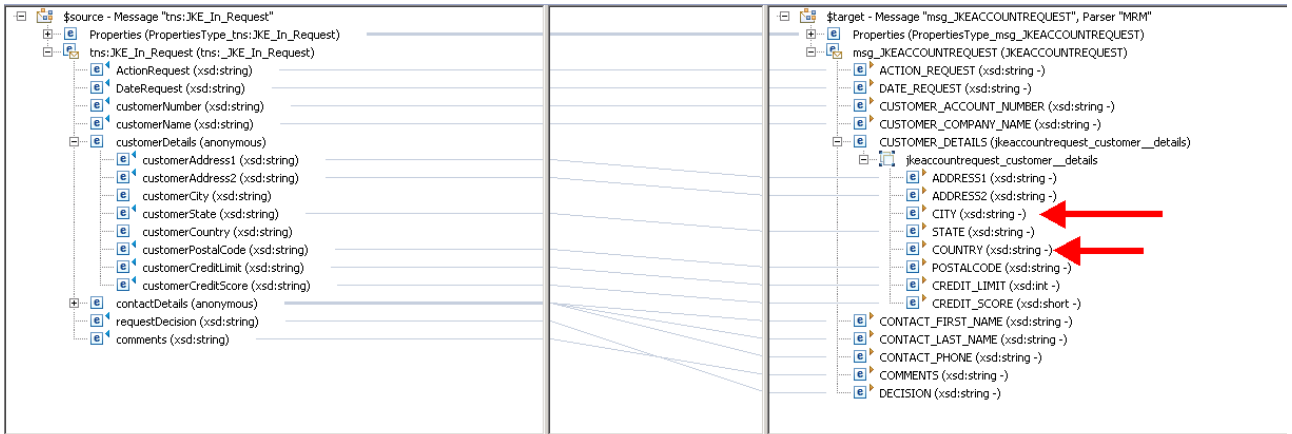
__55. Uncheck the box for **customerCountry** as this is not the proper source for this target.

__56. Click **OK**.



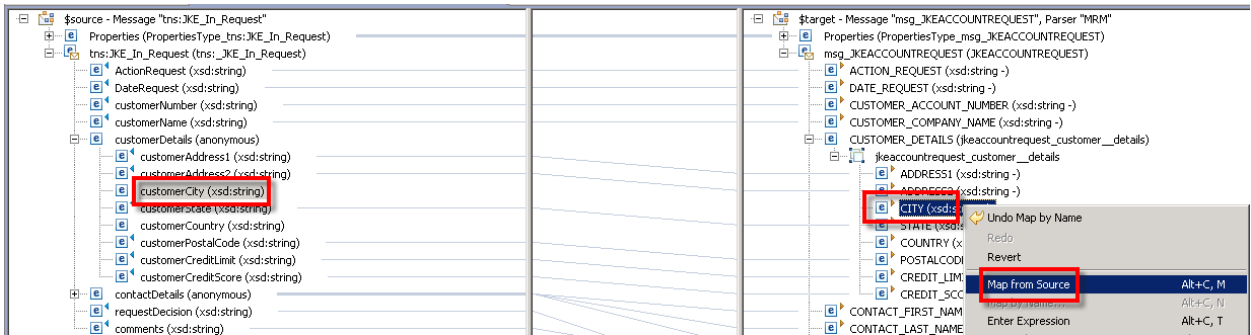
CUSTOMER_ACCOUNT_NUMBER now has a single correct source.

__57. Click on the **Finish** button to perform the map by name operation.

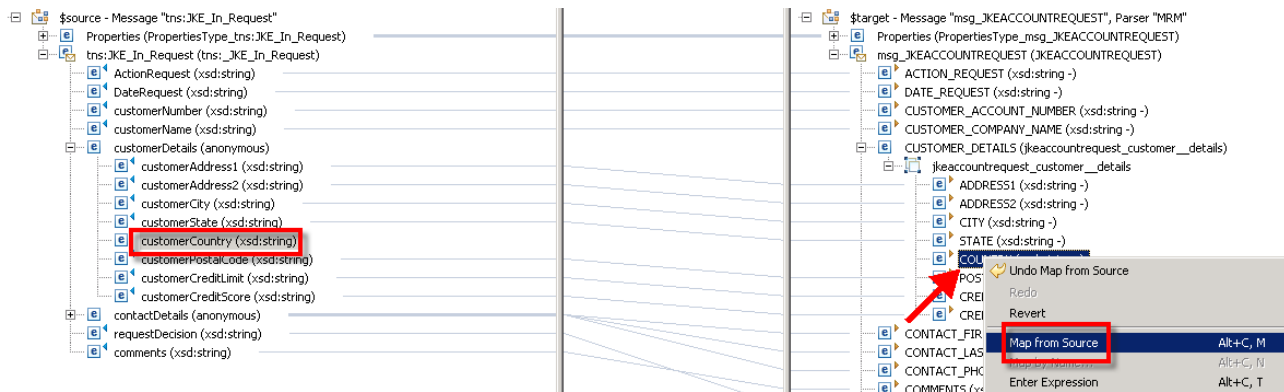


Map Script	Value
JKE_Server_Flow_JKE_In_to_JKE_COBOL	
Parameters	
\$target	
Properties	
msg:JKEACCOUNTREQUEST	
ACTION_REQUEST	\$source/tns:JKE_In_Request/ActionRequest
DATE_REQUEST	\$source/tns:JKE_In_Request/DateRequest
CUSTOMER_ACCOUNT_NUMBER	\$source/tns:JKE_In_Request/customerNumber
CUSTOMER_COMPANY_NAME	\$source/tns:JKE_In_Request/customerName
CUSTOMER_DETAILS	
ADDRESS1	\$source/tns:JKE_In_Request/customerDetails/customerAddress1
ADDRESS2	\$source/tns:JKE_In_Request/customerDetails/customerAddress2
CITY	""
STATE	\$source/tns:JKE_In_Request/customerDetails/customerState
COUNTRY	""
POSTALCODE	\$source/tns:JKE_In_Request/customerDetails/customerPostalCode
CREDIT_LIMIT	\$source/tns:JKE_In_Request/customerDetails/customerCreditLimit

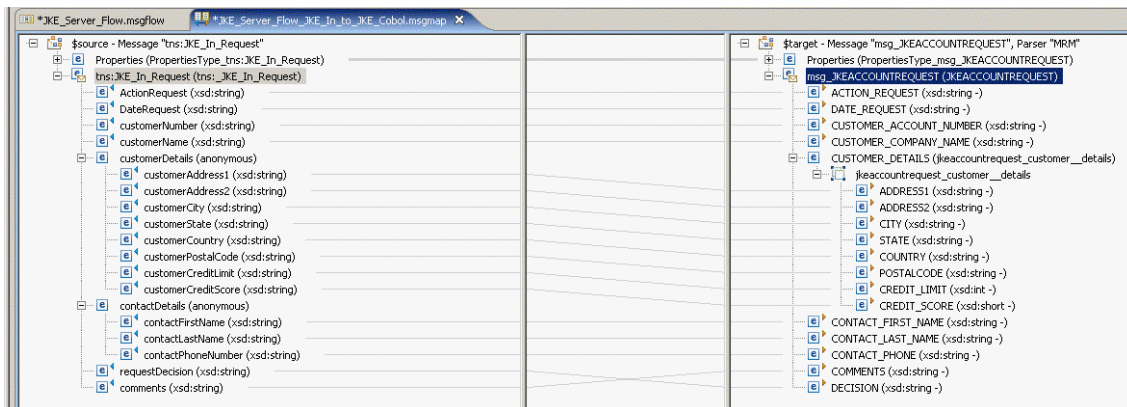
As in the last mapping, two elements did not get mapped, namely **CITY** and **COUNTRY**.



- __58. Click on **customerCity** in the Source pane.
- __59. Right click on **CITY** in the Target pane.
- __60. Select **Map from Source** from the menu.




