Lab 1 Getting Started

1.1 Building and Executing a Simple Message Flow

In this lab, you will build and execute a simple message flow. A message flow is like a program but is developed using a visual paradigm.

The flow will be deployed to an Integration Server in an Integration Node where it will execute. An Integration Server is an operating system process where user flows execute. The flow will then be available to process messages. There is no need to restart the Integration Node or the Integration Server 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". It is essentially a zip file with a deployment descriptor. The deployment descriptor allows certain properties of the message flow to be overridden.

The Broker Archive file will hold the artifacts to be deployed to a specific Integration Server in a specific Integration Node. It may contain applications and libraries as well as message flows, message sets, XSL Transformations (XSLT) Style Sheets, Java™ Archive (JAR) files, XML schema files or WSDL 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 Integration Node. The final validation of the artifacts is done by the Integration Node. If errors are found by the Integration Node, they will be reported back in the Event Log.

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.

NOTE: This lab assumes that the IBM Integration Bus Default Configuration has been created.

The online tutorial

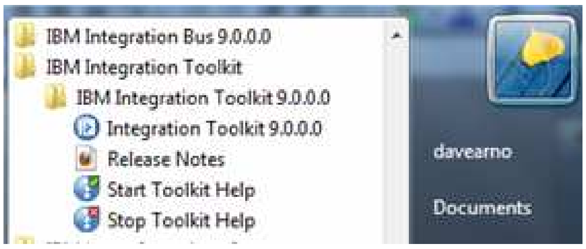
"3.IIBv9.0_Create_Default_Configuration_And_First_Integration"

demonstrates how to do this.

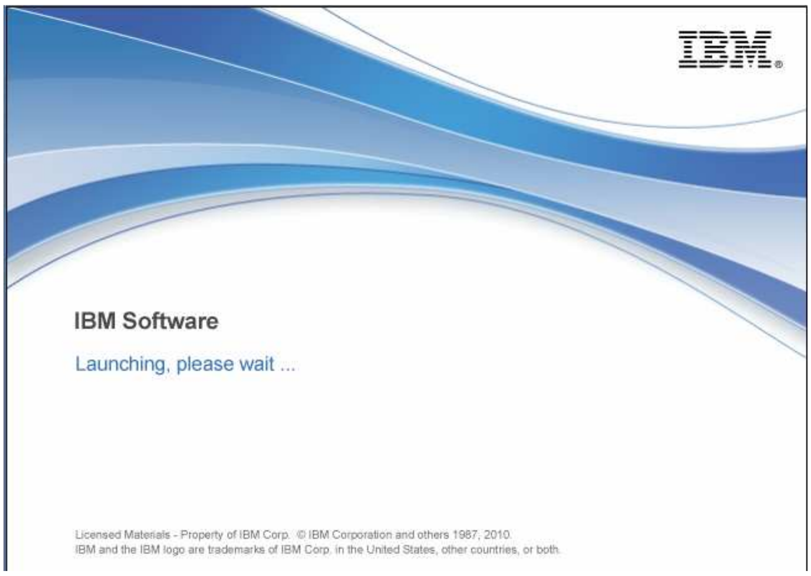
Getting to know the Toolkit

The icon for the IBM Integration Toolkit is located in the system tray on the desktop. In later labs, you will also be using some of the icons that have been placed on the smart bar.

__1. **Click on All Programs->**IBM Integration Toolkit 9.0.0.0 to start the toolkit

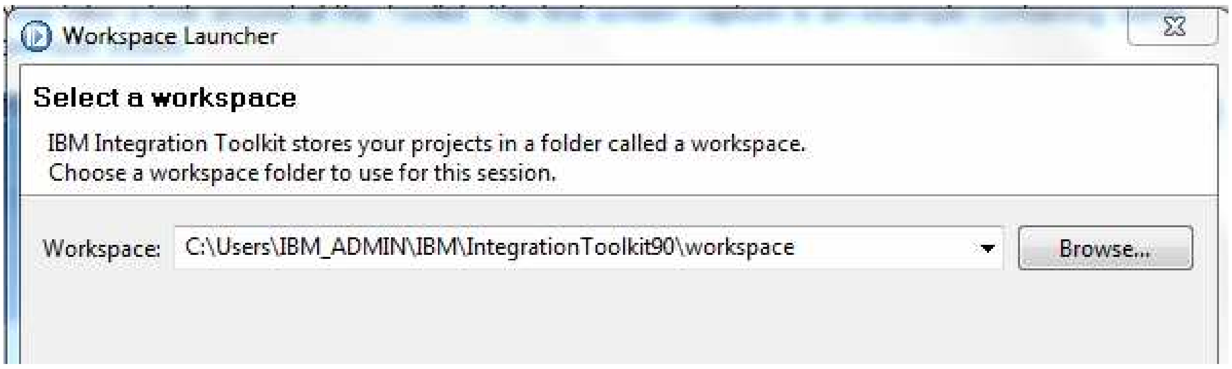


The splash screen is displayed when starting the IBM Integration Toolkit.


IBM.
IBM Software
Launching, please wait ...
Licensed Materials - Property of IBM Corp. © IBM Corporation and others 1987, 2010.
IBM and the IBM logo are trademarks of IBM Corp. in the United States, other countries, or both.

You are prompted to select or create an Eclipse Workspace

__2. The default is quite a lengthy path



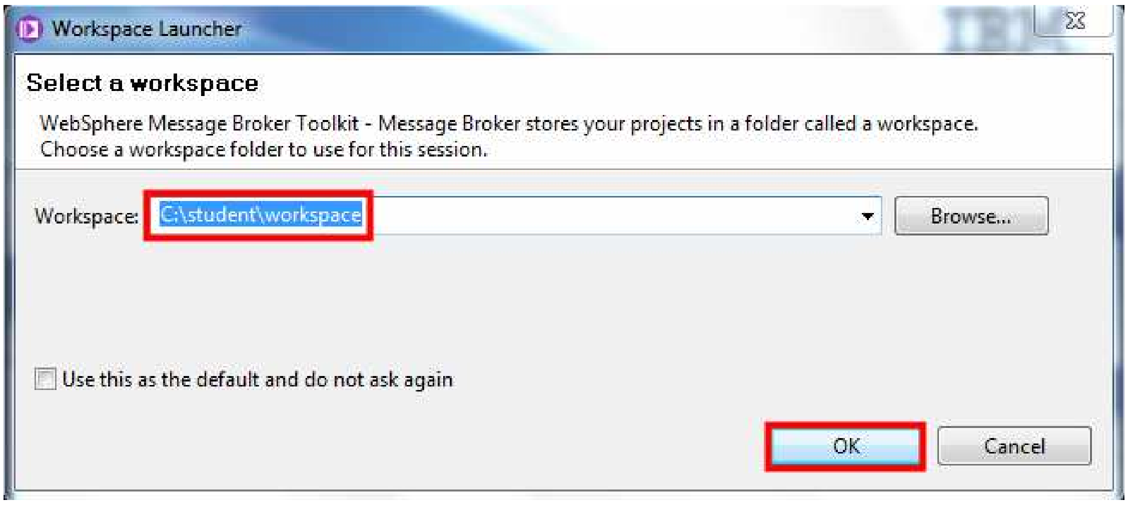
Select a workspace

IBM Integration Toolkit stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: C:\Users\IBM_ADMIN\IBM\IntegrationToolkit90\workspace

__3. Use **C:\student\workspace** directory or one of your own choosing.

__4. Click **OK**.



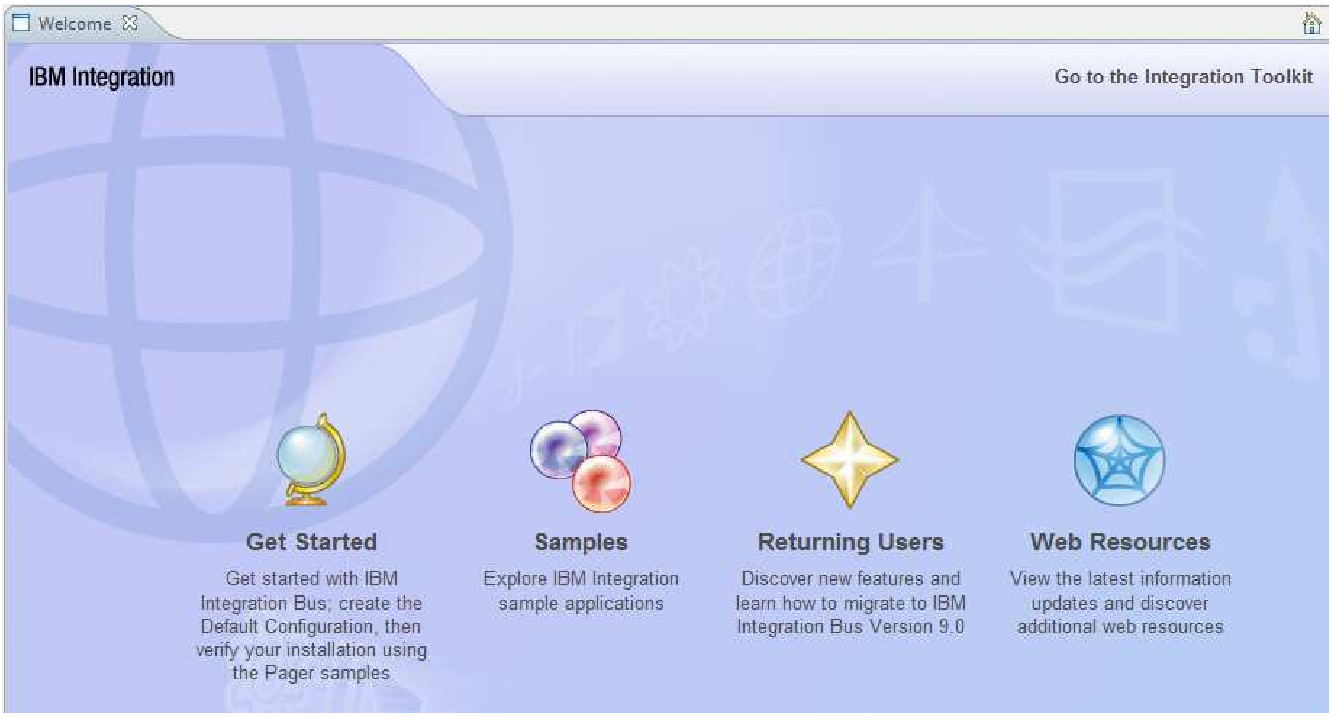
Select a workspace

WebSphere Message Broker Toolkit - Message Broker stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: C:\student\workspace

☐ Use this as the default and do not ask again

If this is the first time you have opened the IIB toolkit you will be presented with the following welcome screen


Welcome
IBM Integration
Go to the Integration Toolkit
Get Started
Get started with IBM Integration Bus; create the Default Configuration, then verify your installation using the Pager samples
Samples
Explore IBM Integration sample applications
Returning Users
Discover new features and learn how to migrate to IBM Integration Bus Version 9.0
Web Resources
View the latest information updates and discover additional web resources

If you have **<u>not</u>** created the default configuration, go in the Getting Started and kick off the wizard before proceeding.

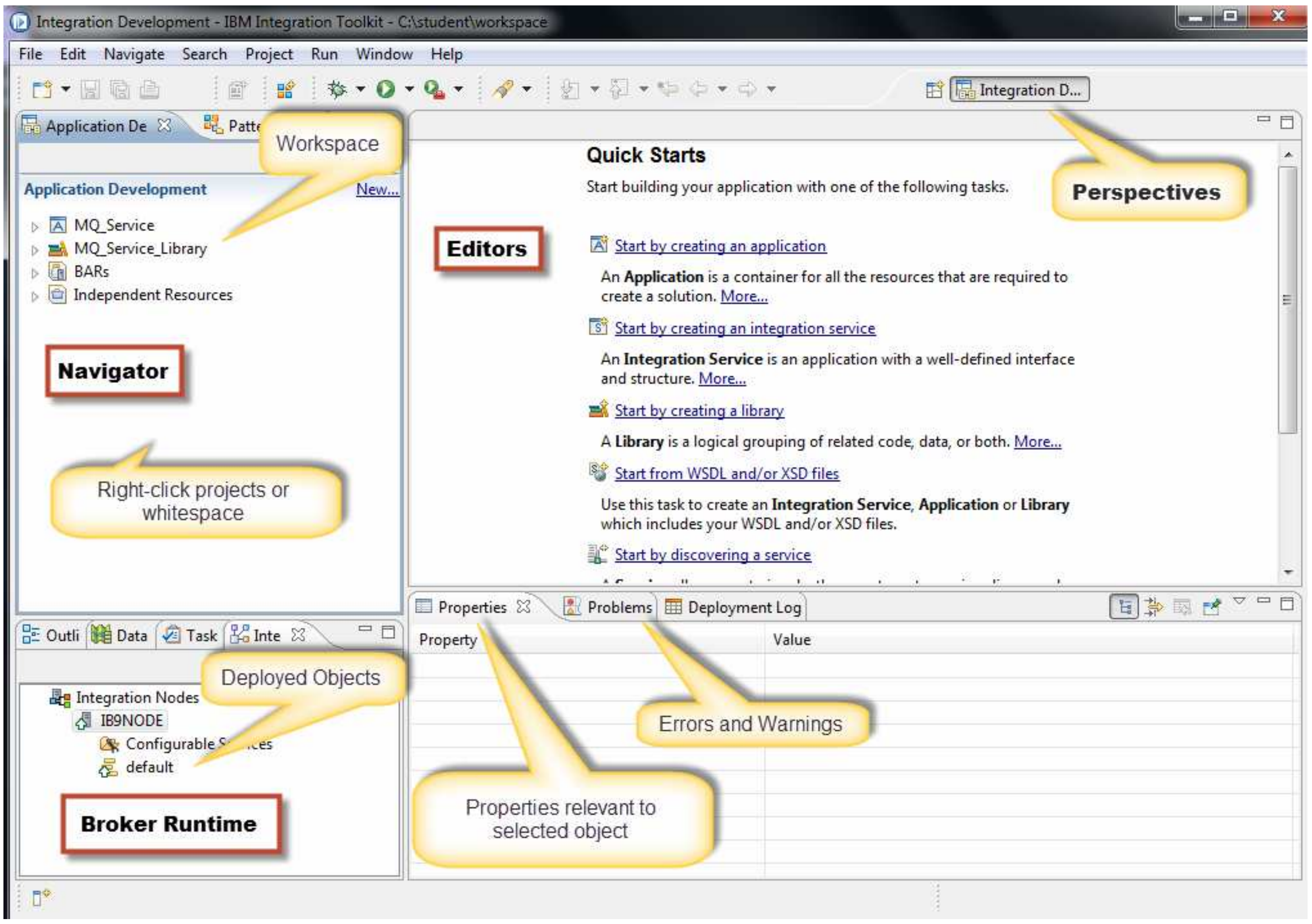
The online tutorial

“3.IIBv9.0_Create_Default_Configuration_And_First_Integration”

demonstrates how to do this.