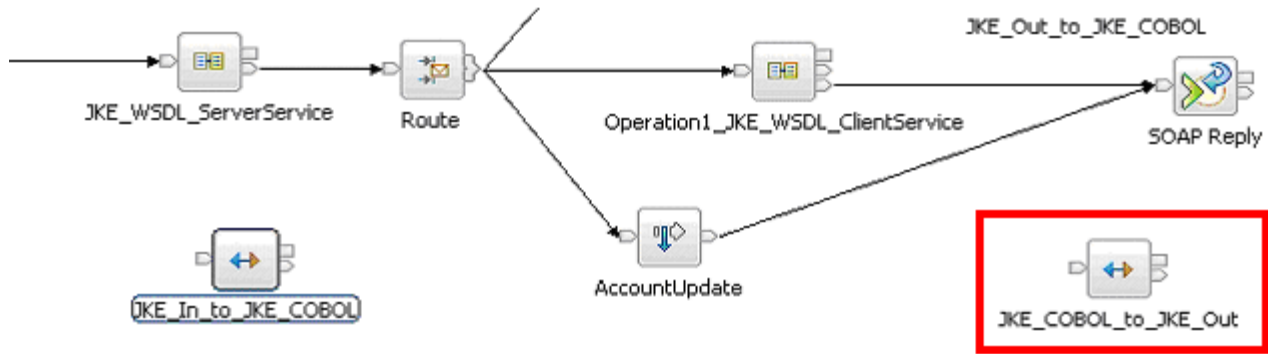
- __61. Click on **customerCountry** in the Source pane.
- __62. Right click on **COUNTRY** in the Target pane.
- __63. Select **Map from Source** from the menu.



- __64. Verify that all items have been mapped. The lower section is not shown.

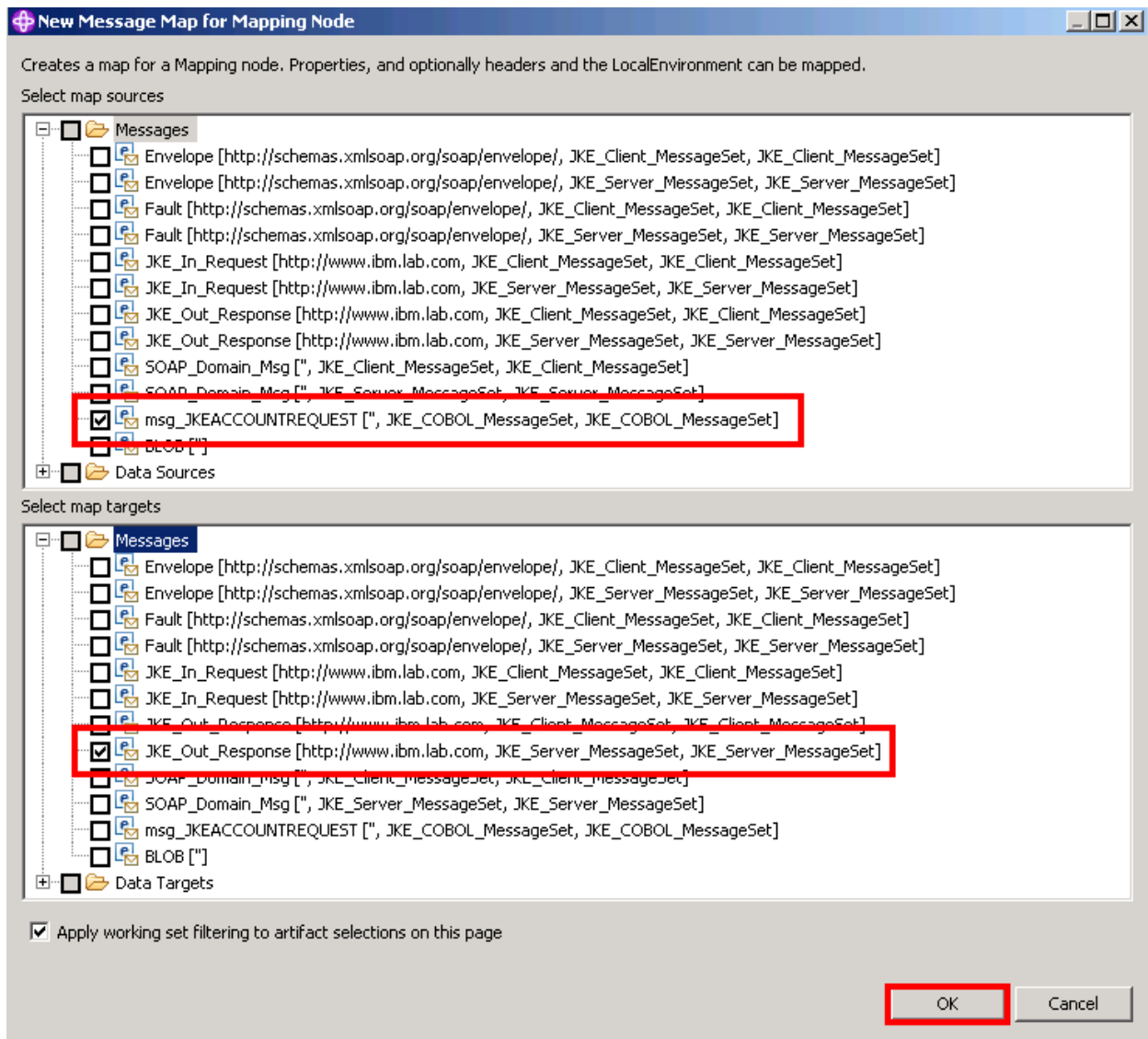
When dealing with fixed format messages, such as COBOL, every field must be mapped unless that field has been defined with a default value. It is also worth noting that the COBOL items do have a max length associated with each field. When mapping from variable length fields, such as XML, if a source field is too long it will be truncated in the target. With the sample messages provided with the lab, this is not the case.

- __65. Save  the map.
- __66. Close the mapping editor.

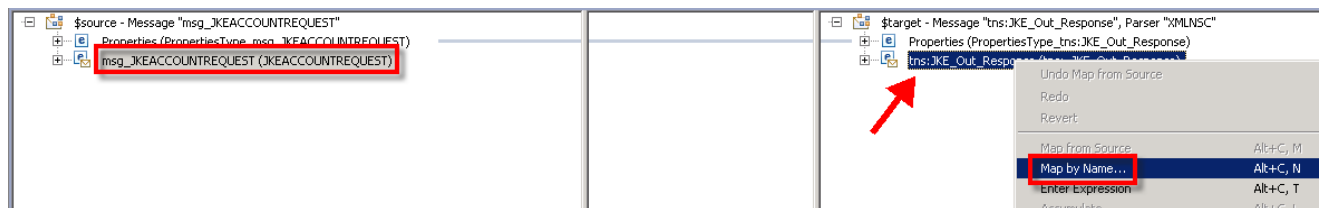


__67. **Double click** on the third Mapping node, **JKE_COBOL_to_JKE_Out** (lower right).

This time the source will be COBOL.



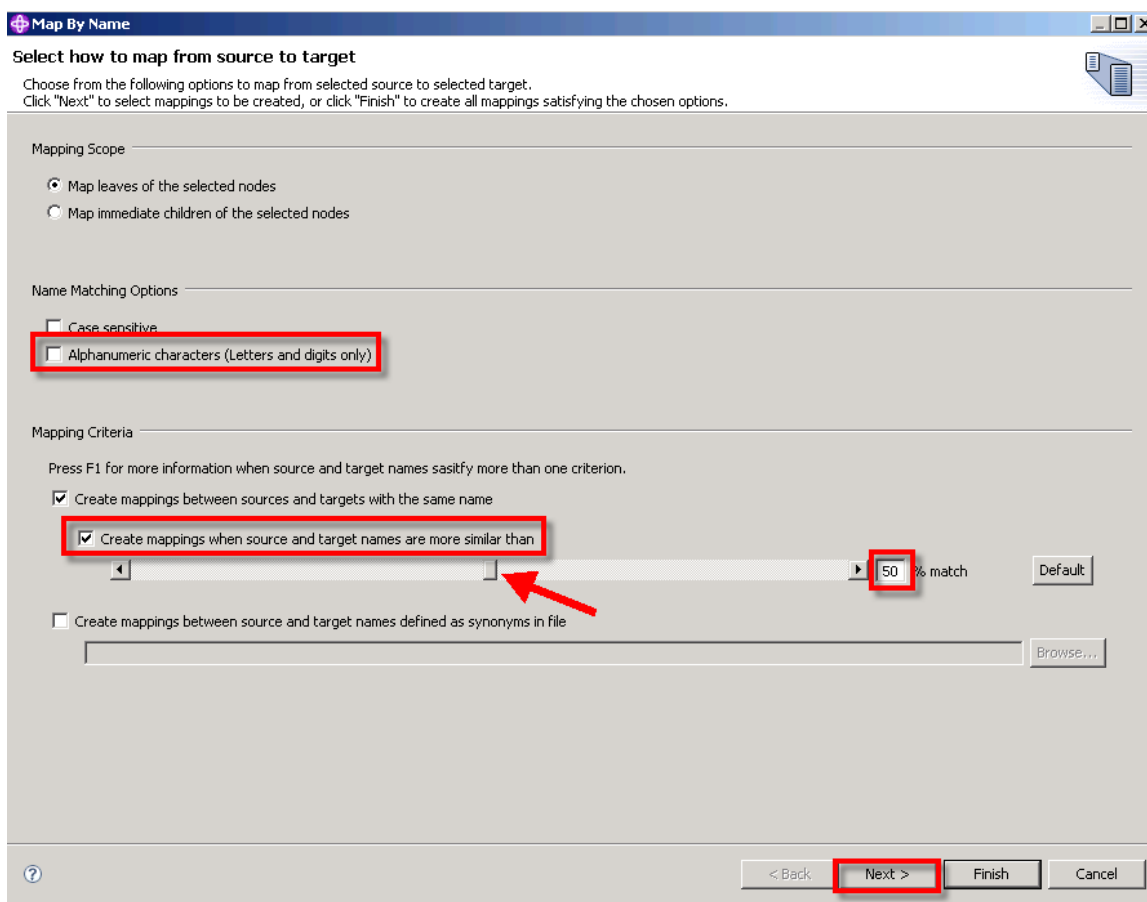
- __68. Click the check box for **msg_JKEACCOUNTREQUEST[JKE_COBOL_MessageSet]** for the map **source**.
- __69. Click the check box for **JKE_Out_Response[JKE_Server_MessageSet]** for the map **target**.
- __70. **Make sure you have selected the correct items.** There are JKE_Out_Response items for both the Client and Server message sets (the server one is correct.)
- __71. Double check you selections to make sure they are accurate.
- __72. Click **OK**.



__73. Highlight **msg_JKEACCOUNTREQUEST** in the **source** pane.

__74. Right-click on **tns:JKE_Out_Response** in the **target** pane.

__75. Select **Map by Name** from the menu.

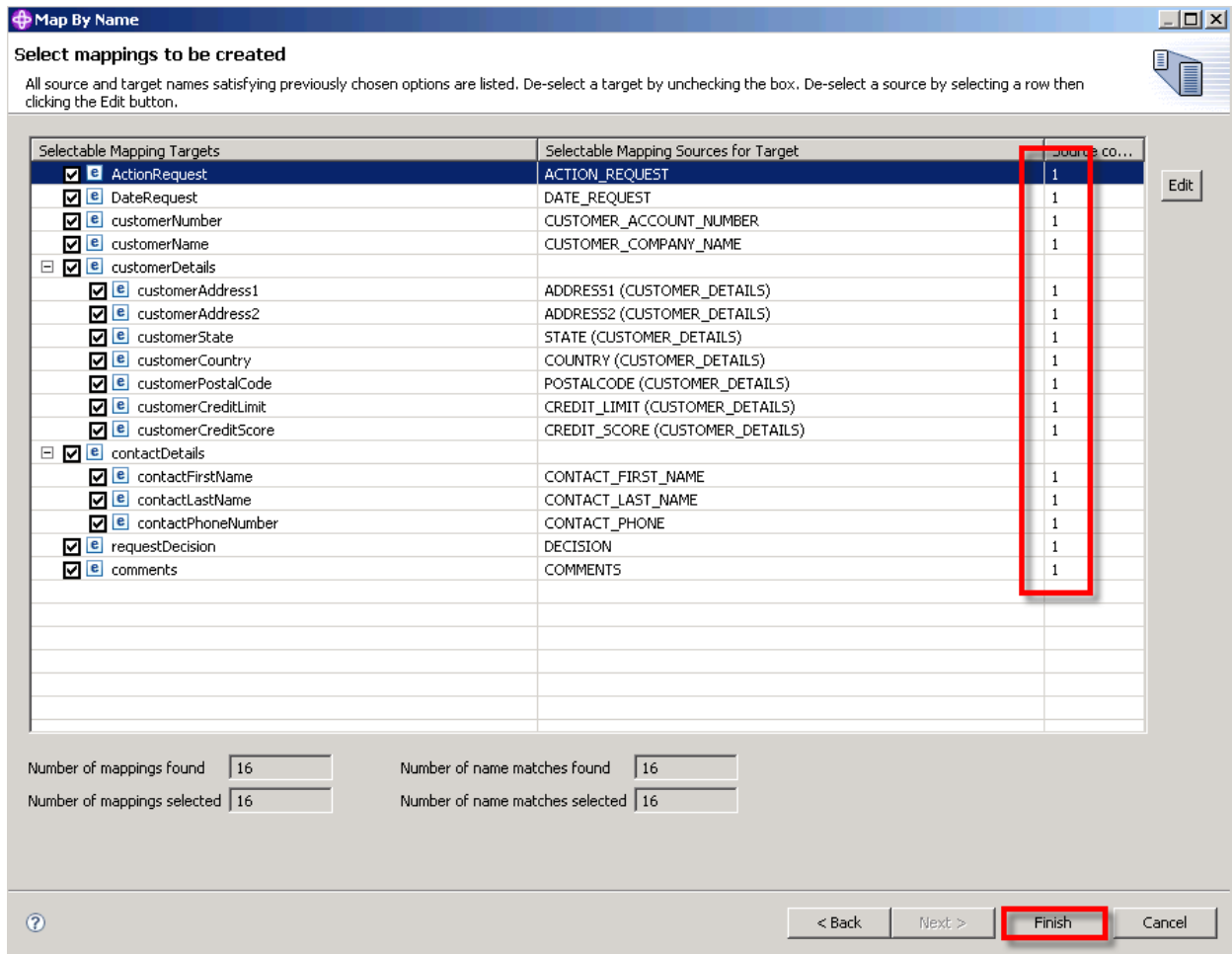


__76. Check the box for **Create mappings when source and target names are more similar than**.

__77. Set the **% match** to **50** using the slider bar or by overtyping.

__78. Uncheck the box for **Alphanumeric** characters.

__79. Click **Next**. Remember that this time you are mapping from COBOL to XML.



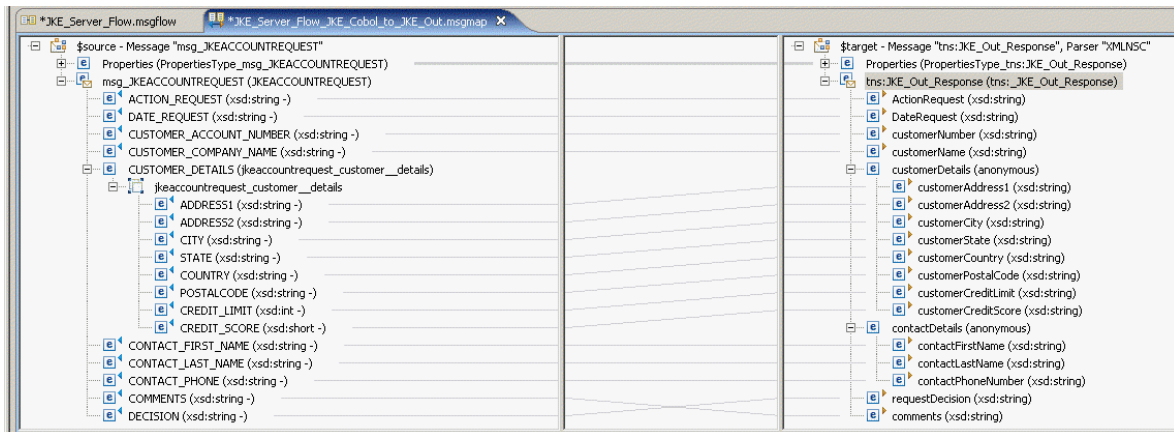
Mapping in this direction does not create any targets that have more than one source.

__80. Click the **Finish** button

Map Script	Value
\$JKE_Server_Flow_JKE_COBOL_To_JKE_Out	
Parameters	
\$target	
Properties	
tns:JKE_Out_Response	
ActionRequest	\$source/msg_JKEACCOUNTREQUEST/ACTION_REQUEST
DateRequest	\$source/msg_JKEACCOUNTREQUEST/DATE_REQUEST
customerNumber	\$source/msg_JKEACCOUNTREQUEST/CUSTOMER_ACCOUNT_NUMBER
customerName	\$source/msg_JKEACCOUNTREQUEST/CUSTOMER_COMPANY_NAME
customerDetails	
contactDetails	
requestDecision	\$source/msg_JKEACCOUNTREQUEST/DECISION
comments	\$source/msa_JKEACCOUNTREQUEST/COMMENTS

A single Target did not get mapped, namely **customerCity**.