Otherwise, click on “Go to the Integration Toolkit”

__5. Now take a look around at the Toolkit. The first screen capture is an example containing some pre-built assets.



Quick Starts

Start building your application with one of the following tasks.

Start by creating an application

An **Application** is a container for all the resources that are required to create a solution. More...

Start by creating an integration service

An **Integration Service** is an application with a well-defined interface and structure. More...

Start by creating a library

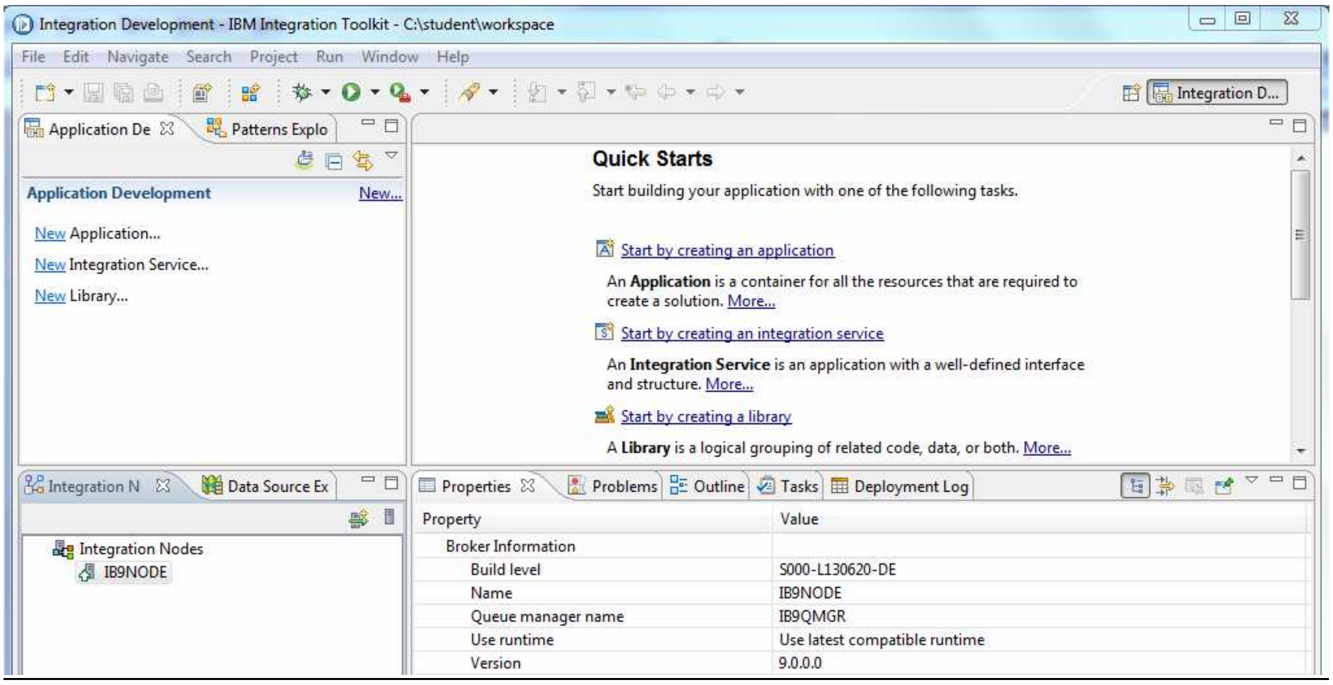
A **Library** is a logical grouping of related code, data, or both. More...

Start from WSDL and/or XSD files

Use this task to create an **Integration Service**, **Application** or **Library** which includes your WSDL and/or XSD files.

Start by discovering a service

An new workspace with just the default configuration created will look as follows.


Integration Development - IBM Integration Toolkit - C:\student\workspace

File Edit Navigate Search Project Run Window Help

Application Development New...

New Application...
New Integration Service...
New Library...

Quick Starts

Start building your application with one of the following tasks.

Start by creating an application

An **Application** is a container for all the resources that are required to create a solution. More...

Start by creating an integration service

An **Integration Service** is an application with a well-defined interface and structure. More...

Start by creating a library

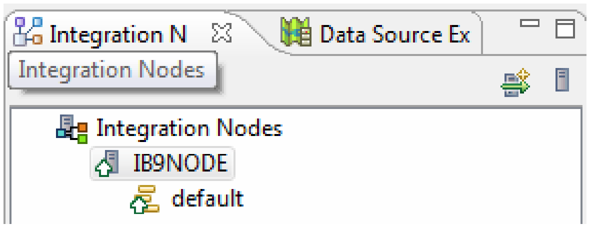
A **Library** is a logical grouping of related code, data, or both. More...

Integration Nodes
IB9NODE

Property	Value
Broker Information	
Build level	S000-L130620-DE
Name	IB9NODE
Queue manager name	IB9QMGR
Use runtime	Use latest compatible runtime
Version	9.0.0.0

Right Click on the IB9NODE in the bottom left pane and select refresh.

If the “Default Configuration” has been created the “default” Integration Server (or Execution Group) will have been created.



Key Idea: The Toolkit

The Integration Toolkit is based on Eclipse and includes components from IBM Rational® Application Developer. It provides one Perspective specifically for IBM Integration Bus 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.

The screenshot on the previous page is of the **Integration 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 (**Integration Development**) and patterns (**Patterns Explorer**). The Navigator view contains the projects that are available within the Eclipse workspace. There is a set of resources already provided for your use during the labs.

The area below the navigator view is the summary area. The **Integration Nodes** tab will show all defined local nodes as well as connections to remote nodes 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 are displayed as links in the center of the screen, which is the default blank palette within the Editor view. You can also access these by right clicking whitespace in the Applications view on the left. The wizards do some of the initial work for creating various types of solutions.

The Integration Toolkit provides several **Quick Start** wizards plus a number of pre-defined patterns to assist in creating Applications and Libraries. We will see more of these in later labs.

Eclipse is project oriented – artifacts are organized into projects. A project is typed. Project types that are specific to IBM Integration Bus are Applications and Libraries. Also, legacy projects of type Message Flow Project and Message Set Project can be created or imported into the toolkit. Since they pre-date the concept of Applications and Libraries, they will be visible in the hierarchy under a Folder called “Independent Resources”.

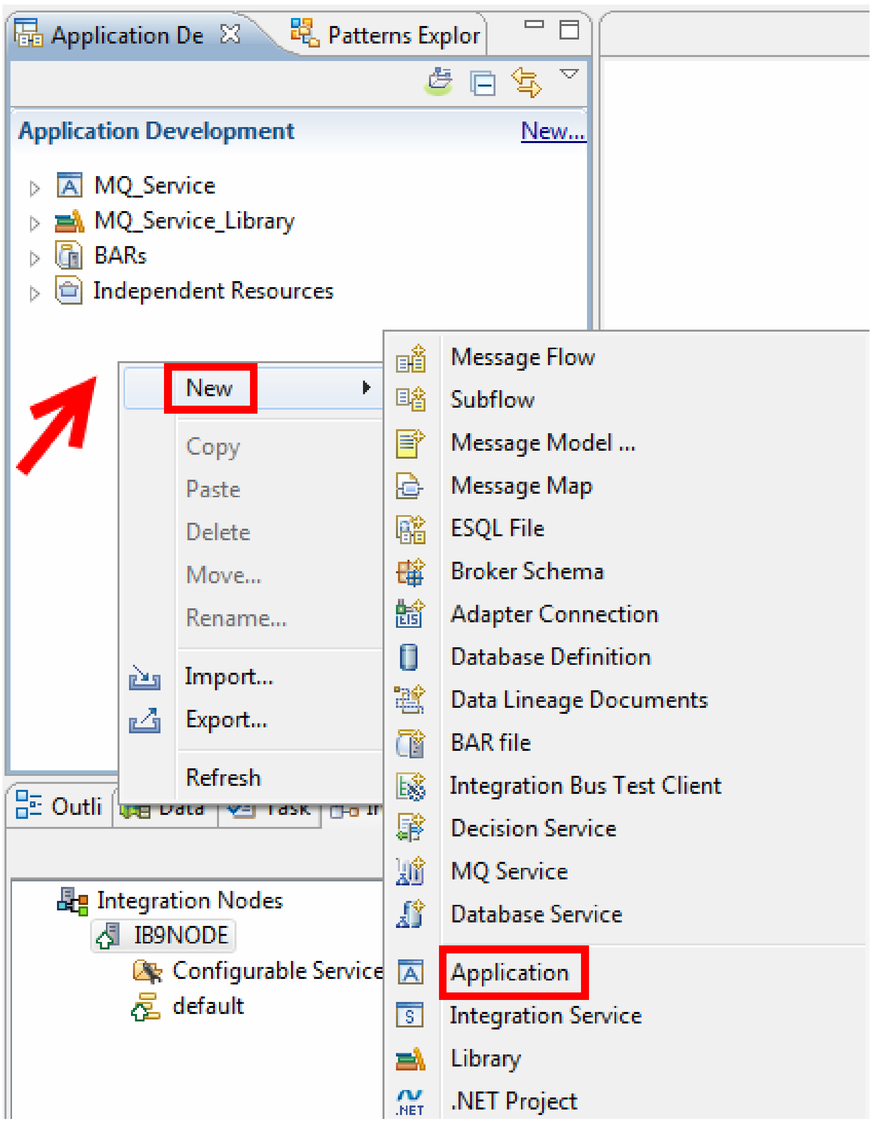
1.2 Building a simple flow

__1. Right click in the white space of the Application Development pane.

__2. Select **New →Application**.

As an alternative, you can click **File** on the Menu Bar, and select **New→Other**, type **Application** in the selection box, and click **Next**.

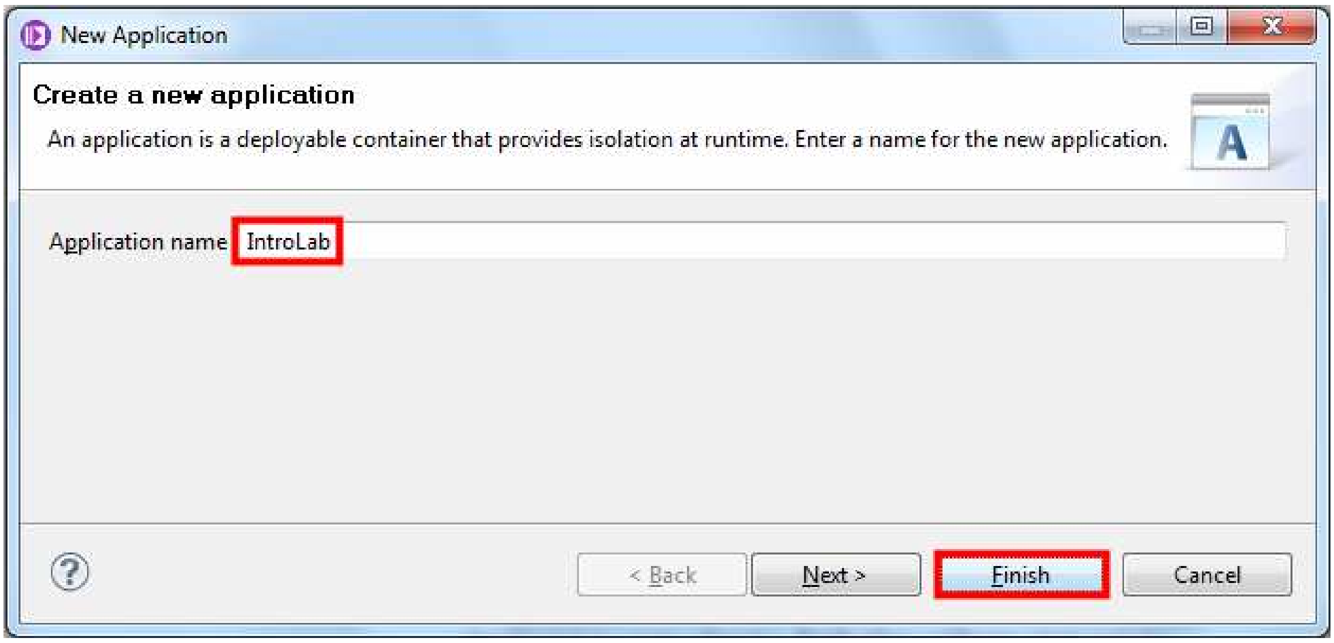
Note: These actions are also available as icons on the toolbar.



You are prompted to enter a name for your Application.

__3. Enter **IntroLab** for the Application name.

__4. Click **Finish**.


New Application

Create a new application

An application is a deployable container that provides isolation at runtime. Enter a name for the new application.

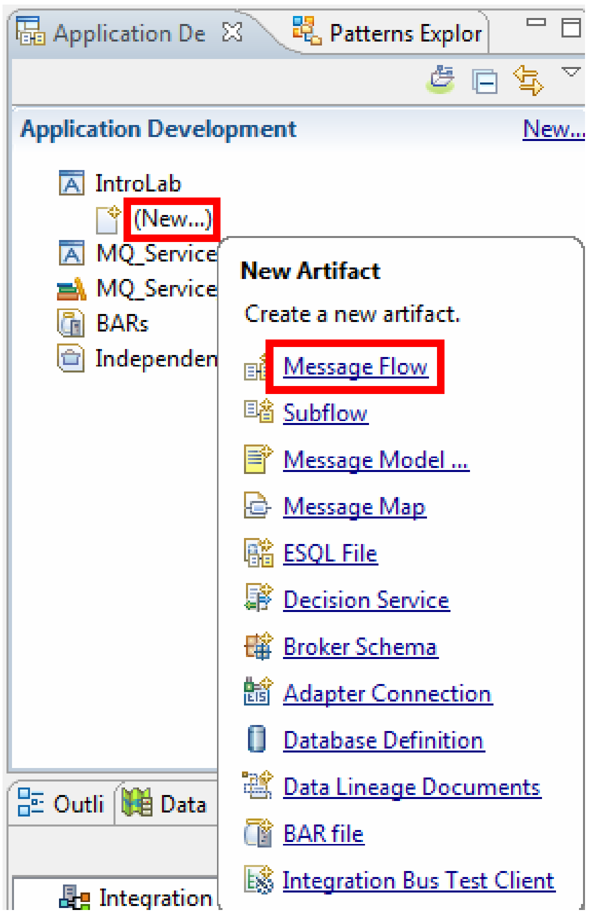
Application name: IntroLab

Now you will create a new message flow.

__5. Under the **IntroLab**, click on **(New…)**.

__6. Select **Message Flow**.

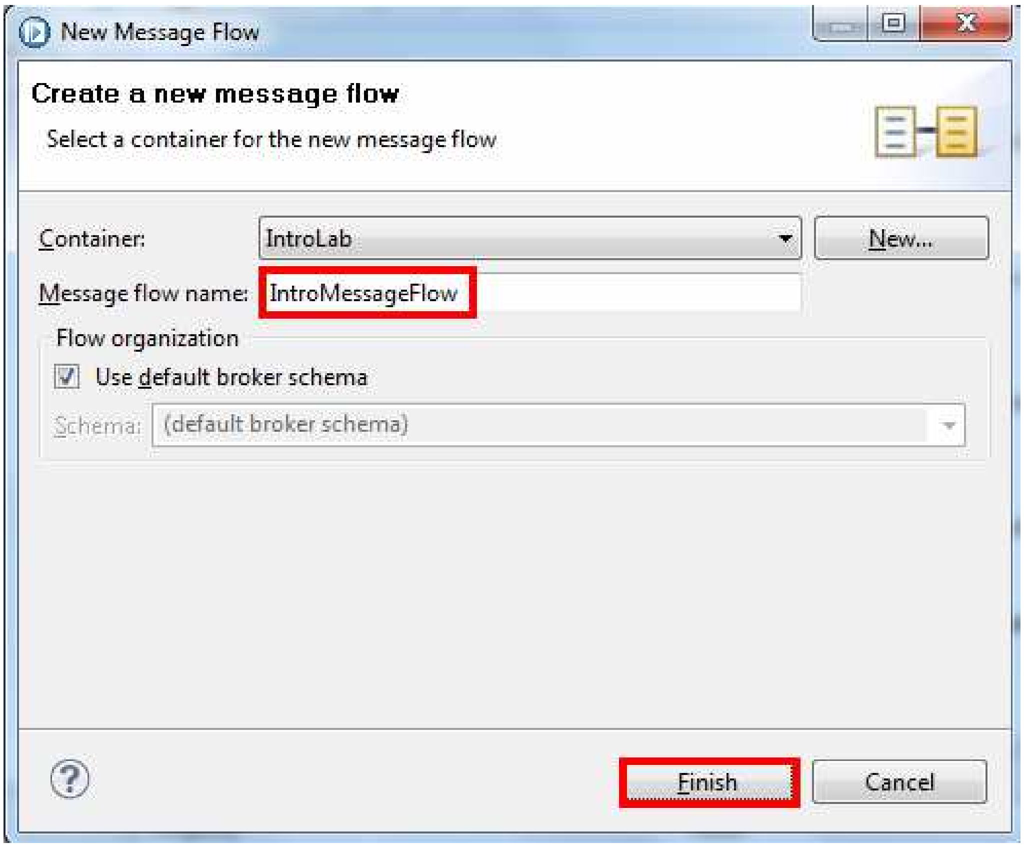
The options for the New action will be different depending on how the request is made. For example, when using the File → New from the Menu bar, all of the options will be listed. However, in this case, by starting from an Application, the only options shown are those that are related to the selected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



Here you are asked to name the message flow. An **Application** may contain multiple message flows.

__7.   Enter **IntroMessageFlow** in the Message flow name box.

__8.   Click **Finish**.



New Message Flow

Create a new message flow

Select a container for the new message flow

Container: IntroLab

Message flow name: IntroMessageFlow

Flow organization

[✓] Use default broker schema

Schema: (default broker schema)

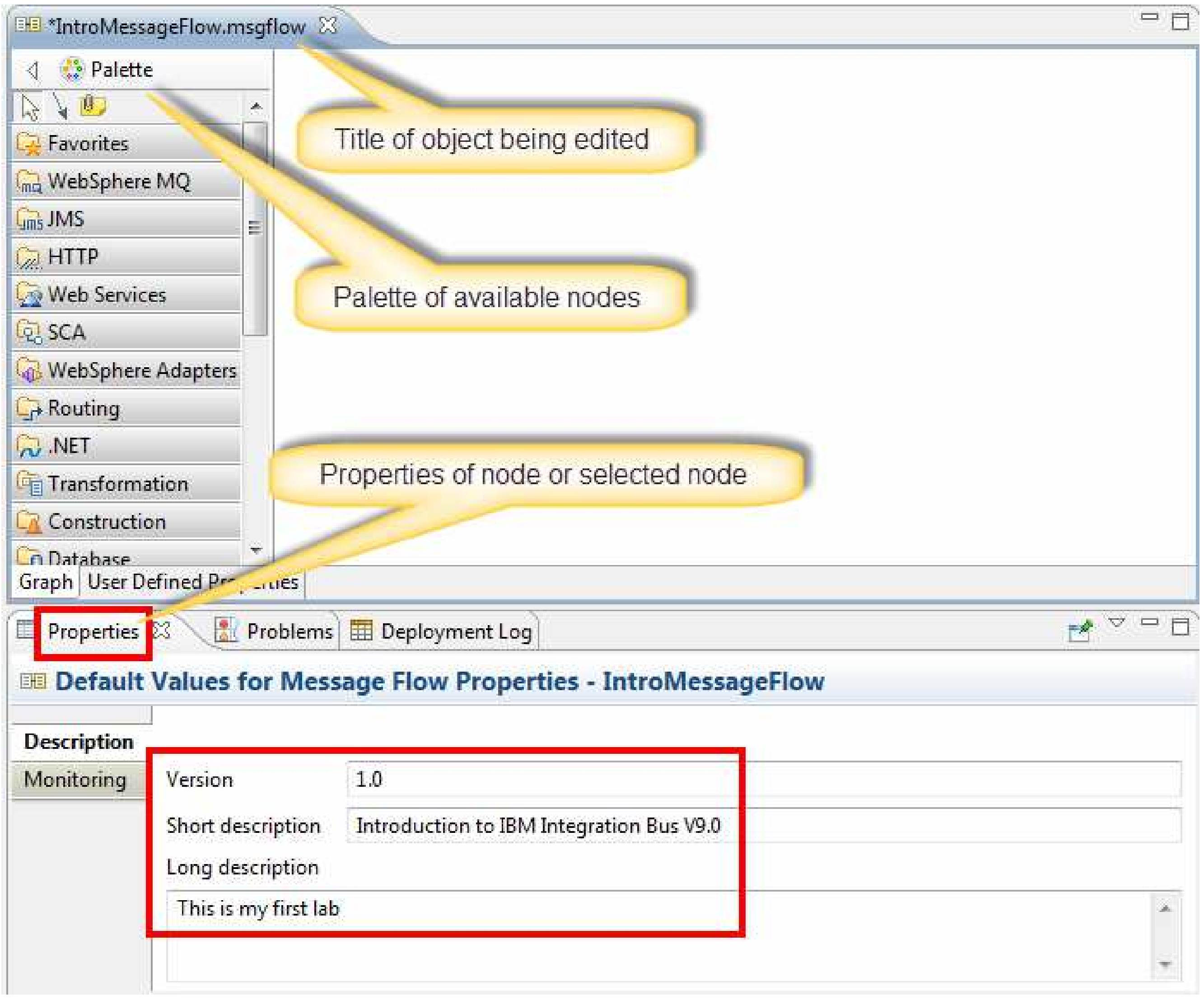
You are placed into the Message Flow Editor where you can compose the message flow. When you click on the Message Flow Editor, information about the message flow appears in the Properties pane.

__9. Select the **Properties** tab.

__10. Select the **Description** tab.

__11. Enter the following:

- Enter **1.0** for the **Version** field;
- Enter **Introduction to IBM Integration Bus V9.0** for the **Short description** field;
- Your choice of information in the **Long description** field.



Default Values for Message Flow Properties - IntroMessageFlow

Version: 1.0

Short description: Introduction to IBM Integration Bus V9.0

Long description: This is my first lab

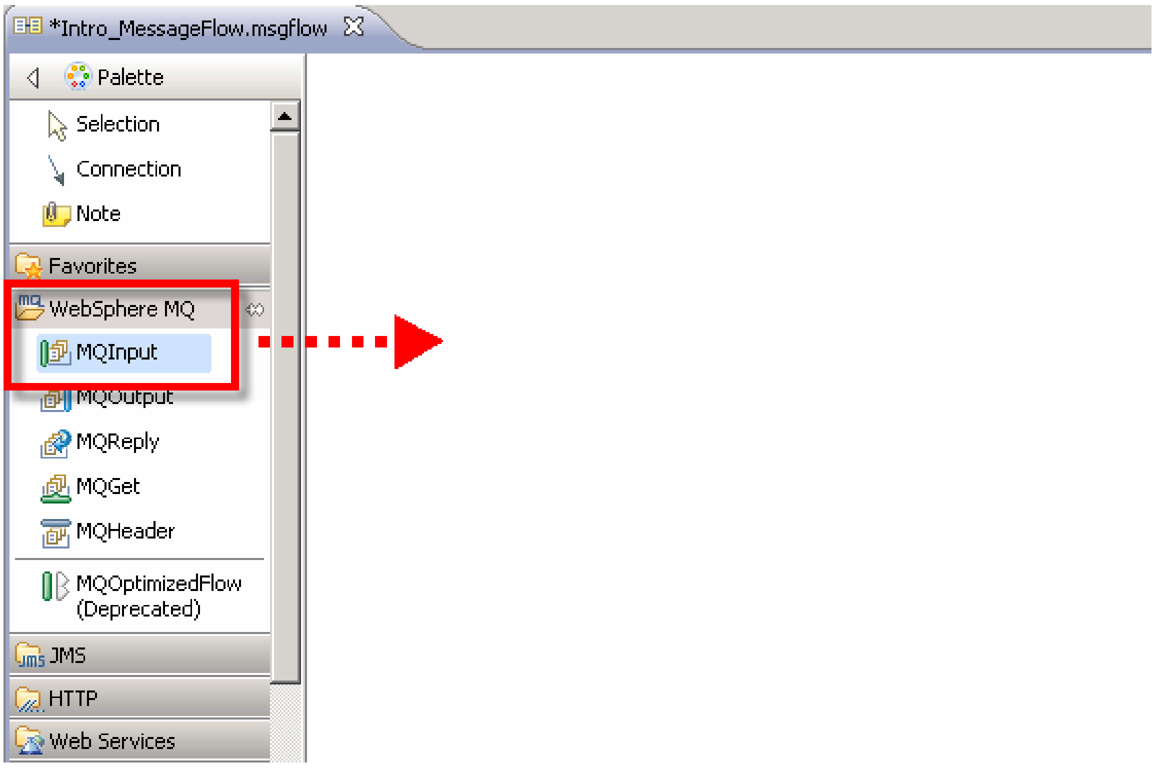
A message flow must begin with an Input node. An input node establishes the environment for the flow. There is an Input node for each of the various protocols that IBM Integration Bus supports. We will process a WebSphere MQ message with this flow so we need an MQInput node.

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.

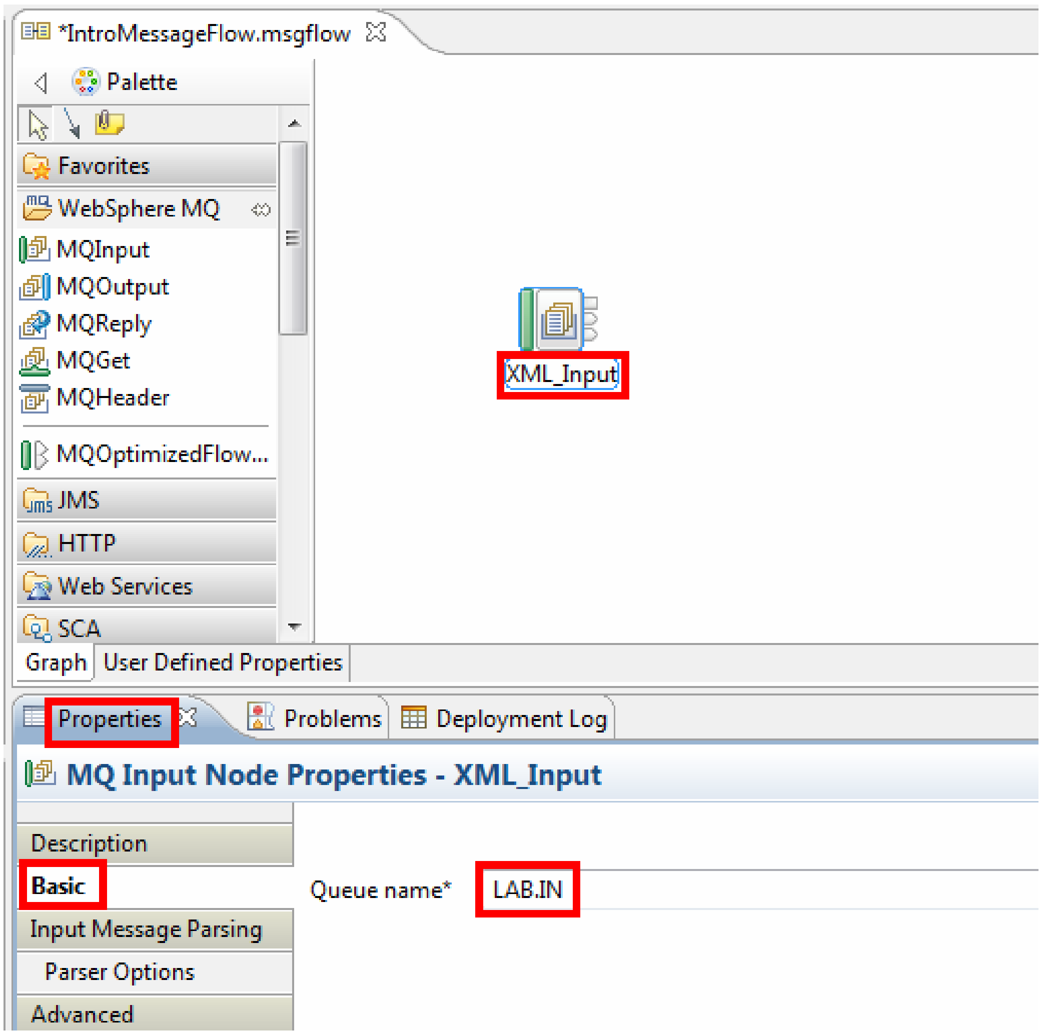
__15. Change the name of the node to **XML_Input**.

__16. Press the **Enter** key to complete the rename.

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

__17. Select the **Basic** tab in the **Properties** pane.

__18. Enter **LAB.IN** as the **Queue name**. Queue names are case sensitive. All queue names in the labs are upper case.

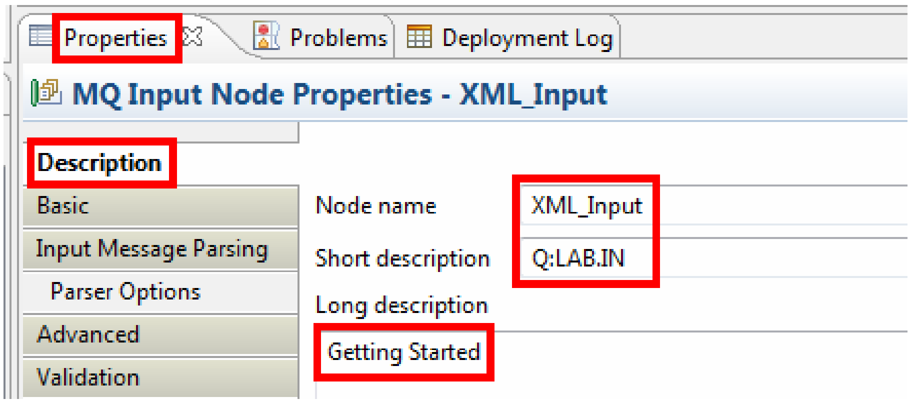


Queue name* LAB.IN

__19. Select the **Description** tab. This section is used for documentation. The name of the node may also be changed. The node name is shown in the message flow editor. It does not affect the operation of the message flow.