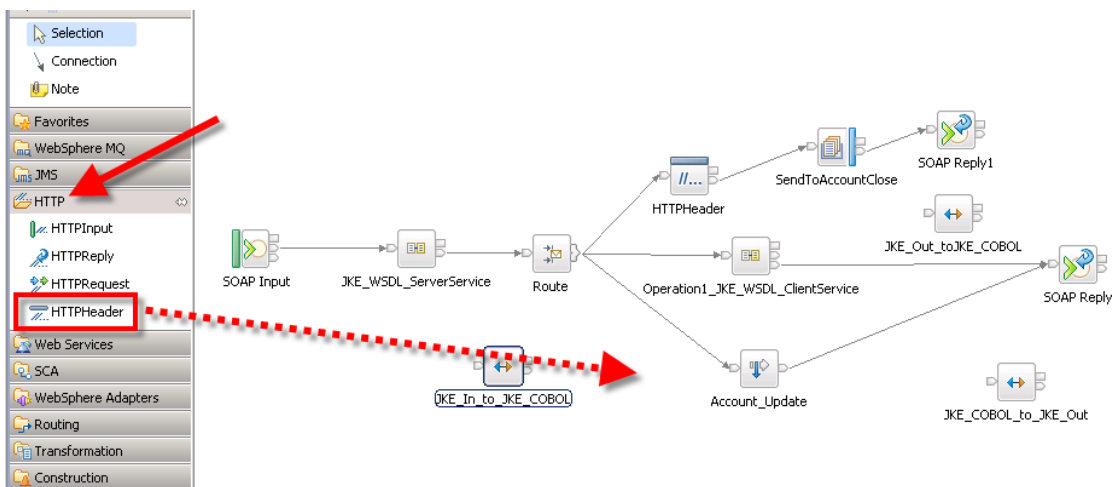
- __81. Click on **CITY** in the **Source** pane.
- __82. Right click on **customerCity** in the **Target** pane.
- __83. Select **Map from Source** from the menu.



All the fields are now mapped. This completes the mapping portion of this lab.

__84. Save  the map.

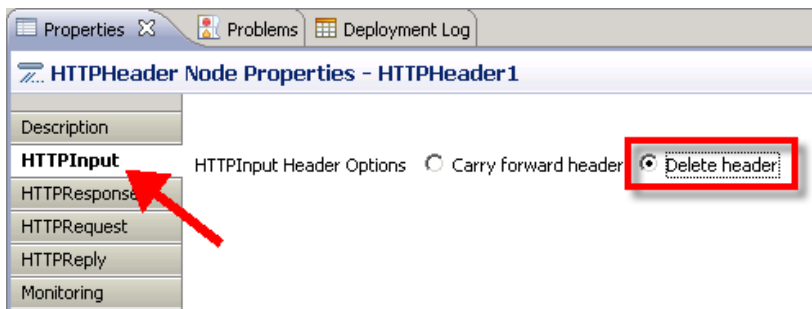
__85. Close the mapping editor.



There is one final task before testing the message flow. The COBOL message will be sent to a back-end (simulated) CICS system in the next lab. Since the message originated as a SOAP Web service, it does not have a WebSphere MQ Message Descriptor (MQMD) header but it does have the HTTPInput header that was shown in the AccountUpdate.txt trace file. The HTTPInput header must be removed before sending the message to CICS via an MQOutput node. An MQMD will be added automatically.

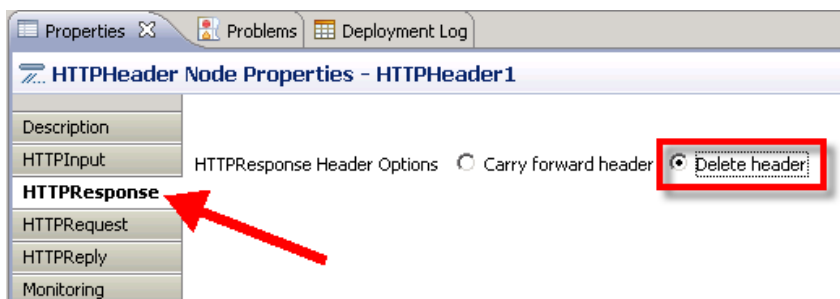
__86. Open the **HTTP** drawer.

__87. Drag an **HTTPHeader** node to the message flow placing it after the **JKE_In_to_JKE_COBOL** mapping node.



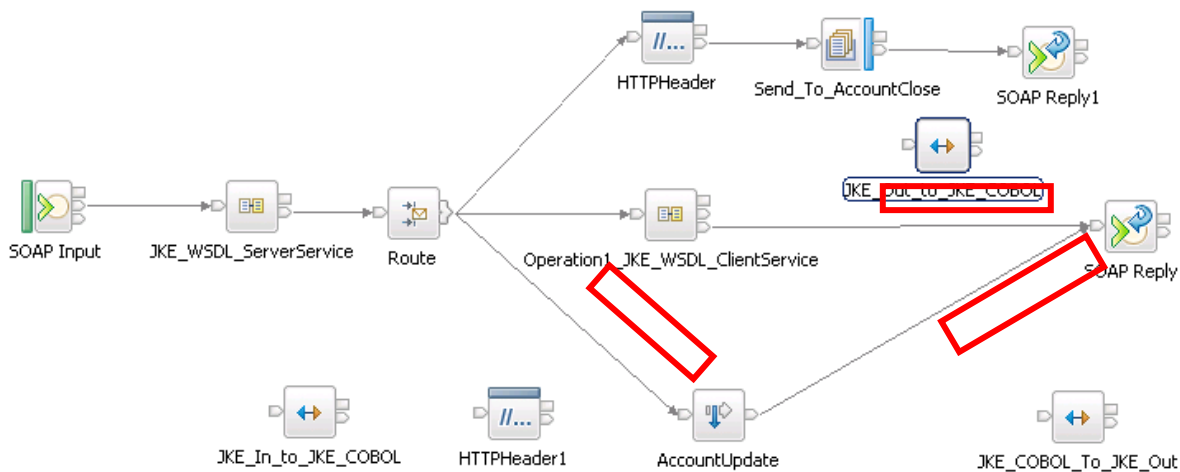
__88. Select the **HTTPInput** tab.

__89. Click the radio button for **Delete header**.



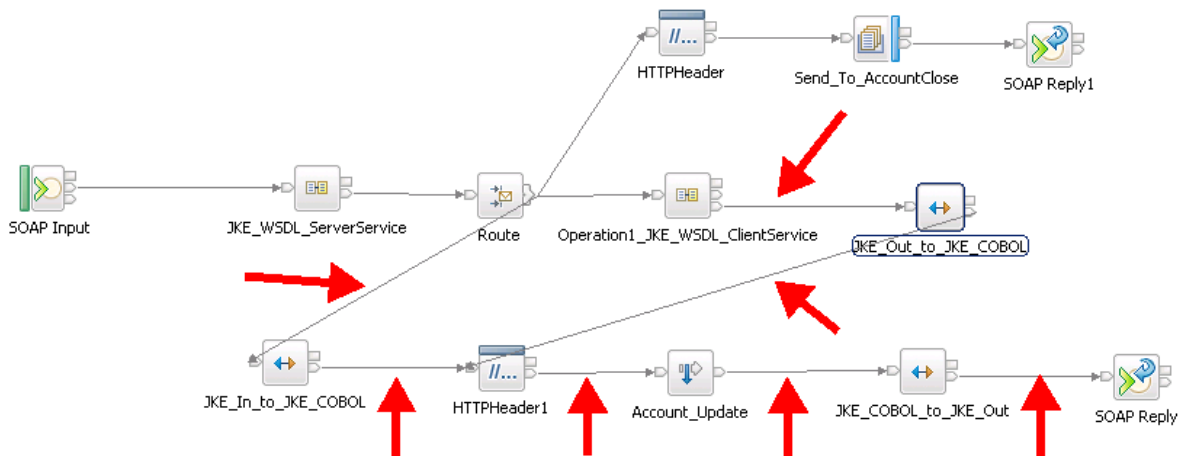
__90. Select the **HTTPResponse** tab.

__91. Click the radio button for **Delete header**.




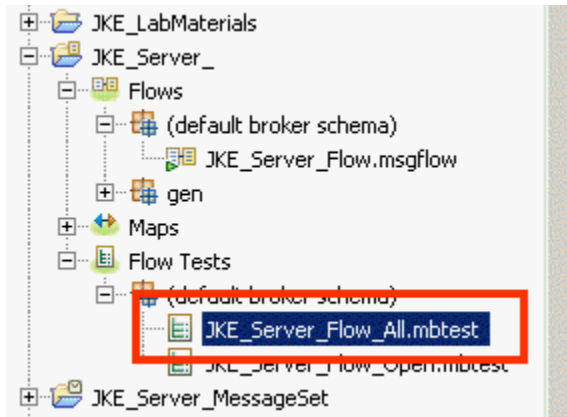
Remove the existing connection between the **Operation1_JKE_WSDL_ClientService** node and the **SOAP Reply** node.