__20. Enter **Q:LAB.IN** in the **Short description** field.

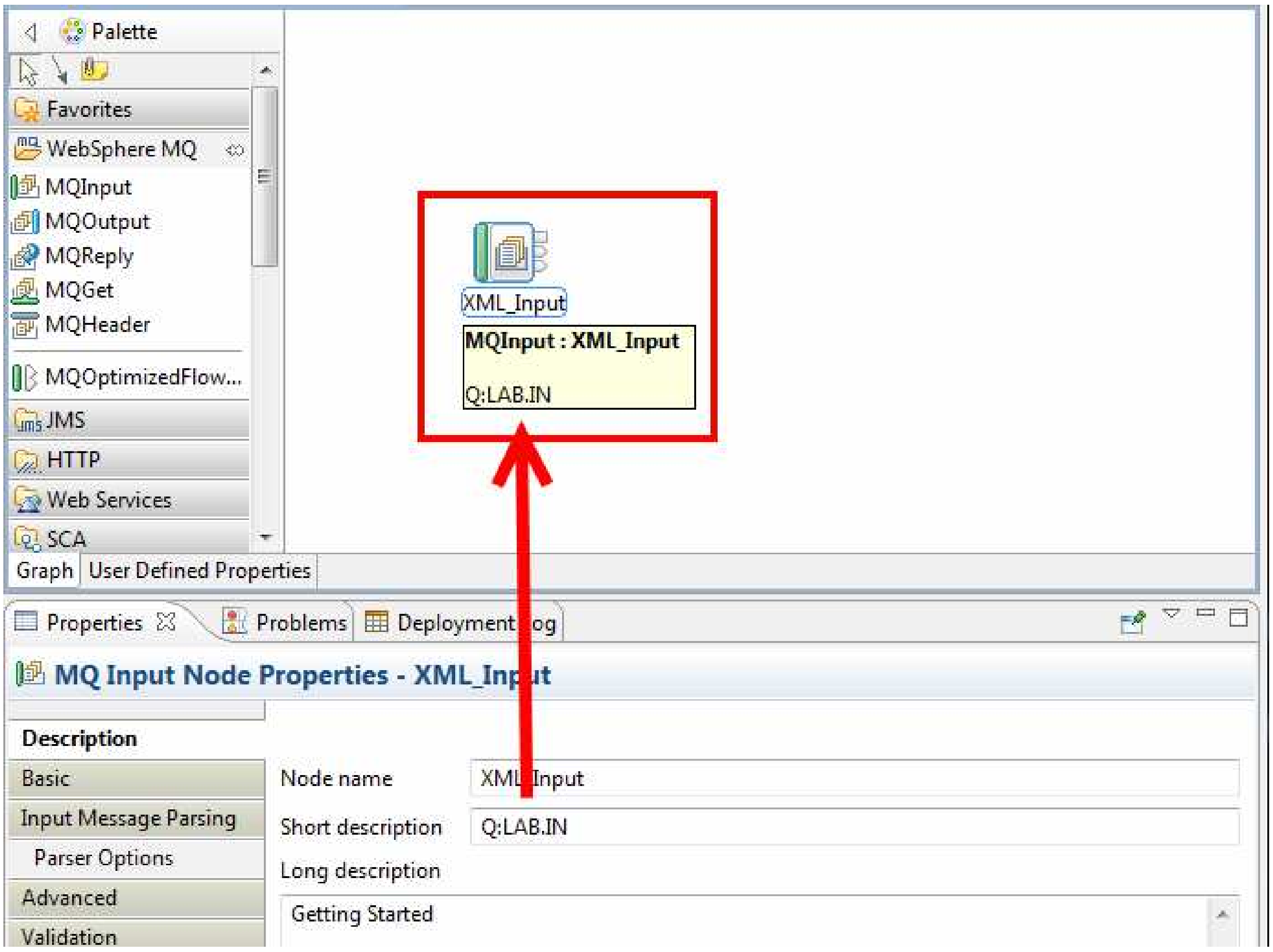
__21. Enter your choice of text for the **Long description** (**Getting Started** is shown in the screen shot).



Node name: XML_Input
Short description: Q:LAB.IN
Long description: Getting Started

__22. Hold the mouse pointer 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.



MQInput : XML_Input

Q:LAB.IN

MQ Input Node Properties - XML_Input

Node name: XML_Input

Short description: Q:LAB.IN

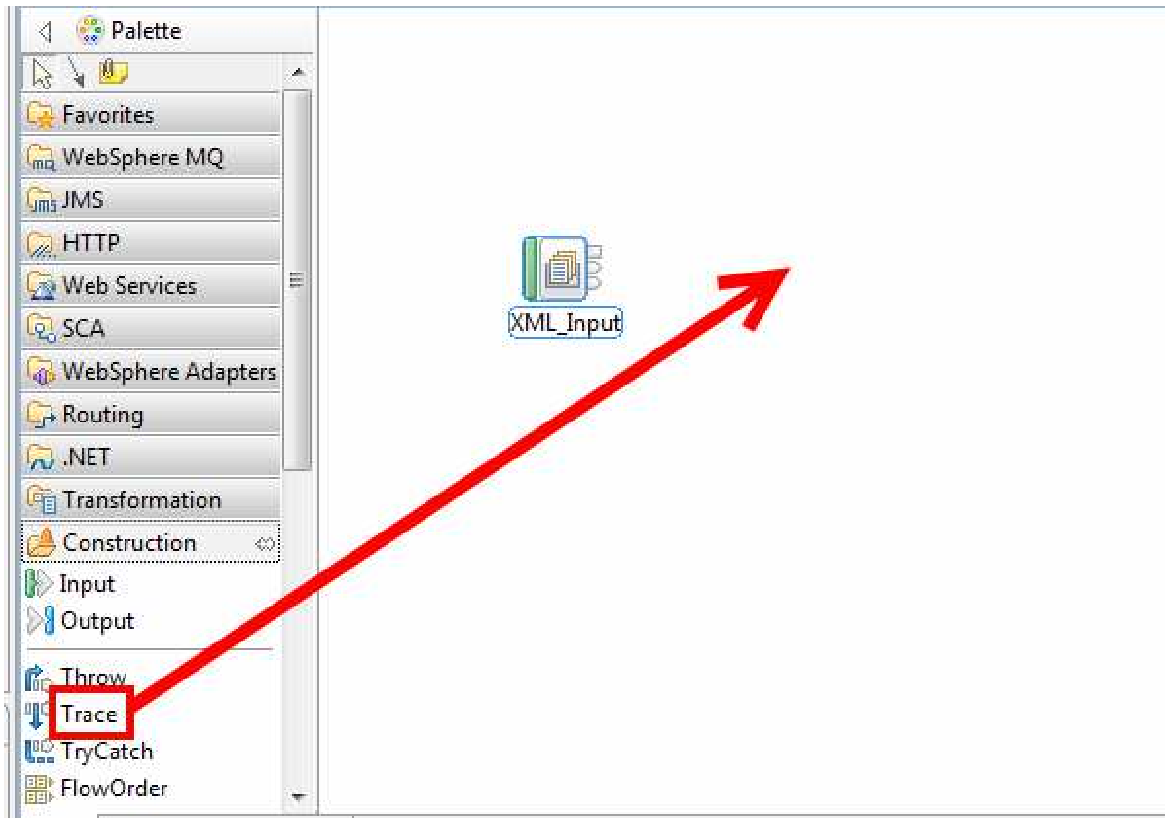
Long description: Getting Started

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

__23. Open the **Construction** drawer by clicking it with the mouse.

__24. 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.

__25. Press the **Enter** key to accept the default node name (**Trace**).



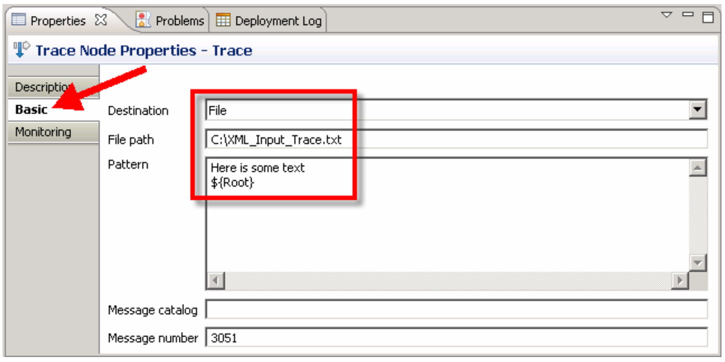
__26. In the **Basic** tab, use the pull down list on the **Destination** field to select **File**.

__27. 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.

__28. Enter a line of your choice in the **Pattern** box – no quotes are needed.

__29. 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. The expression between the curly braces will be evaluated at run time.

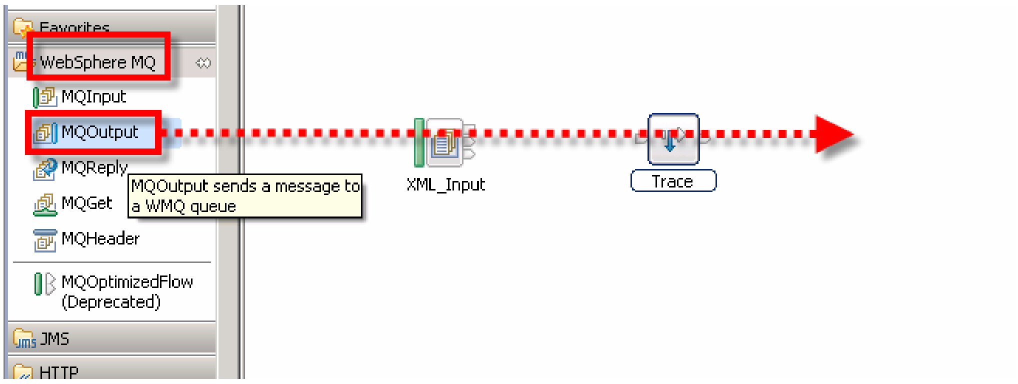


Destination File
File path C:\XML_Input_Trace.txt
Pattern Here is some text
${Root}
Message catalog
Message number 3051

__30. Open the **WebSphere MQ** drawer.

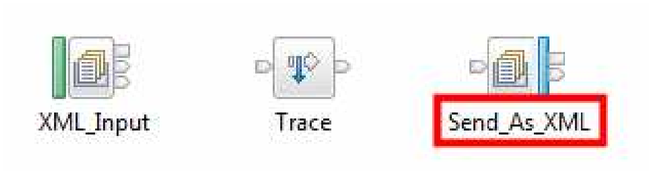
__31. Select an **MQOutput** node from the drawer.

__32. Place it to the right of the **Trace** node.



__33. While the node name is highlighted, enter **Send_As_XML** as the new name.

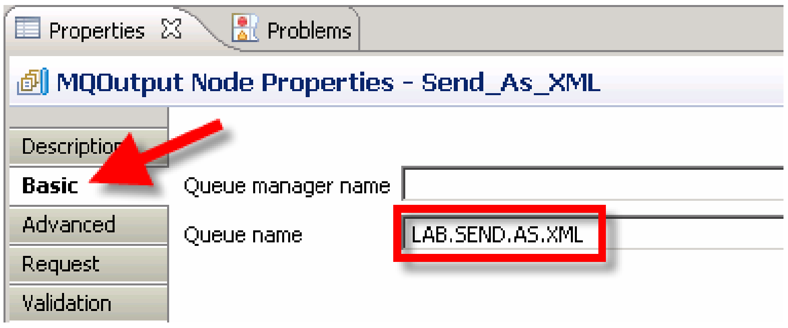
__34. Press the **Enter** key to accept the new node name.



__35. Click the **Basic** tab on the **Properties** tab.

__36. 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.

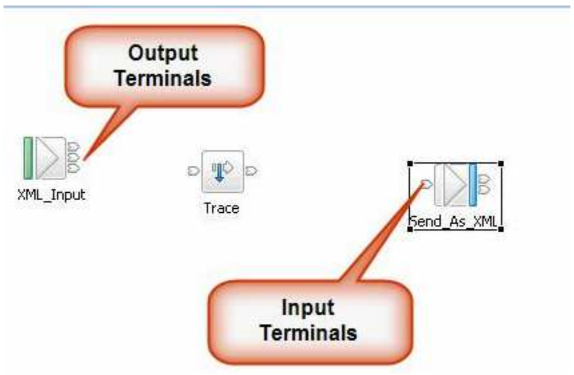
__37. Make sure you did not enter this information in the **Queue manager name** field!!



Queue manager name

Queue name LAB.SEND.AS.XML

Key Concept: 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:

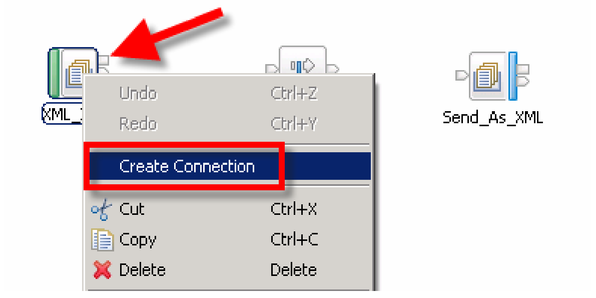
One – position the mouse over the Out terminal (in the middle), click and drag to the target and click again.

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.

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

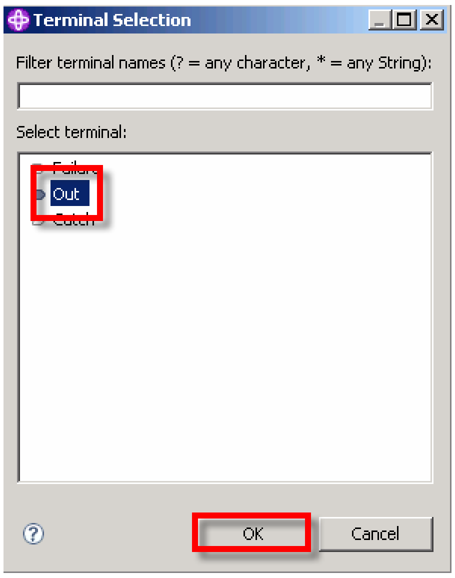
__39. Select **Create Connection** from the menu.



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

__40. Select the **Out** terminal.

__41. Click **OK.**

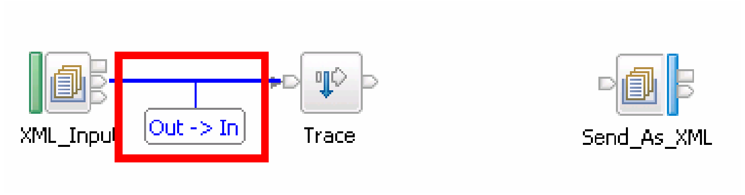


You now have a “rubber-band” connector.