- __92. Right click on the existing connection between the **Route** and **AccountUpdate** nodes.
- __93. Select **Delete** from the menu.
- __94. Right click on the existing connection between the **AccountUpdate** and **SOAP Reply** nodes.
- __95. Select **Delete** from the menu.



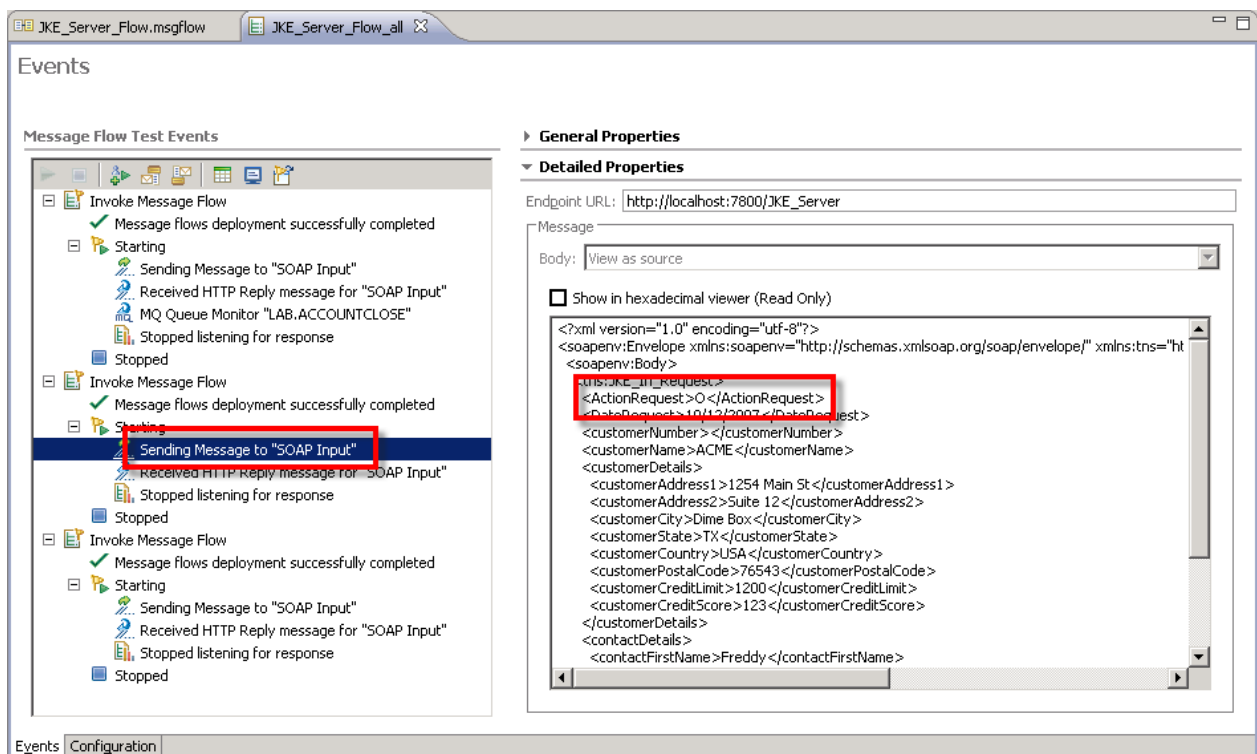
Wire the new nodes as shown.

- __96. The **JKE_Out_Reponse** terminal of the **Operation1_JKE_WSDL_ClientService** node is connected to the **JKE_Out_to_JKE_COBOL** node.
- __97. Verify that the **Account_Update** path from the Route node is connected to the **JKE_In_to_JKE_COBOL** node, which is then connected to the **HTTPHeader1** node.
- __98. The **JKE_Out_to_JKE_COBOL** node is also connected to the **HTTPHeader1** node.
- __99. The **HTTPHeader1** is connected to the **AccountUpdate** Trace node. This will allow us to see the message tree after it has been transformed by the Mapping nodes from XML to COBOL.
- __100. The **AccountUpdate** Trace node is connected to the **JKE_COBOL_to_JKE_Out** node which is then connected to the SOAP Reply node.
- __101. Verify that the connections are correct.
- __102.  **Save** the message flow.

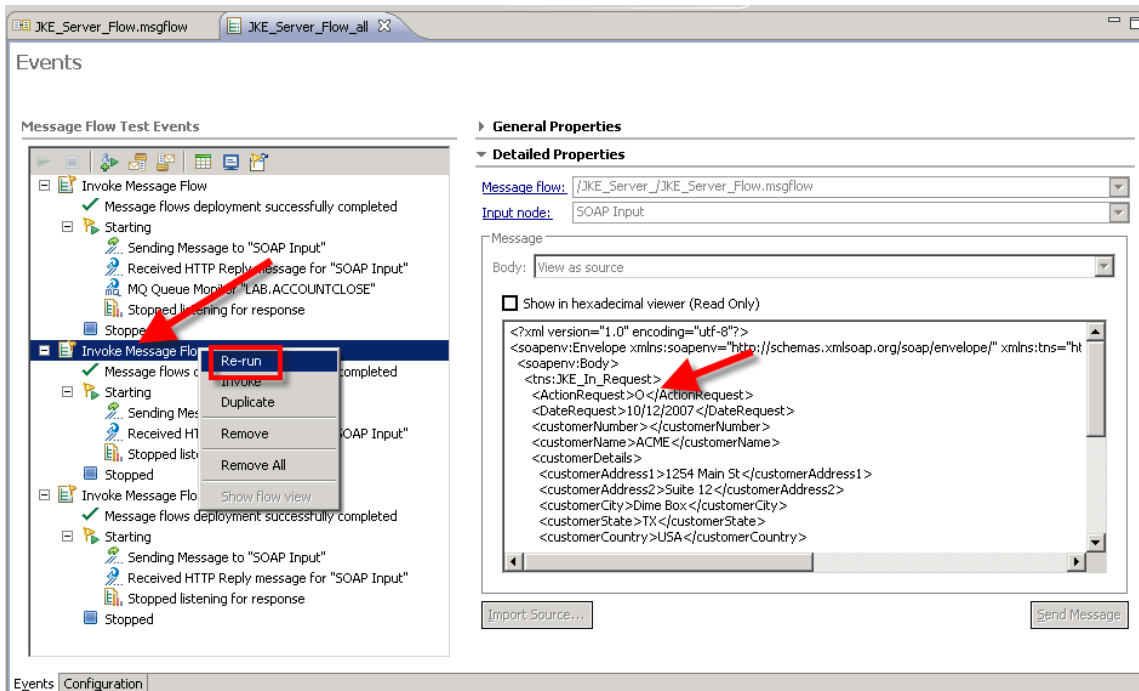


It is now time to test the Account_Open and Account_Update paths using the saved test configuration from the previous lab.

- ___103. Select the **JKE_Server_Flow_All.mbstest** test configuration in the **Flow Tests** folder under the **JKE_Server_** project.
- ___104. Double-click on the file to launch the Test Client.

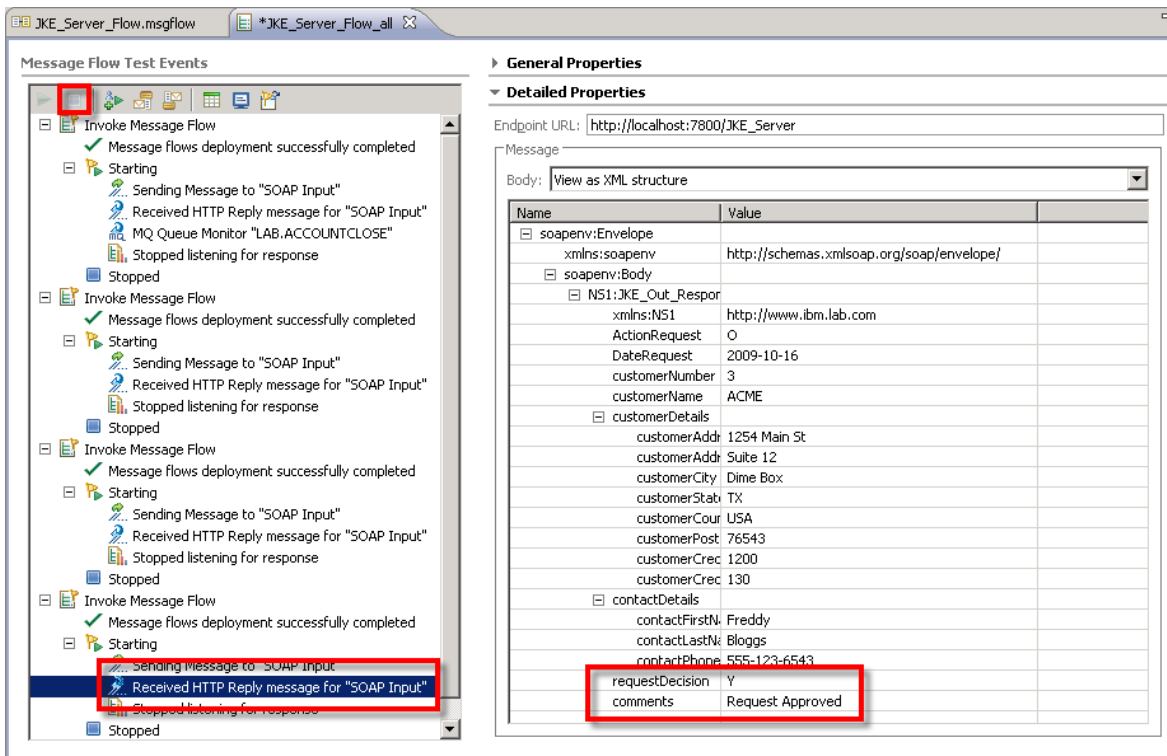


You want to rerun the second test, which was Account Open. You can validate that the test is for Account Open by highlighting the second **Sending Message to "Input"** line and looking at the **Action Request** field in the message to verify that it contains an 'O'.



__105. Right-click on the *second* Invoke Message Flow entry.

__106. Select **Re-run**.



A response is returned to the Web Services client and the Test Client continues to wait for a response on the LAB.ACCOUNTCLOSE queue.

Message Flow Test Events

- Invoke Message Flow
 - Message flows deployment successfully completed
 - Starting
 - Sending Message to "SOAP Input"
 - Received HTTP Reply message for "SOAP Input"
 - MQ Queue Monitor "LAB.ACCOUNTCLOSE"
 - Stopped listening for response
 - Stopped
- Invoke Message Flow
 - Message flows deployment successfully completed
 - Starting
 - Sending Message to "SOAP Input"
 - Received HTTP Reply message for "SOAP Input"
 - Stopped listening for response
 - Stopped
- Invoke Message Flow
 - Message flows deployment successfully completed
 - Starting
 - Sending Message to "SOAP Input"
 - Received HTTP Reply message for "SOAP Input"
 - Stopped listening for response
 - Stopped
- Invoke Message Flow
 - Message flows deployment successfully completed
 - Starting
 - Sending Message to "SOAP Input"
 - Received HTTP Reply message for "SOAP Input"
 - Stopped listening for response
 - Stopped

General Properties

Detailed Properties

Endpoint URL: `http://localhost:7800/JKE_Server`

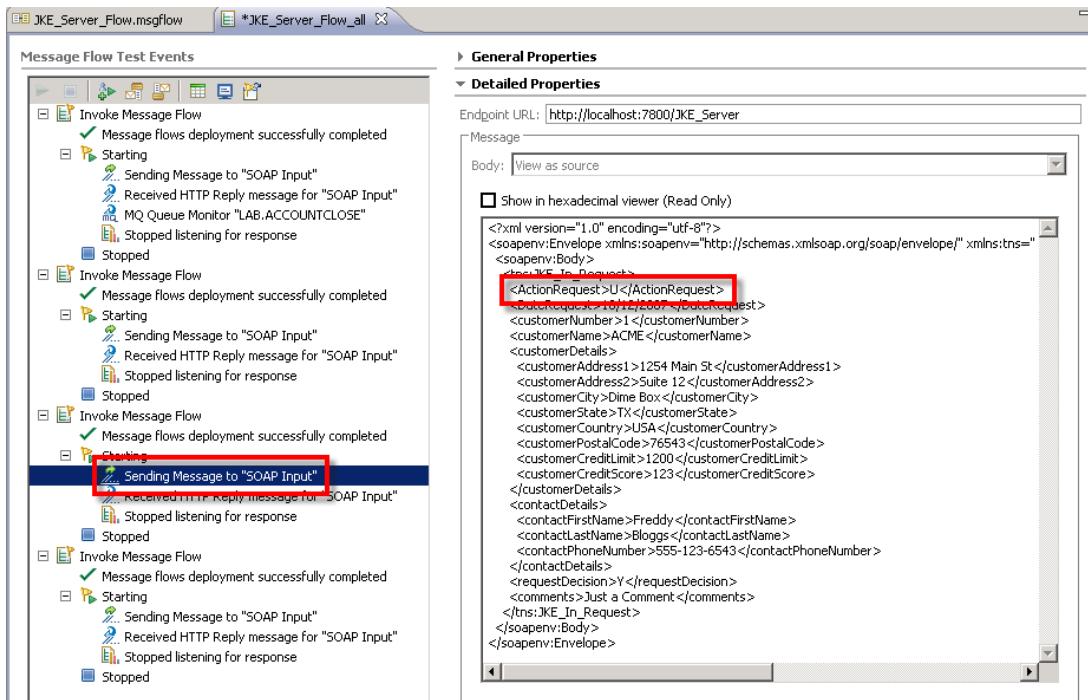
Message

Body: View as XML structure

Name	Value
soapenv:Envelope	
xmlns:soapenv	http://schemas.xmlsoap.org/soap/envelope/
soapenv:Body	
NS1:JKE_Out_Respons	
xmlns:NS1	http://www.ibm.lab.com
ActionRequest	0
DateRequest	2009-10-16
customerNumber	3
customerName	ACME
customerDetails	
customerAddr	1254 Main St
customerAddr	Suite 12
customerCity	Dime Box
customerState	TX
customerCountry	USA
customerPost	76543
customerCrec	1200
customerCrec	130
contactDetails	
contactFirstN	Freddy
contactLastN	Bloggs
contactPhone	555-123-6543
requestDecision	Y
comments	Request Approved

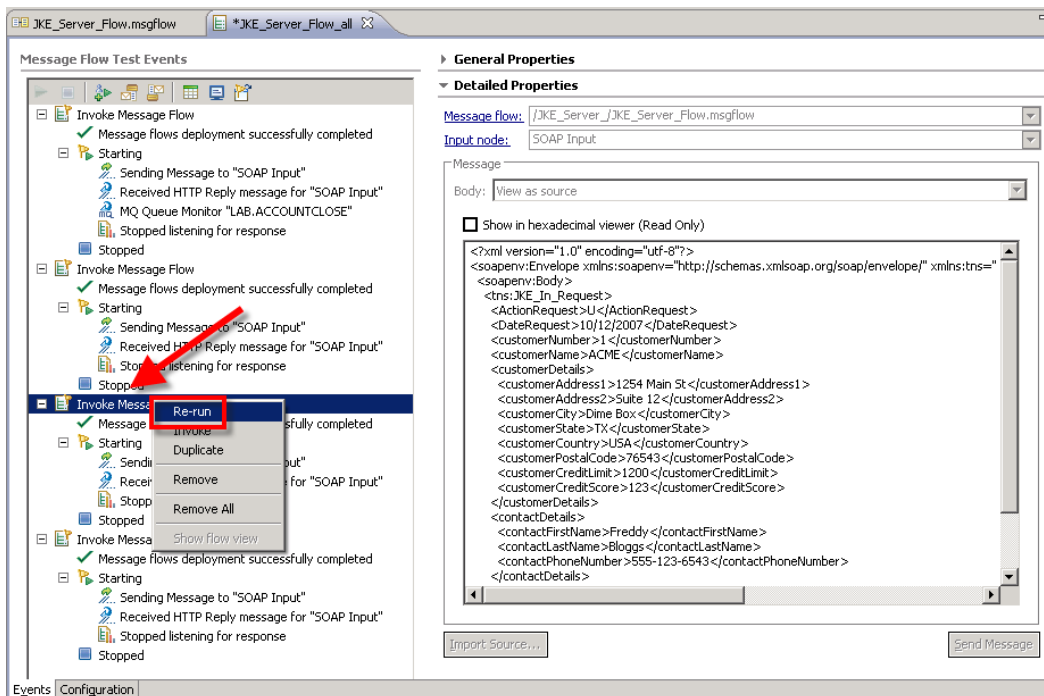
__107. Press the small square icon to stop the test. The flow test is waiting for a message to arrive on the account close queue, which will not happen for an open request.