__42. Position the connector over the **Trace** node.

__43. Click the left mouse button to anchor it, creating a connection between the two nodes.

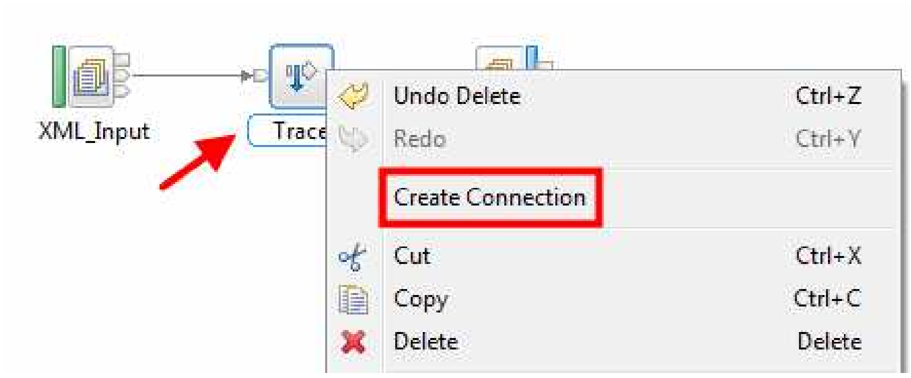
__44. Place your mouse pointer on the connection. A pop up a summary of “from-to” information will appear.



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

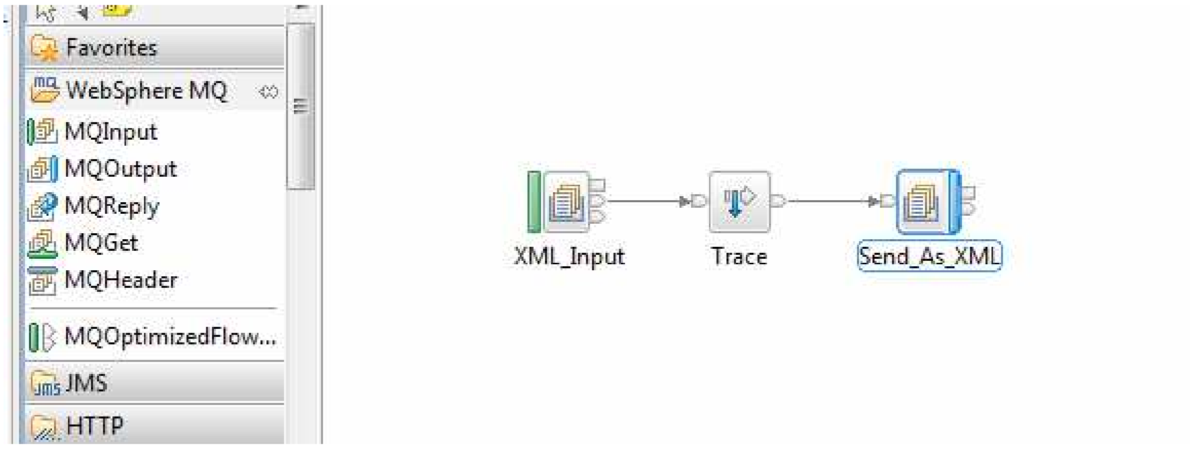
__45. Right click on the **Trace** node.

__46. Select **Create Connection**. This time you immediately get a “rubber-band” connector. No selection list of terminals is presented because there is only an **Out** terminal on the Trace node.

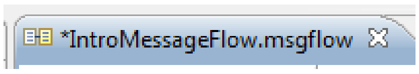


__47. Position the mouse pointer over the **Send_As_XML** node.

__48. Click the left mouse button to create the connection.



Your message flow should now look like the above diagram. Notice the small asterisk next to the message flow name. This indicates that the message flow has not been saved to disk.



__49. 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 or use **File → Save** to perform a save.

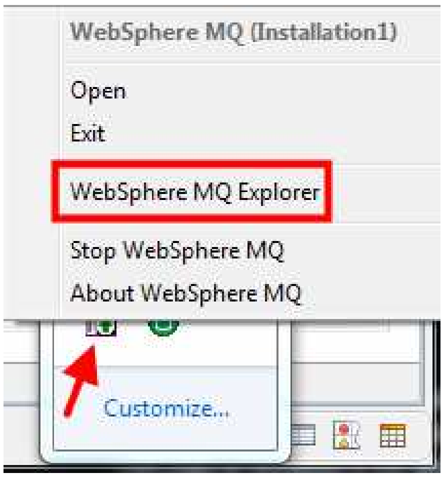
The following graphic will be used as a reminder when it is time to save your work.

The Message flow we have built uses WebSphere MQ as its Input and Output transport so we must create the MQ Input and Output queues in the WebSphere MQ Explorer.

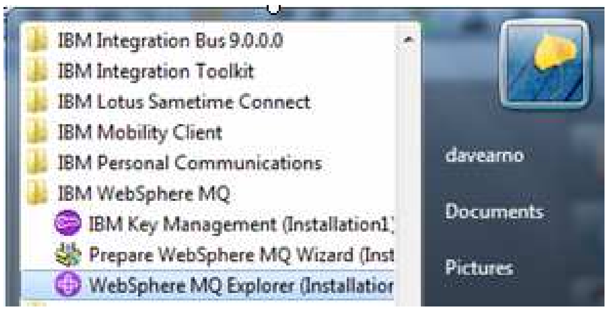
__50. Find the WebSphere MQ Alert Monitor in the Windows system tray.

__51. Press the right mouse button.

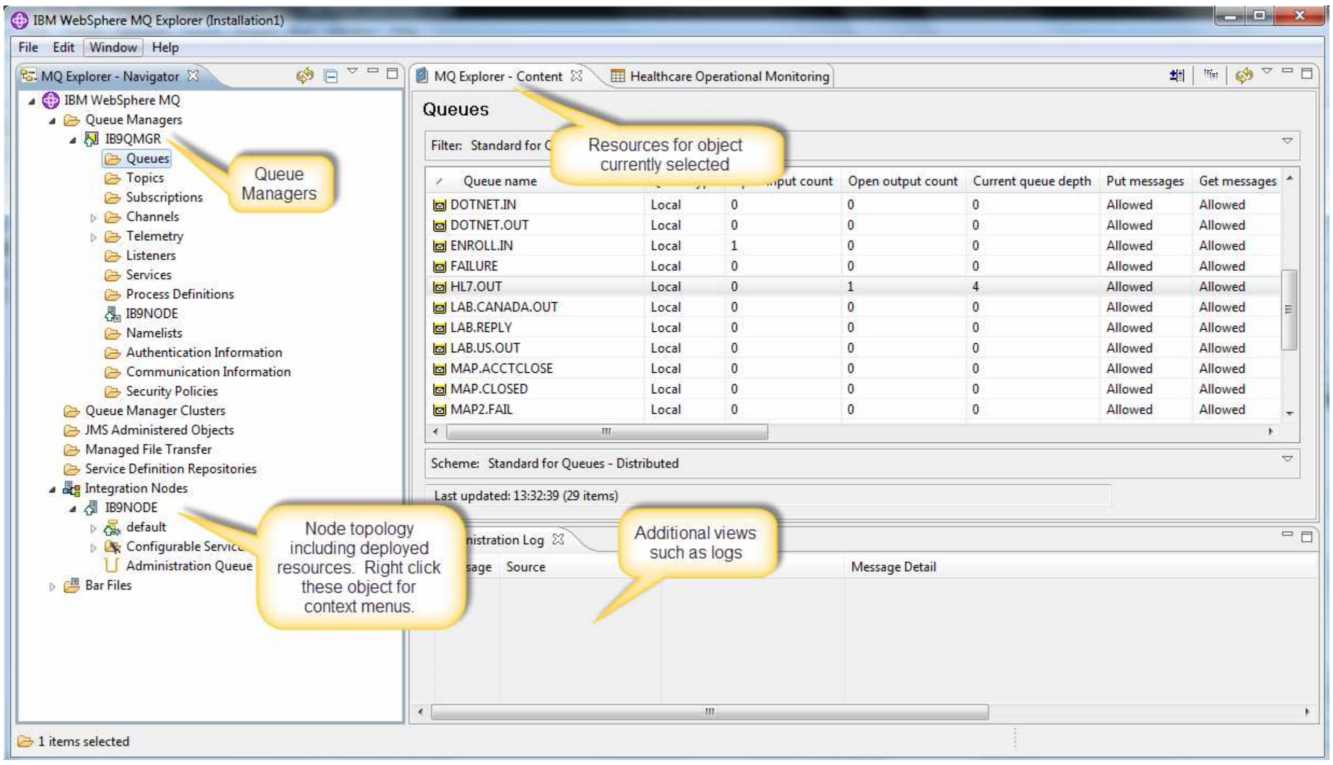
__52. Select **WebSphere MQ Explorer** from the menu.



OR use Start->All Programs


IBM Integration Bus 9.0.0.0
IBM Integration Toolkit
IBM Lotus Sametime Connect
IBM Mobility Client
IBM Personal Communications
IBM WebSphere MQ
IBM Key Management (Installation1)
Prepare WebSphere MQ Wizard (Inst
WebSphere MQ Explorer (Installatior
davearno
Documents
Pictures

Take a moment to familiarize yourself with the Explorer. This tool will be used again in later labs.



Queues

Filter: Standard for C

Queue name	[illegible]	put count	Open output count	Current queue depth	Put messages	Get messages
DOTNET.IN	Local	0	0	0	Allowed	Allowed
DOTNET.OUT	Local	0	0	0	Allowed	Allowed
ENROLL.IN	Local	1	0	0	Allowed	Allowed
FAILURE	Local	0	0	0	Allowed	Allowed
HL7.OUT	Local	0	1	4	Allowed	Allowed
LAB.CANADA.OUT	Local	0	0	0	Allowed	Allowed
LAB.REPLY	Local	0	0	0	Allowed	Allowed
LAB.US.OUT	Local	0	0	0	Allowed	Allowed
MAP.ACCTCLOSE	Local	0	0	0	Allowed	Allowed
MAP.CLOSED	Local	0	0	0	Allowed	Allowed
MAP2.FAIL	Local	0	0	0	Allowed	Allowed

Scheme: Standard for Queues - Distributed

Last updated: 13:32:39 (29 items)

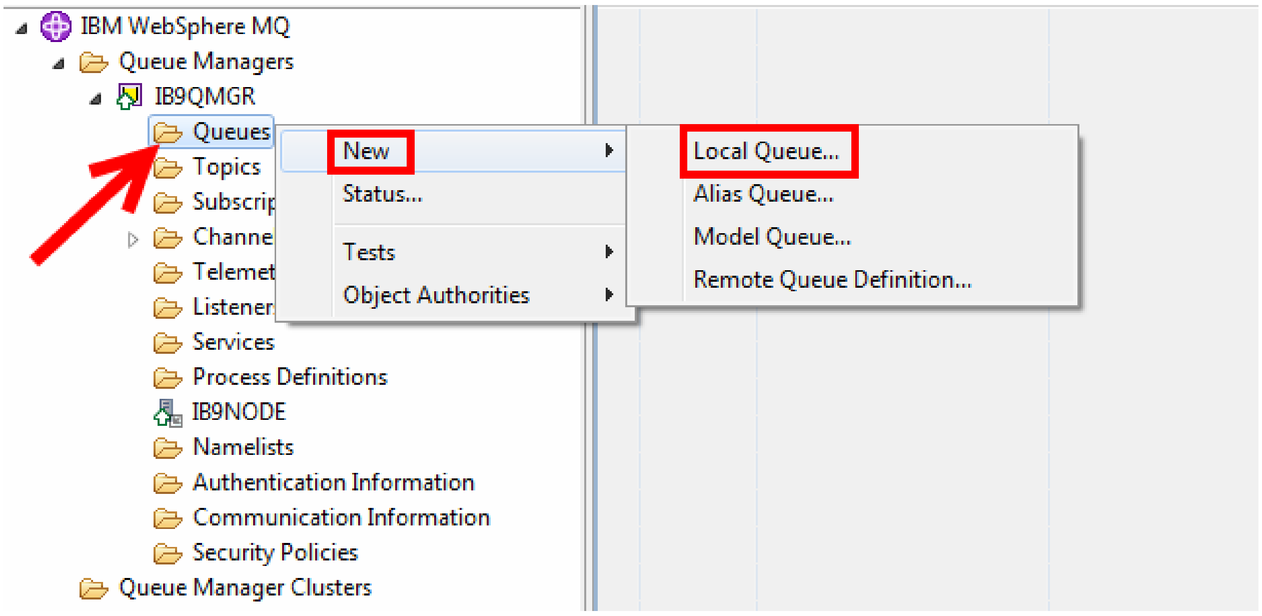
Key Concept: Integration Explorer (sometimes also called MQ Explorer)

The Integration Explorer is a lightweight eclipse tool for the administration of your Integration Bus and MQ topology. Technically, the Integration Explorer is an extension to the MQ Explorer that adds Integration Bus administrative capabilities to the tool. A few examples of these tasks include adding/modifying/deleting Brokers, access control, the deployment and modification of Apps and Libraries, viewing Administrative and Event logs, and administering the MQ Pub/Sub engine.