__108. Ensure the test has stopped.....



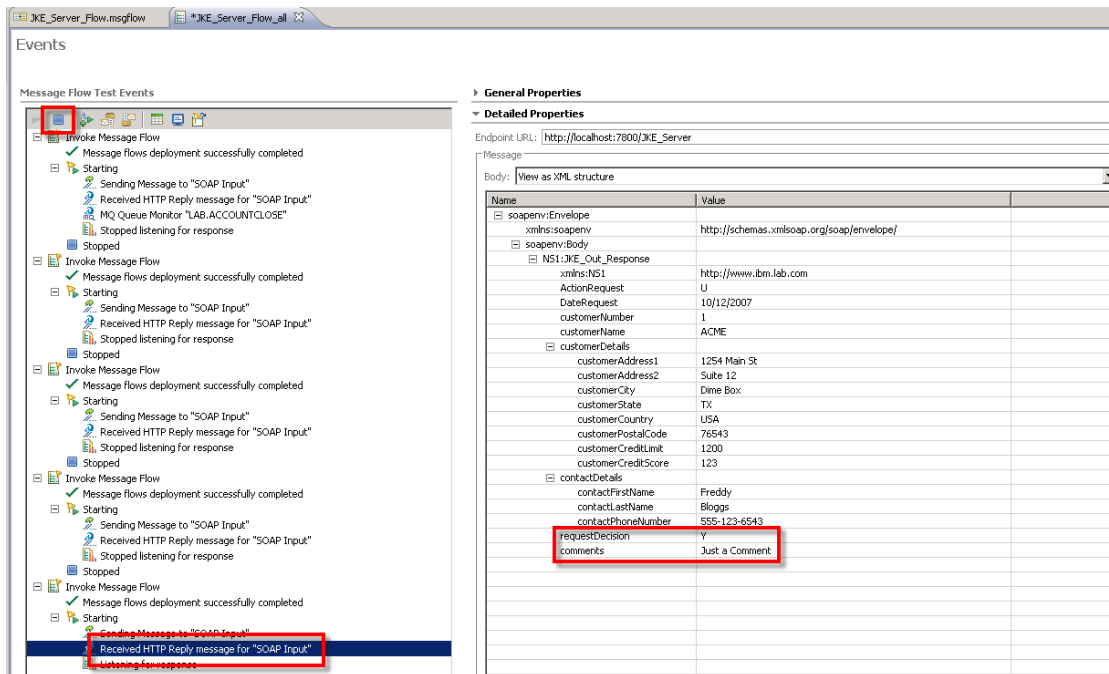
__109. Click on the third **Sending Message to "Input"** entry.

__110. Verify that it is for the Account Update request, as shown above.



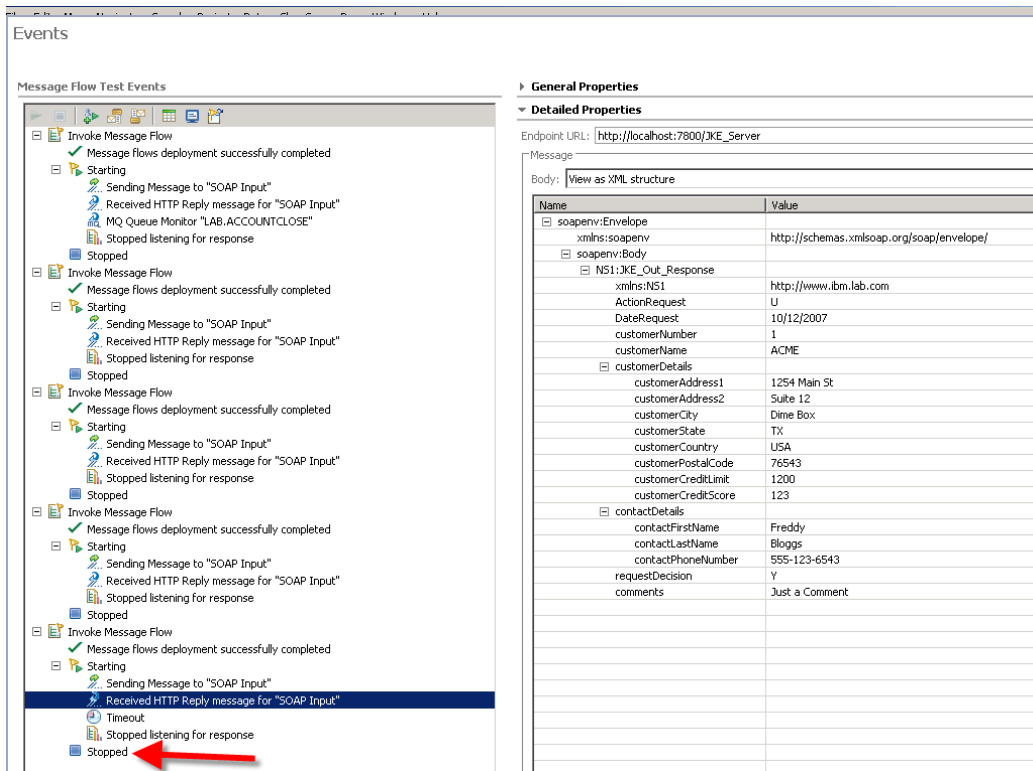
__111. Right-click on the third **Invoke Message Flow** entry.

__112. Select **Re-run**.

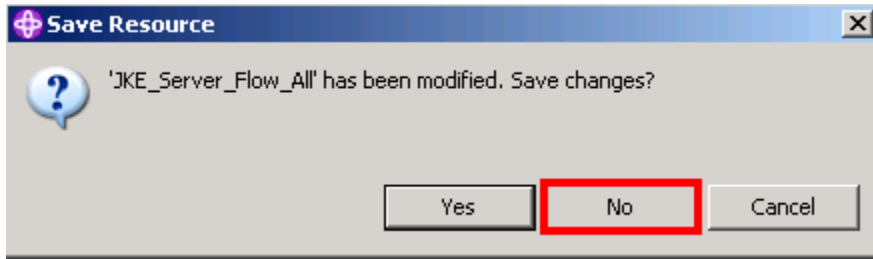


A response is returned to the Web Services client and the Test Client continues to wait for a response on the LAB.ACCOUNTCLOSE queue.

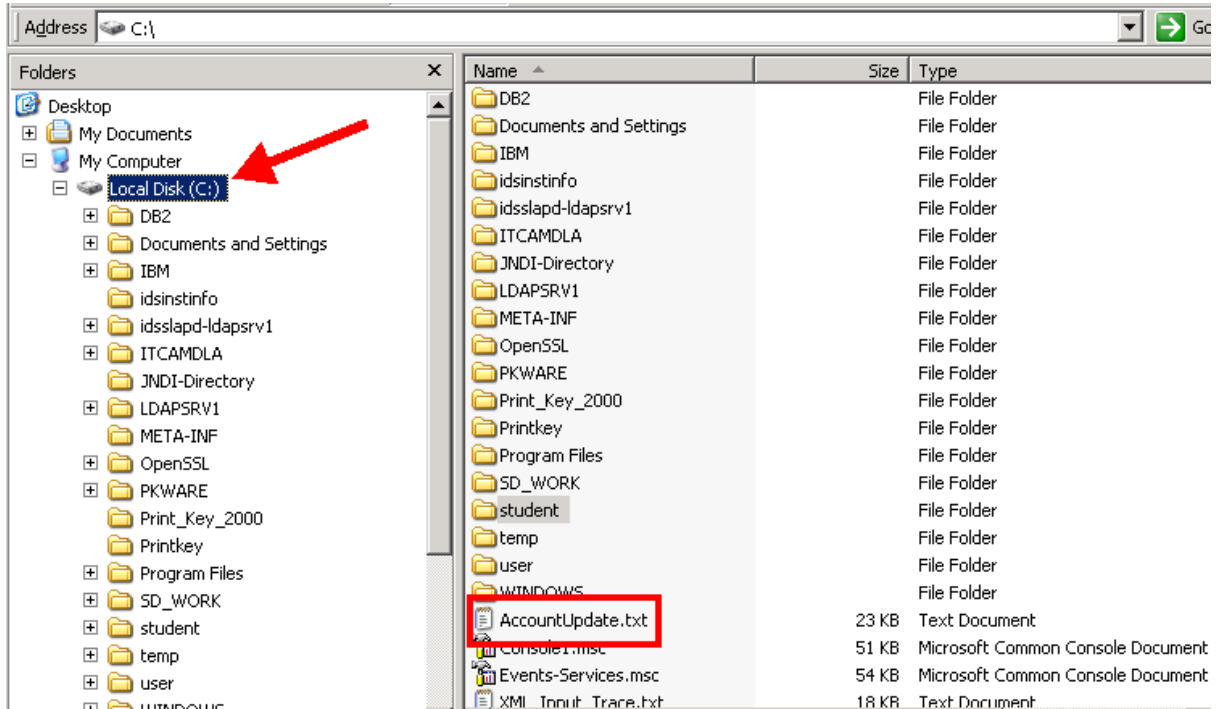
__113. Either allow the test Client to timeout or use the Stop icon in the upper left.