__53. Expand **IB9QMGR** (if necessary).

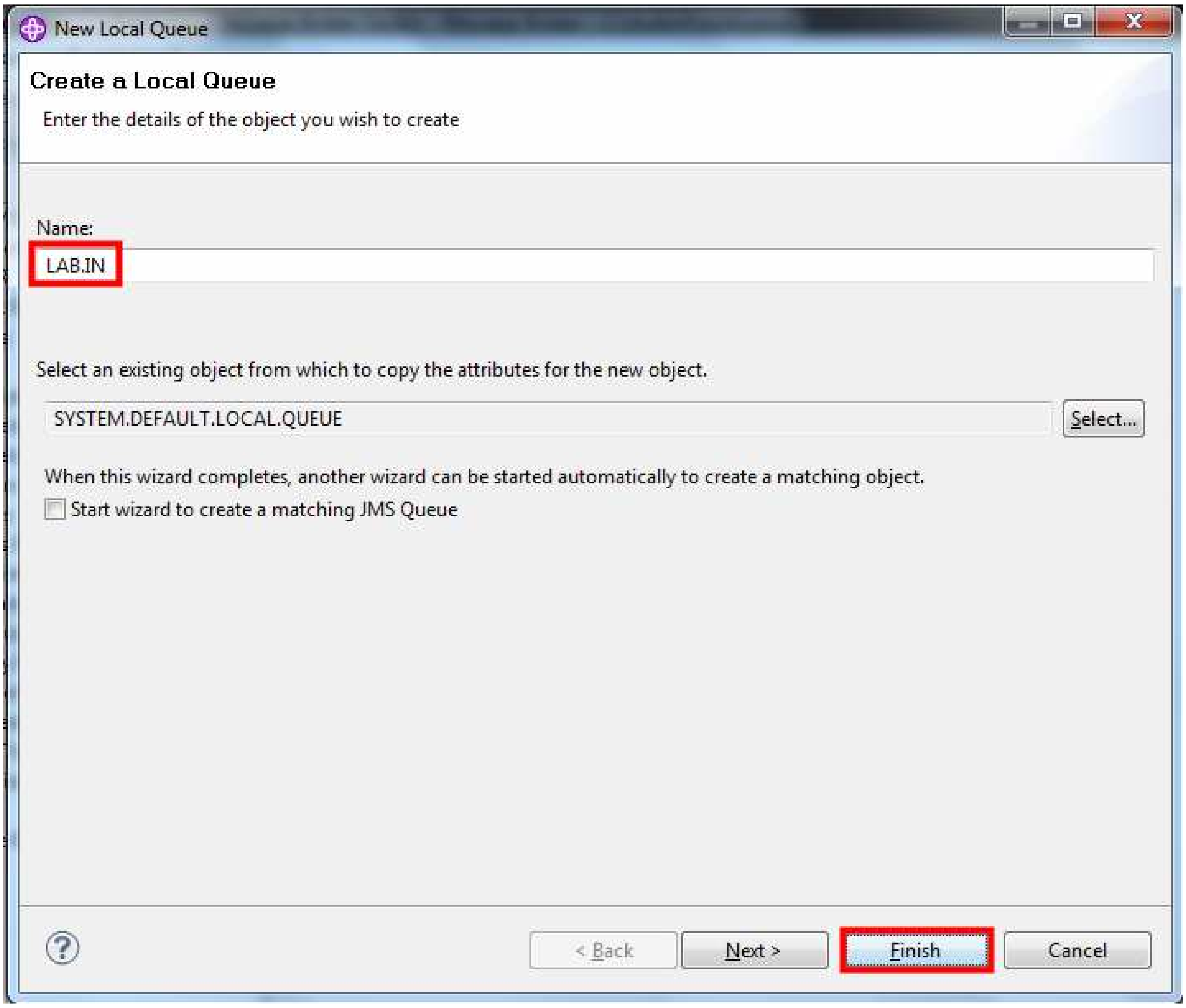
__54. Right click on the **IB9QMGR→Queues** folder.

__55. Select **New→Local Queue**.



__56. In the dialog, for the **Name** of the queue, enter **LAB.IN**.

__57. Click **Finish**.



New Local Queue

Create a Local Queue

Enter the details of the object you wish to create

Name: LAB.IN

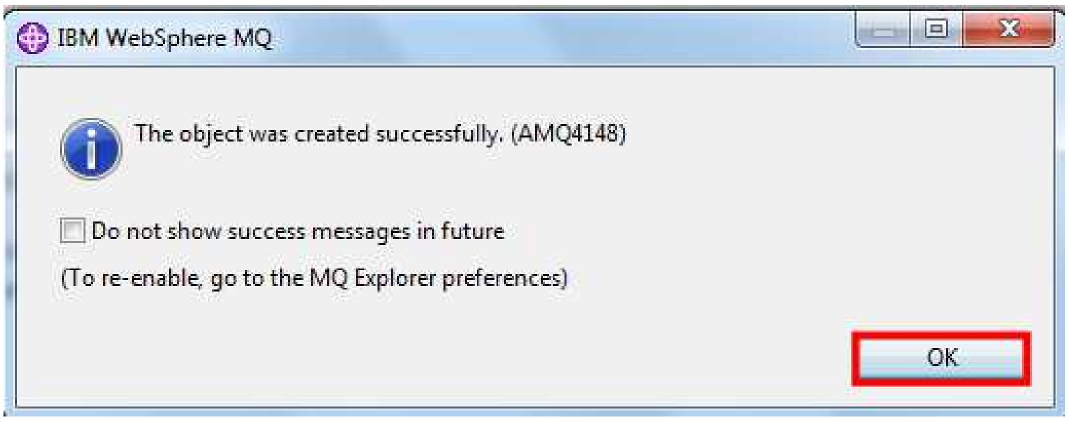
Select an existing object from which to copy the attributes for the new object.

SYSTEM.DEFAULT.LOCAL.QUEUE

When this wizard completes, another wizard can be started automatically to create a matching object.

☐ Start wizard to create a matching JMS Queue

__58. Click **OK** to close the dialog. The dialog confirms that the queue was created successfully.


IBM WebSphere MQ

The object was created successfully. (AMQ4148)

☐ Do not show success messages in future

(To re-enable, go to the MQ Explorer preferences)

__59. **IMPORTANT!** Repeat steps 55 through 59 to create a queue called **LAB.SEND.AS.XML**.

Now we are ready to retry the Test Client.

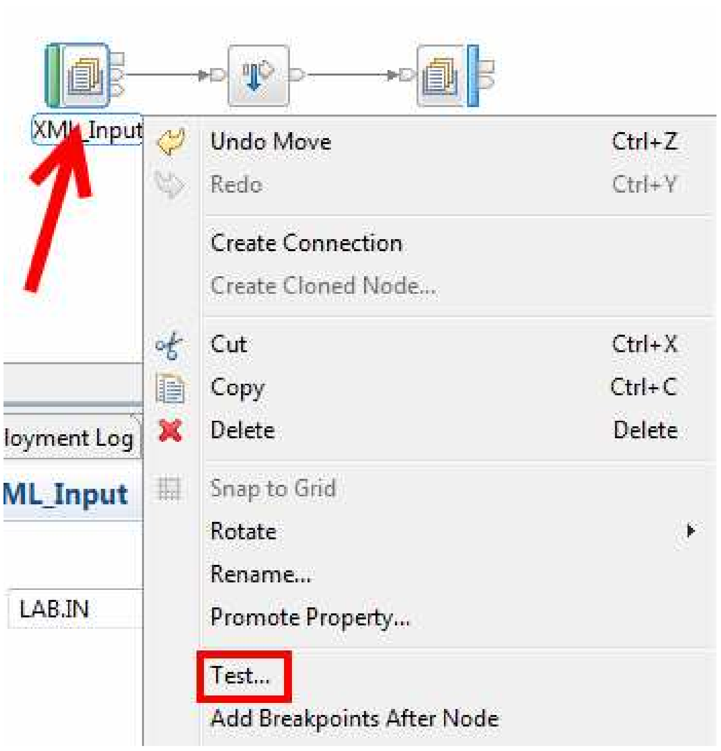
1.3 Testing the Flow

The message flow is now complete. The next step is to send it to the runtime environment for testing. The integrated Test Client will be used to test the message flow.

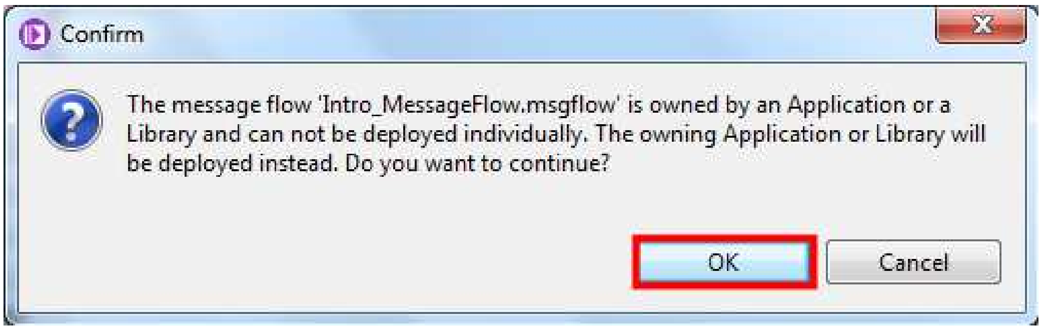
__1. Select the **MQInput** node (which we called **XML_Input**).

__2. Press the right mouse button.

__3. Select **Test…** from the menu.

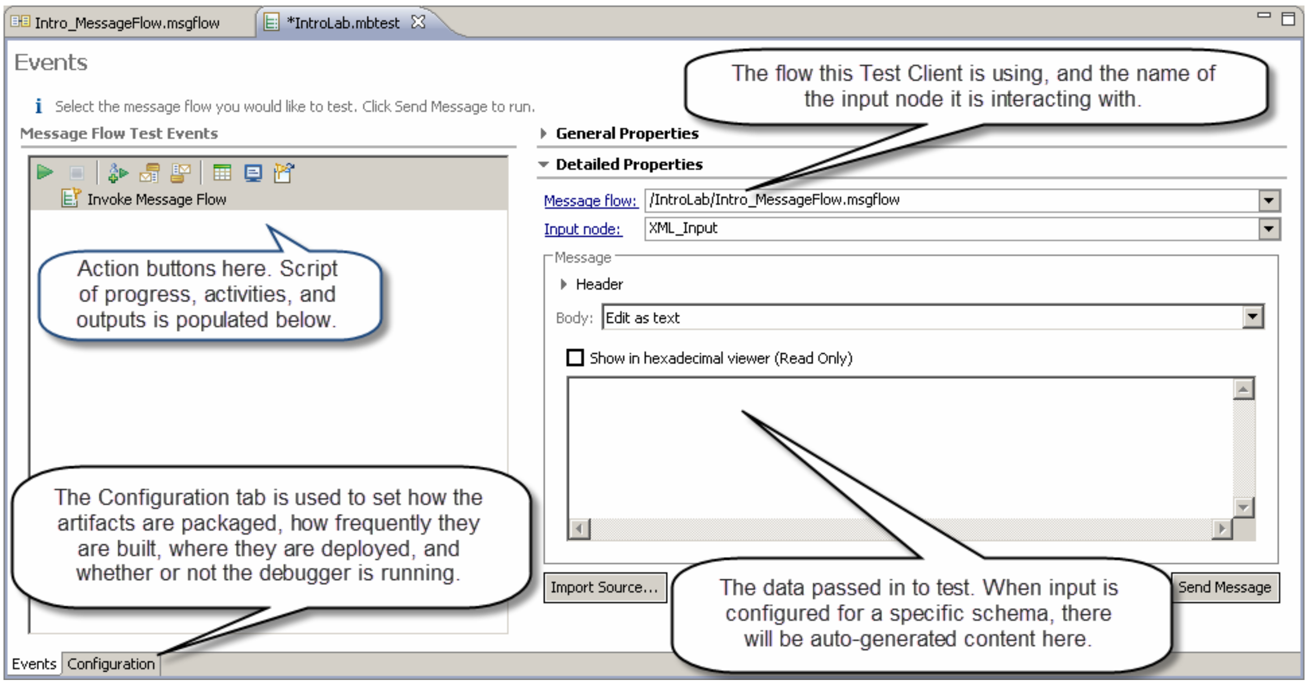


__4. Click **OK** to the popup.



The message flow 'Intro_MessageFlow.msgflow' is owned by an Application or a Library and can not be deployed individually. The owning Application or Library will be deployed instead. Do you want to continue?

Take a moment to familiarize yourself with the Test Client:


Intro_MessageFlow.msgflow

*IntroLab.mbtest

Events

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

Message Flow Test Events

Invoke Message Flow

Action buttons here. Script of progress, activities, and outputs is populated below.

The Configuration tab is used to set how the artifacts are packaged, how frequently they are built, where they are deployed, and whether or not the debugger is running.

Events | Configuration

The flow this Test Client is using, and the name of the input node it is interacting with.

General Properties

Detailed Properties

Message flow: /IntroLab/Intro_MessageFlow.msgflow

Input node: XML_Input

Message

Header

Body: Edit as text

Show in hexadecimal viewer (Read Only)

The data passed in to test. When input is configured for a specific schema, there will be auto-generated content here.

Import Source...

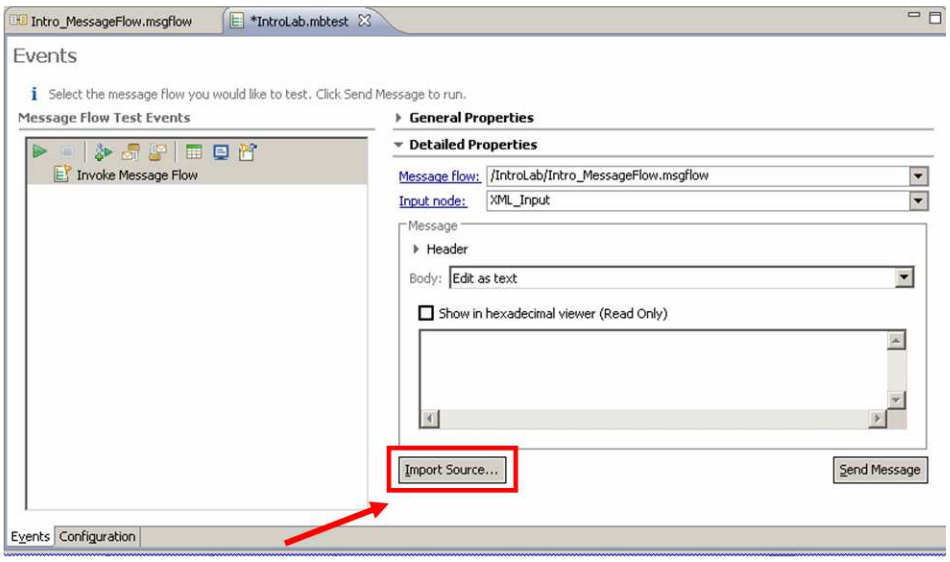
Send Message

Key Concept: Test Client

Test Client instances can be created for MQ, JMS, HTTP, SOAP and SCA input nodes. They exist as a single file in the workspace with a .mbtest file extension. They can be embedded into the Applications or Libraries and thus passed from developer to developer with a project.

The Test Client has many useful features such as monitoring all supported output paths through a flow and storing sample messages to a data pool. It also encapsulates the build and deploy process, and can be used to launch the debugger. Since it encapsulates a full test, including data, it can be used for regression testing.

__**5.** Select **Import Source…**



Message flow: /IntroLab/Intro_MessageFlow.msgflow

Input node: XML_Input

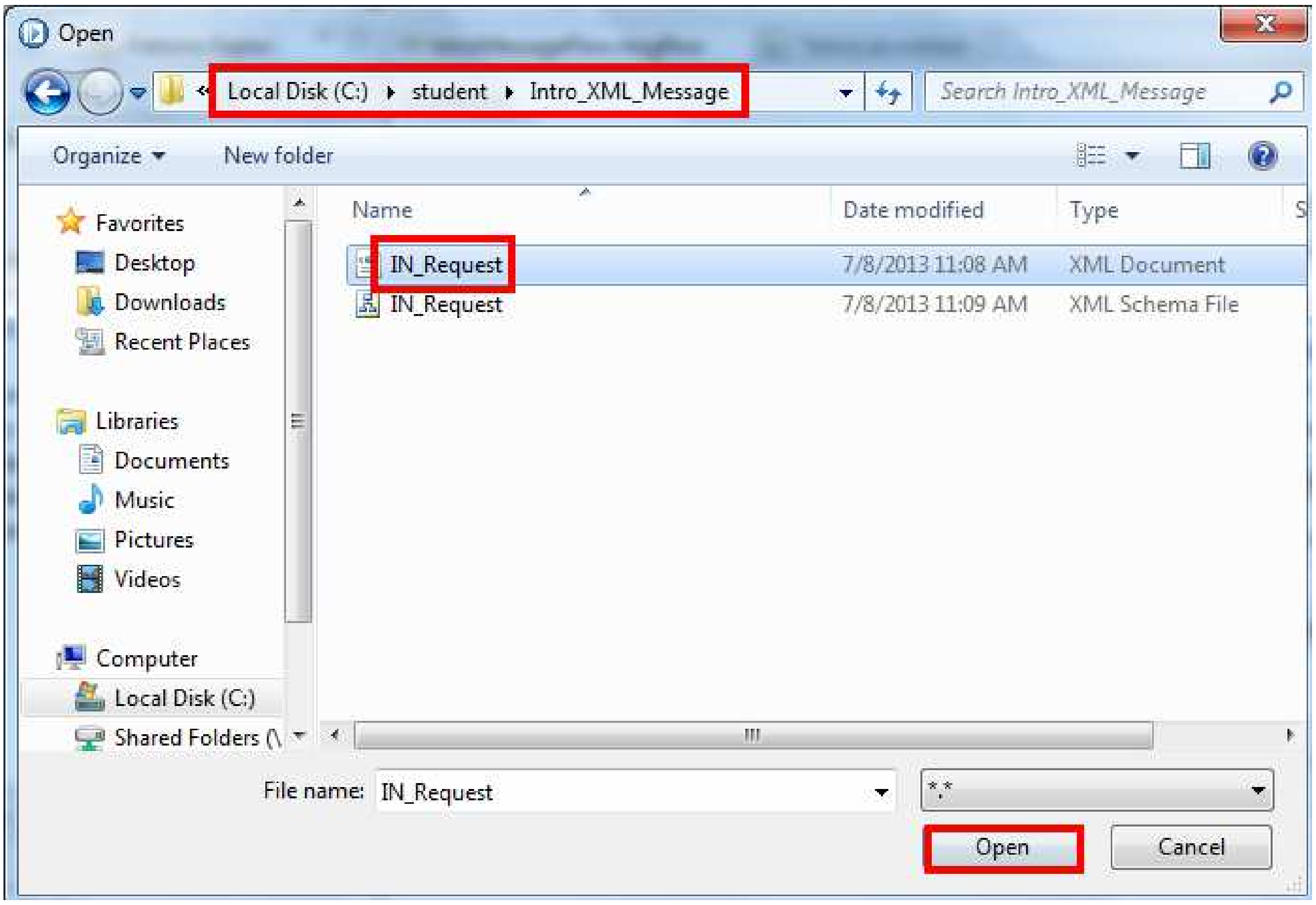
Body: Edit as text

☐ Show in hexadecimal viewer (Read Only)

__6. Navigate to the **C:\student\Intro_XML_Message** folder.

__7. Select **IN_Request** file. The **Type** should be **XML Document**.

__8. Click **Open**.



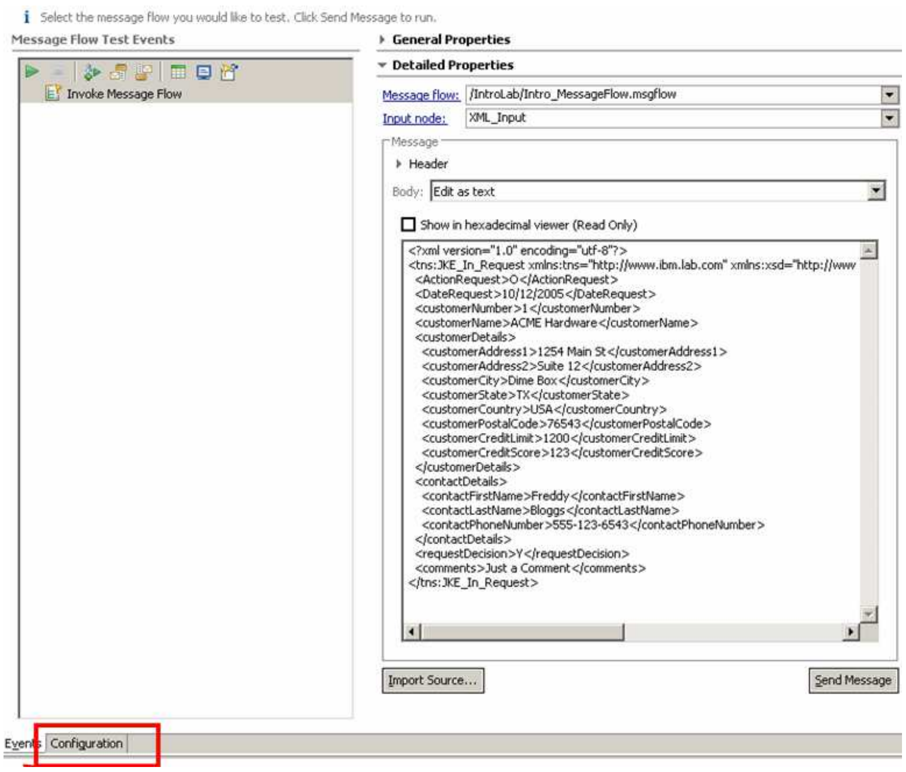
Name	Date modified	Type
IN_Request	7/8/2013 11:08 AM	XML Document
IN_Request	7/8/2013 11:09 AM	XML Schema File

File name: IN_Request

.

The XML sample message is now loaded into the Test Client.

__9. Select the **Configuration** tab.


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

Message Flow Test Events

Invoke Message Flow

General Properties

Detailed Properties

Message flow: /IntroLab/Intro_MessageFlow.msgflow

Input node: XML_Input

Message

Header

Body: Edit as text

Show in hexadecimal viewer (Read Only)

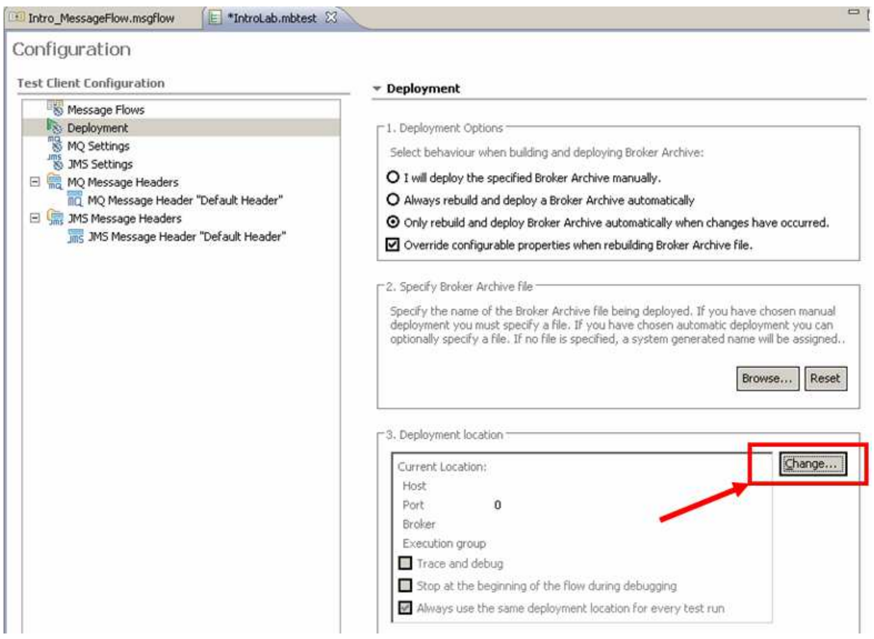
```
<?xml version="1.0" encoding="utf-8"?>
<tns:JKE_In_Request xmlns:tns="http://www.ibm.lab.com" xmlns:xsd="http://www
 <ActionRequest>O</ActionRequest>
 <DateRequest>10/12/2005</DateRequest>
 <customerNumber>1</customerNumber>
 <customerName>ACME Hardware</customerName>
 <customerDetails>
  <customerAddress1>1254 Main St</customerAddress1>
  <customerAddress2>Suite 12</customerAddress2>
  <customerCity>Dime Box</customerCity>
  <customerState>TX</customerState>
  <customerCountry>USA</customerCountry>
  <customerPostalCode>76543</customerPostalCode>
  <customerCreditLimit>1200</customerCreditLimit>
  <customerCreditScore>123</customerCreditScore>
 </customerDetails>
 <contactDetails>
  <contactFirstName>Freddy</contactFirstName>
  <contactLastName>Bloggs</contactLastName>
  <contactPhoneNumber>555-123-6543</contactPhoneNumber>
 </contactDetails>
 <requestDecision>Y</requestDecision>
 <comments>Just a Comment</comments>
</tns:JKE_In_Request>
```

Import Source... Send Message

Events Configuration

__10. Select **Deployment**.

__11. Click the **Change** button in **Section 3: Deployment Location**.


Intro_MessageFlow.msgflow *IntroLab.mbtest

Configuration

Test Client Configuration

- Message Flows
- Deployment
- MQ Settings
- JMS Settings
- MQ Message Headers
  - MQ Message Header "Default Header"
- JMS Message Headers
  - JMS Message Header "Default Header"

Deployment

1. Deployment Options

Select behaviour when building and deploying Broker Archive:

- ○ I will deploy the specified Broker Archive manually.
- ○ Always rebuild and deploy a Broker Archive automatically
- ● Only rebuild and deploy Broker Archive automatically when changes have occurred.
- ☑ Override configurable properties when rebuilding Broker Archive file.

2. Specify Broker Archive file

Specify the name of the Broker Archive file being deployed. If you have chosen manual deployment you must specify a file. If you have chosen automatic deployment you can optionally specify a file. If no file is specified, a system generated name will be assigned..

Browse... Reset

3. Deployment location

Current Location:

Host	
Port	0
Broker	
Execution group	

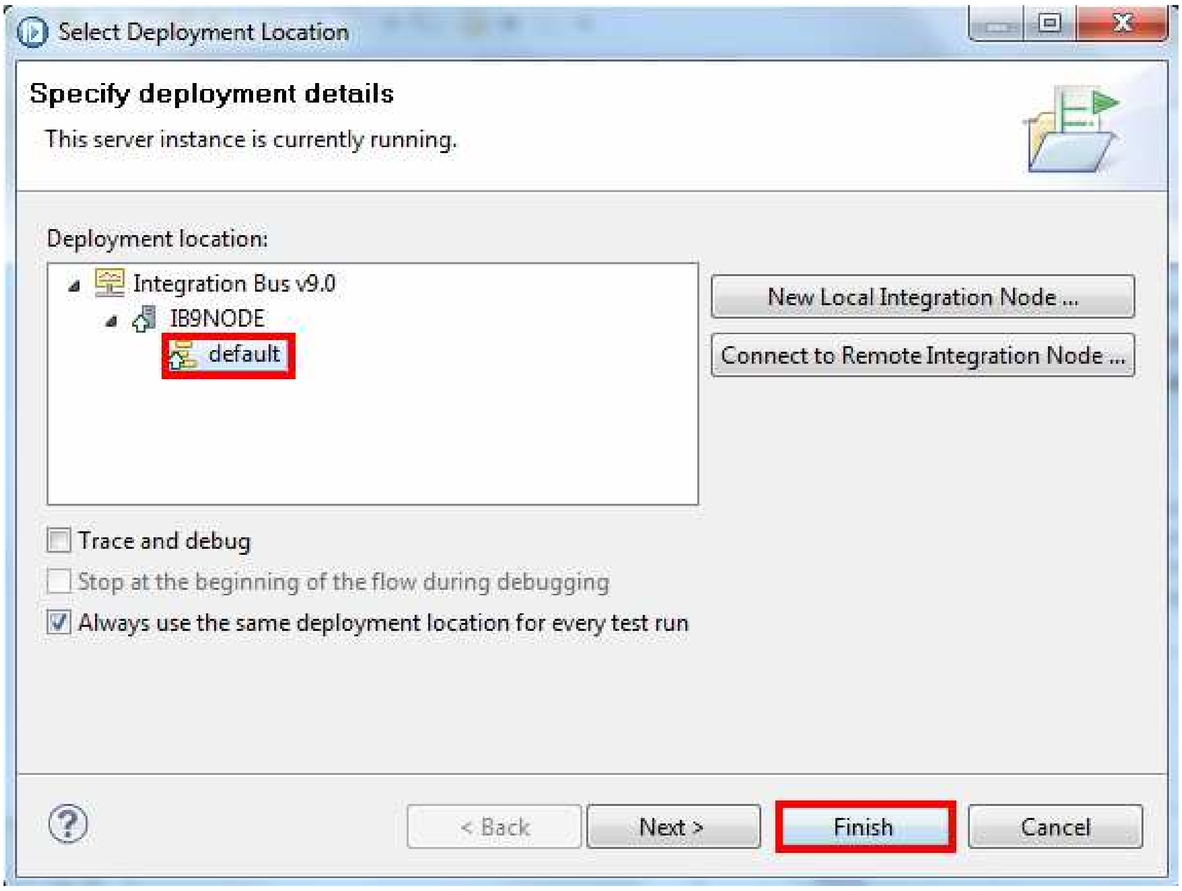
- ☐ Trace and debug
- ☐ Stop at the beginning of the flow during debugging
- ☑ Always use the same deployment location for every test run

Change...

Note that the local Integration Node (**IB9NODE**) and (**default)** Integration Server are displayed.

__12. Select the **default** integration server.

__13. Click **Finish**.



Select Deployment Location

Specify deployment details

This server instance is currently running.

Deployment location:

- Integration Bus v9.0
  - IB9NODE
    - default

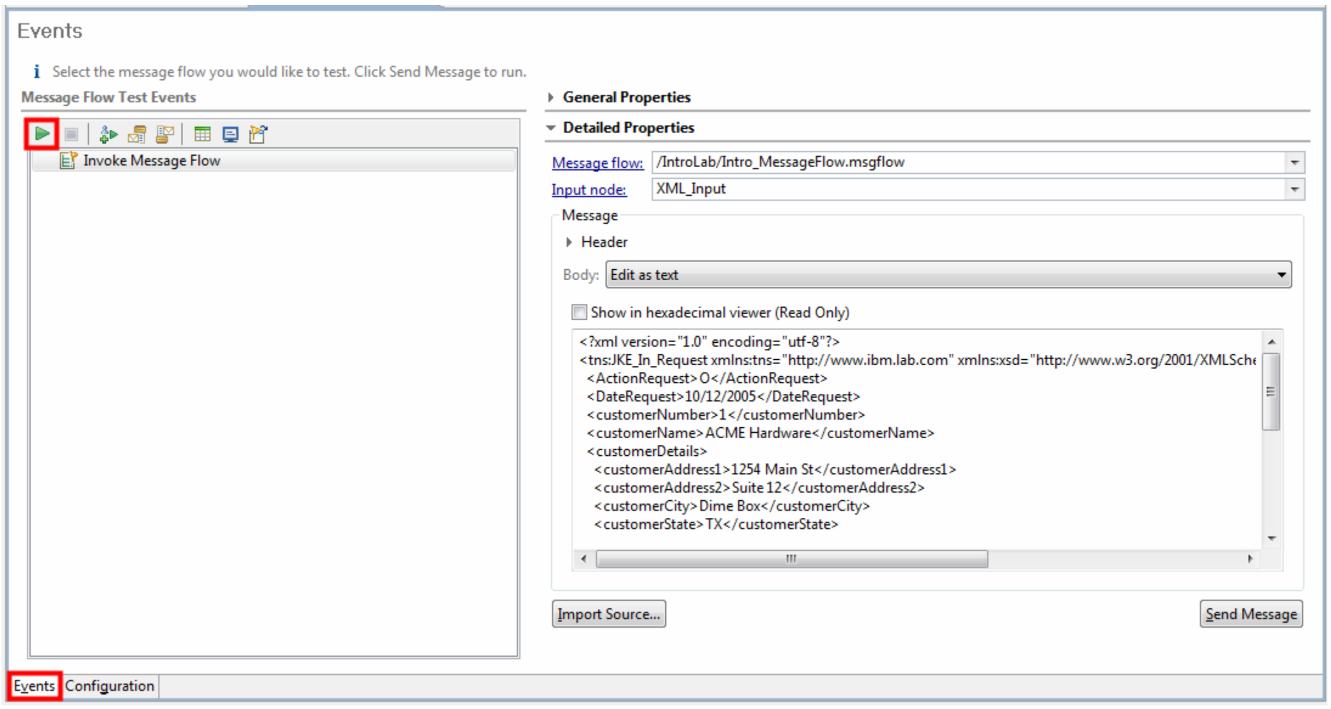
New Local Integration Node ...
Connect to Remote Integration Node ...