__114. Ensure that the test has stopped.



__115. Close, but do not save, the Test Client.



__116. Double-click the **AccountUpdate.txt** trace file.

There should be two additional entries, one for Account_Open and one for Account_Update. The new entries should be at the end of the trace file.

```

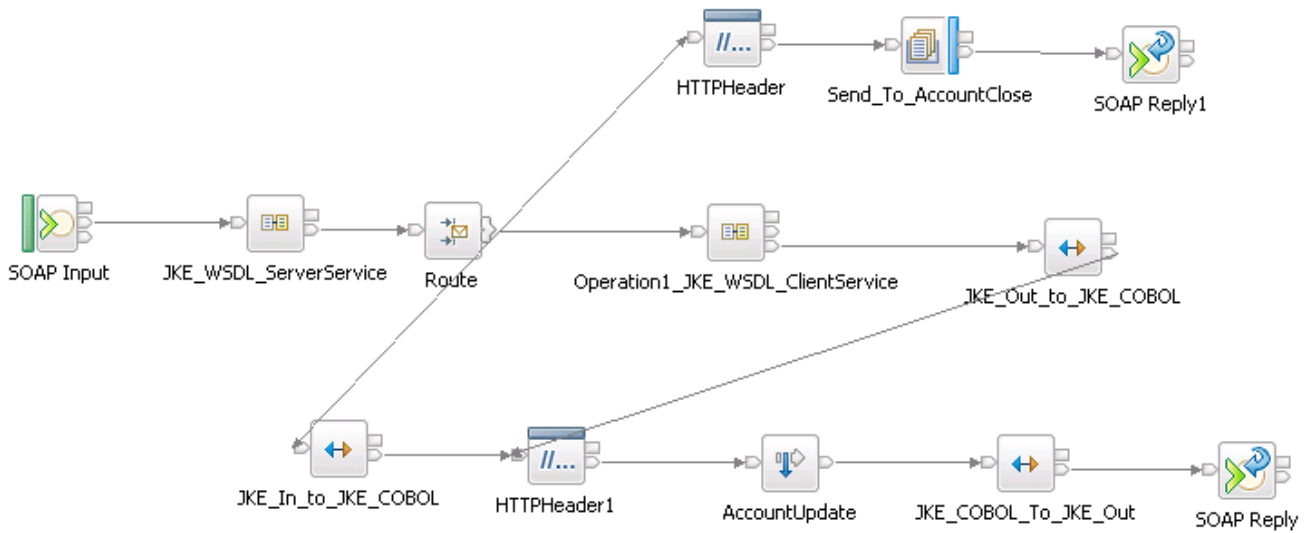
AccountUpdate.txt - Notepad
File Edit Format View Help
Message: ( ['SOAPRoot' : 0x77396f8]
(0x01000000:Name):Properties = ( ['MQPROPERTYPARSER' : 0x32499f0]
(0x03000000:NameValue):MessageSet = '01CE55002002' (CHARACTER)
(0x03000000:NameValue):MessageType = '{':msg_JKEACCOUNTREQUEST' (CHARACTER)
(0x03000000:NameValue):MessageFormat = 'Binary1' (CHARACTER)
(0x03000000:NameValue):Encoding = 'utf-8' (CHARACTER)
(0x03000000:NameValue):codedcharsetid = 1208 (INTEGER)
(0x03000000:NameValue):Transactional = FALSE (BOOLEAN)
(0x03000000:NameValue):Persistence = FALSE (BOOLEAN)
(0x03000000:NameValue):CreationTime = GMTTIMESTAMP '2009-01-20 19:28:59.477' (GMTTIMESTAMP)
(0x03000000:NameValue):ExpirationTime = -1 (INTEGER)
(0x03000000:NameValue):Priority = 0 (INTEGER)
(0x03000000:NameValue):ReplyIdentifier = X'0000000000000000000000000000000000000000000000000000000000000000' (BLOB)
(0x03000000:NameValue):ReplyProtocol = 'SOAP-AXIS2' (CHARACTER)
(0x03000000:NameValue):Topic = NULL
(0x03000000:NameValue):ContentType = 'text/xml; charset=utf-8' (CHARACTER)
(0x03000000:NameValue):IdentitySourceType = '' (CHARACTER)
(0x03000000:NameValue):IdentitySourceToken = '' (CHARACTER)
(0x03000000:NameValue):IdentitySourcePassword = '' (CHARACTER)
(0x03000000:NameValue):IdentitySourceIssuedBy = '' (CHARACTER)
(0x03000000:NameValue):IdentityMappedType = '' (CHARACTER)
(0x03000000:NameValue):IdentityMappedToken = '' (CHARACTER)
(0x03000000:NameValue):IdentityMappedPassword = '' (CHARACTER)
(0x03000000:NameValue):IdentityMappedIssuedBy = '' (CHARACTER)
)
(0x01000021:Name+):MRM = ( ['mrm' : 0x74f8d38]
(0x03000000:NameValue):ACTION_REQUEST = 'U' (CHARACTER)
(0x03000000:NameValue):DATE_REQUEST = '10/12/2007' (CHARACTER)
(0x03000000:NameValue):CUSTOMER_ACCOUNT_NUMBER = '1' (CHARACTER)
(0x03000000:NameValue):CUSTOMER_COMPANY_NAME = 'ACME' (CHARACTER)
(0x01000000:Name):CUSTOMER_DETAILS = (
(0x03000000:NameValue):ADDRESS1 = '1254 Main St' (CHARACTER)
(0x03000000:NameValue):ADDRESS2 = 'Suite 12' (CHARACTER)
(0x03000000:NameValue):CITY = 'Dime Box' (CHARACTER)
(0x03000000:NameValue):STATE = 'TX' (CHARACTER)
(0x03000000:NameValue):COUNTRY = 'USA' (CHARACTER)
(0x03000000:NameValue):POSTALCODE = '76543' (CHARACTER)
(0x03000000:NameValue):CREDIT_LIMIT = '1200' (CHARACTER)
(0x03000000:NameValue):CREDIT_SCORE = '123' (CHARACTER)
)
(0x03000000:NameValue):CONTACT_FIRST_NAME = 'Freddy' (CHARACTER)
(0x03000000:NameValue):CONTACT_LAST_NAME = 'Bloggs' (CHARACTER)
(0x03000000:NameValue):CONTACT_PHONE = '555-123-6543' (CHARACTER)
(0x03000000:NameValue):COMMENTS = 'Just a comment' (CHARACTER)
(0x03000000:NameValue):DECISION = 'Y' (CHARACTER)
)
)
)

```

Here is a trace view of the message tree after it has been “reshaped” for the output COBOL message. The element names reflect those of the COBOL copybook that was imported to define the message. This message tree is now ready to be serialized into a COBOL bit stream. But it is important to understand that the data is still physical neutral here in the message tree.

___117. Close the Notepad session.

___118. Minimize the Windows Explorer session.



This is the completed message flow. In the next lab, nodes will be added to allow the message flow to pass an Account_Open and an Account_Update message to CICS.

This is the end of Lab 8

Appendix A. Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have

been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental. All references to fictitious companies or individuals are used for illustration purposes only.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Appendix B. Trademarks and copyrights

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM	AIX	CICS	ClearCase	ClearQuest	Cloudscape
Cube Views	DB2	developerWorks	DRDA	IMS	IMS/ESA
Informix	Lotus	Lotus Workflow	MQSeries	OmniFind	
Rational	Redbooks	Red Brick	RequisitePro	System i	
<i>System z</i>	<i>Tivoli</i>	<i>WebSphere</i>	<i>Workplace</i>	<i>System p</i>	

Adobe, Acrobat, Portable Document Format (PDF), and PostScript are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, other countries, or both.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both. See Java Guidelines

Microsoft, Windows, Windows NT, and the Windows logo are registered trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

ITIL is a registered trademark and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

Other company, product and service names may be trademarks or service marks of others.



© Copyright IBM Corporation 2011.

The information contained in these materials is provided for informational purposes only, and is provided AS IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, these materials. Nothing contained in these materials is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software. References in these materials to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. This information is based on current IBM product plans and strategy, which are subject to change by IBM without notice. Product release dates and/or capabilities referenced in these materials may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way.

IBM, the IBM logo and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at ibm.com/legal/copytrade.shtml

Other company, product and service names may be trademarks or service marks of others.



Please Recycle