- [ ] Trace and debug
- [ ] Stop at the beginning of the flow during debugging
- [x] Always use the same deployment location for every test run

< Back
Next >
Finish
Cancel

__14. Switch back to the **Events** tab.

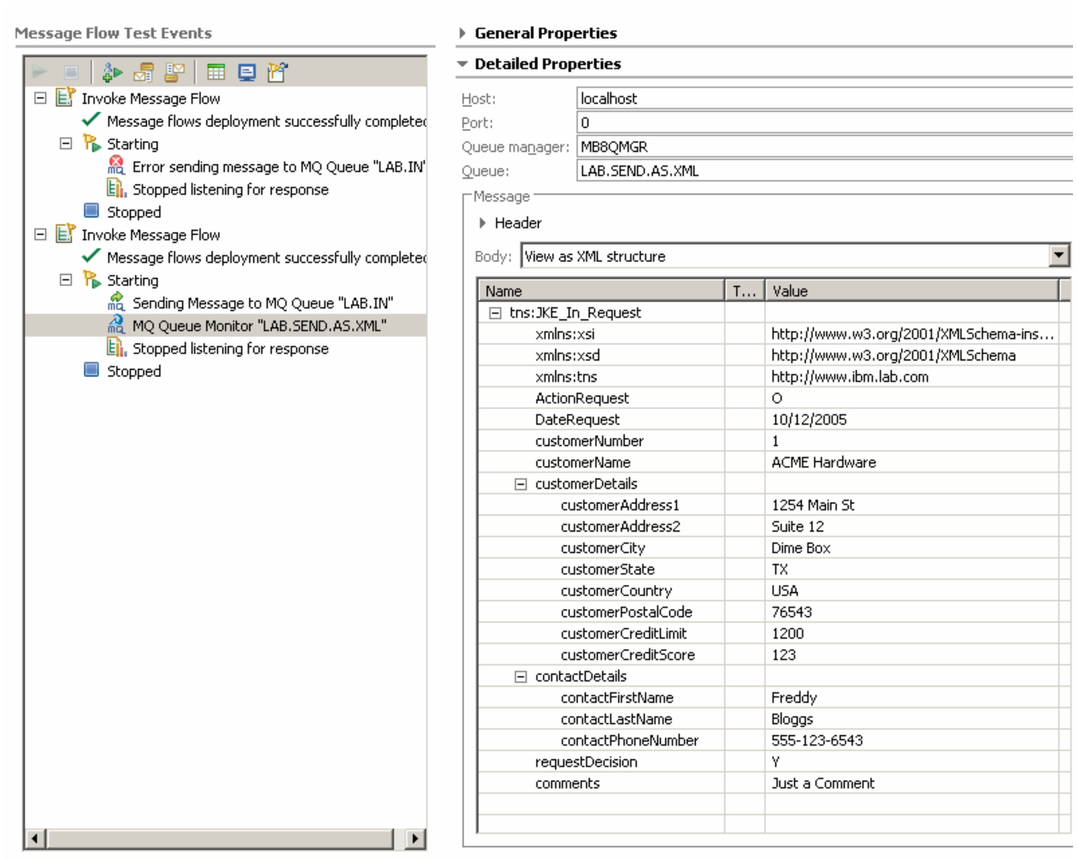
__15. Click the green **Play** button. (The **Send Message** button will also do the same thing.)



Message flow: /IntroLab/Intro_MessageFlow.msgflow
Input node: XML_Input
Message
Body: Edit as text

```
<?xml version="1.0" encoding="utf-8"?>
<tns:JKE_In_Request xmlns:tns="http://www.ibm.lab.com" xmlns:xsd="http://www.w3.org/2001/XMLSche
 <ActionRequest>0</ActionRequest>
 <DateRequest>10/12/2005</DateRequest>
 <customerNumber>1</customerNumber>
 <customerName>ACME Hardware</customerName>
 <customerDetails>
  <customerAddress1>1254 Main St</customerAddress1>
  <customerAddress2>Suite 12</customerAddress2>
  <customerCity>Dime Box</customerCity>
  <customerState>TX</customerState>
```

Recall that the output from our flow was a queue. By selecting the event starting with **MQ Queue Monitor…,** we see the output message. In other words, the Test Client did a “GET” operation off of the output queue and displays the message.


Message Flow Test Events

- Invoke Message Flow
  - Message flows deployment successfully completed
  - Starting
    - Error sending message to MQ Queue "LAB.IN"
    - Stopped listening for response
  - Stopped
- Invoke Message Flow
  - Message flows deployment successfully completed
  - Starting
    - Sending Message to MQ Queue "LAB.IN"
    - MQ Queue Monitor "LAB.SEND.AS.XML"
    - Stopped listening for response
  - Stopped

General Properties

Detailed Properties

Host:	localhost
Port:	0
Queue manager:	MB8QMGR
Queue:	LAB.SEND.AS.XML

Message

Header

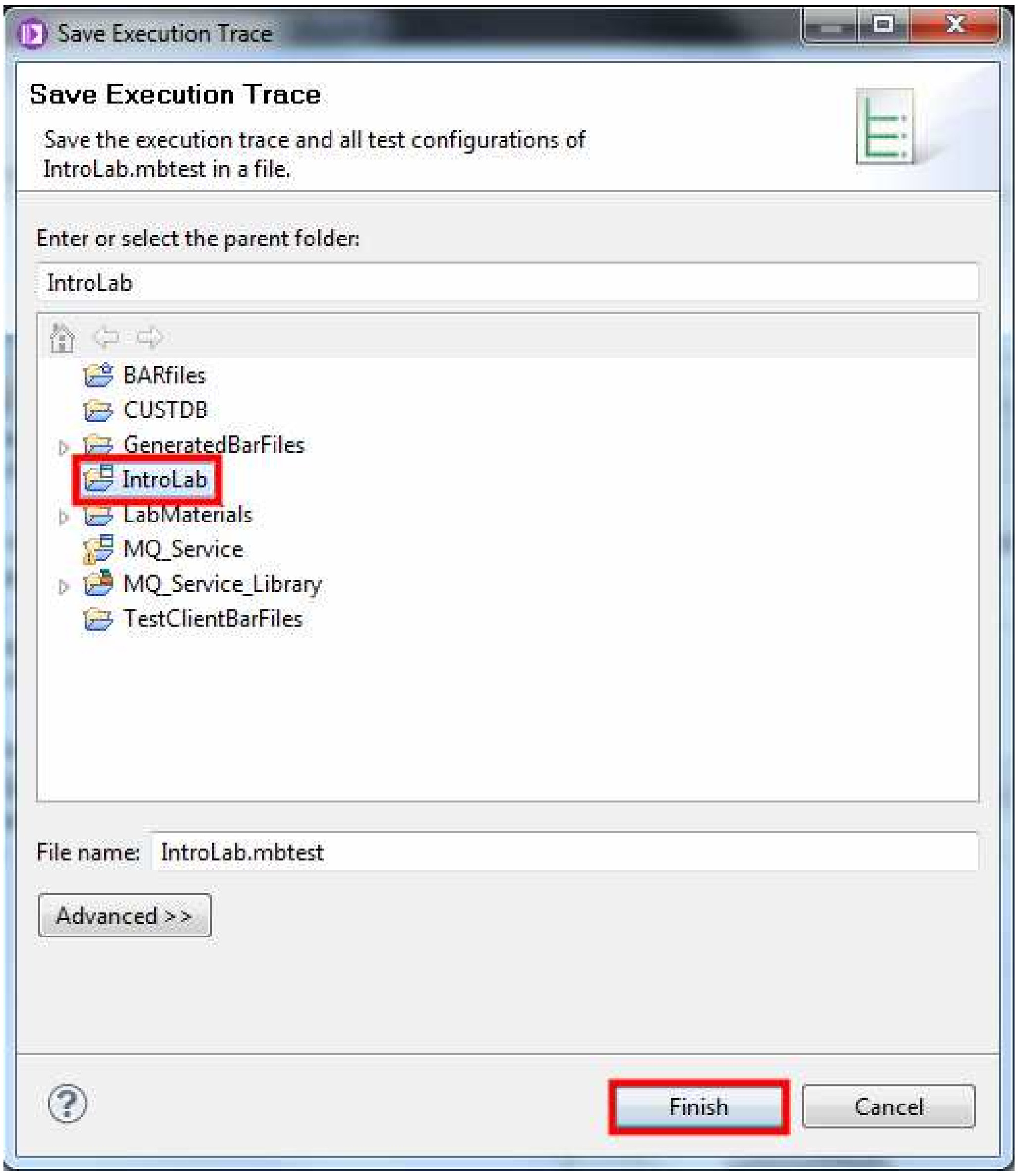
Body: View as XML structure

Name	T...	Value
tns:JKE_In_Request		
xmlns:xsi		http://www.w3.org/2001/XMLSchema-ins...
xmlns:xsd		http://www.w3.org/2001/XMLSchema
xmlns:tns		http://www.ibm.lab.com
ActionRequest		O
DateRequest		10/12/2005
customerNumber		1
customerName		ACME Hardware
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
requestDecision		Y
comments		Just a Comment

__16. Select **File→Save** to save the **IntroLab.mbtest** test client configuration.

__17. In the dialog that appears, select the **IntroLab** application.

__18. Click **Finish**.


Save Execution Trace

Save Execution Trace

Save the execution trace and all test configurations of IntroLab.mbtest in a file.

Enter or select the parent folder:

IntroLab

- BARfiles
- CUSTDB
- GeneratedBarFiles
- IntroLab
- LabMaterials
- MQ_Service
- MQ_Service_Library
- TestClientBarFiles

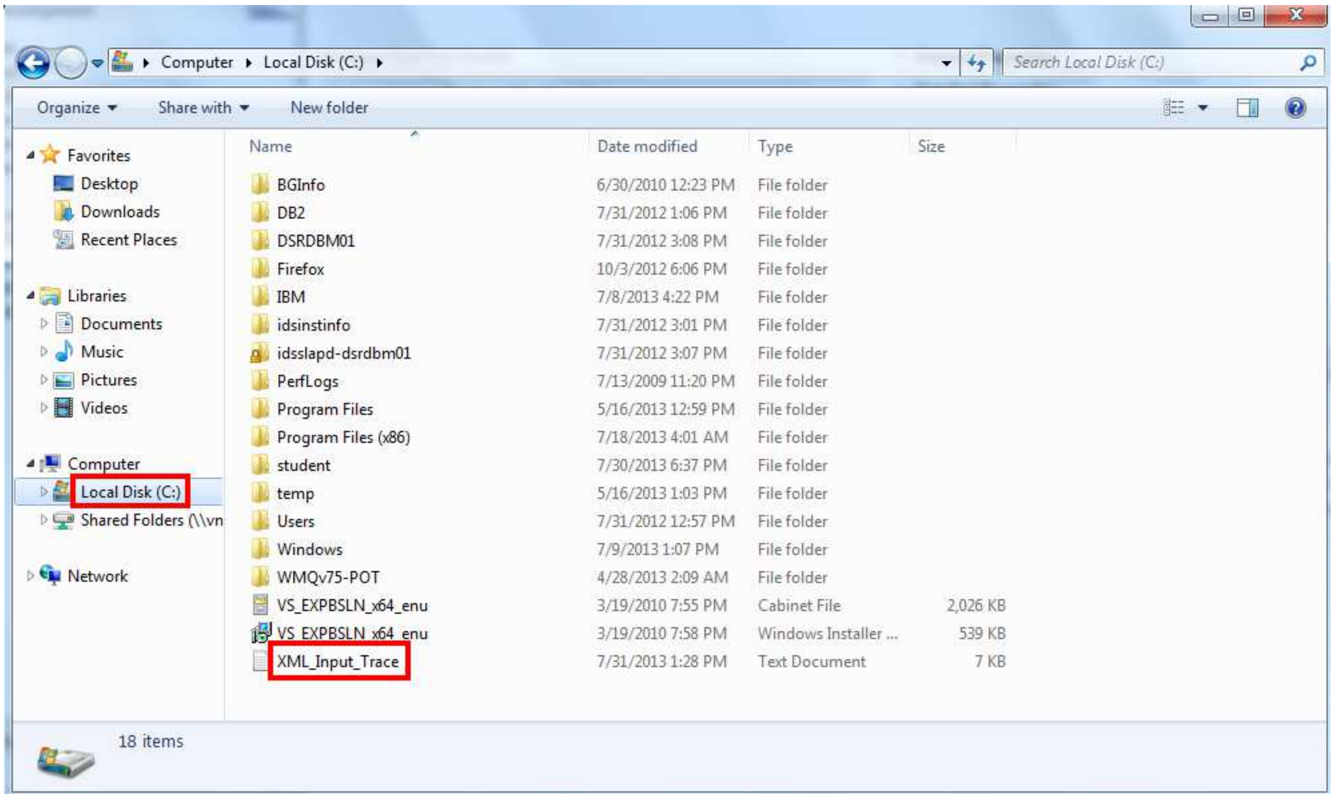
File name: IntroLab.mbtest

The output of the Trace Node in the file system will be examined next.

__19. Bring up Windows Explorer.

__20. Select the **Local Disk (C:)** folder (root directory).

__21. Double click on **XML_Input_Trace.txt**.



Computer ▸ Local Disk (C:) ▸

Search Local Disk (C:)

Organize ▾ Share with ▾ New folder

Favorites
- Desktop
- Downloads
- Recent Places

Libraries
- Documents
- Music
- Pictures
- Videos

Computer
- Local Disk (C:)
- Shared Folders (\\vn

Network

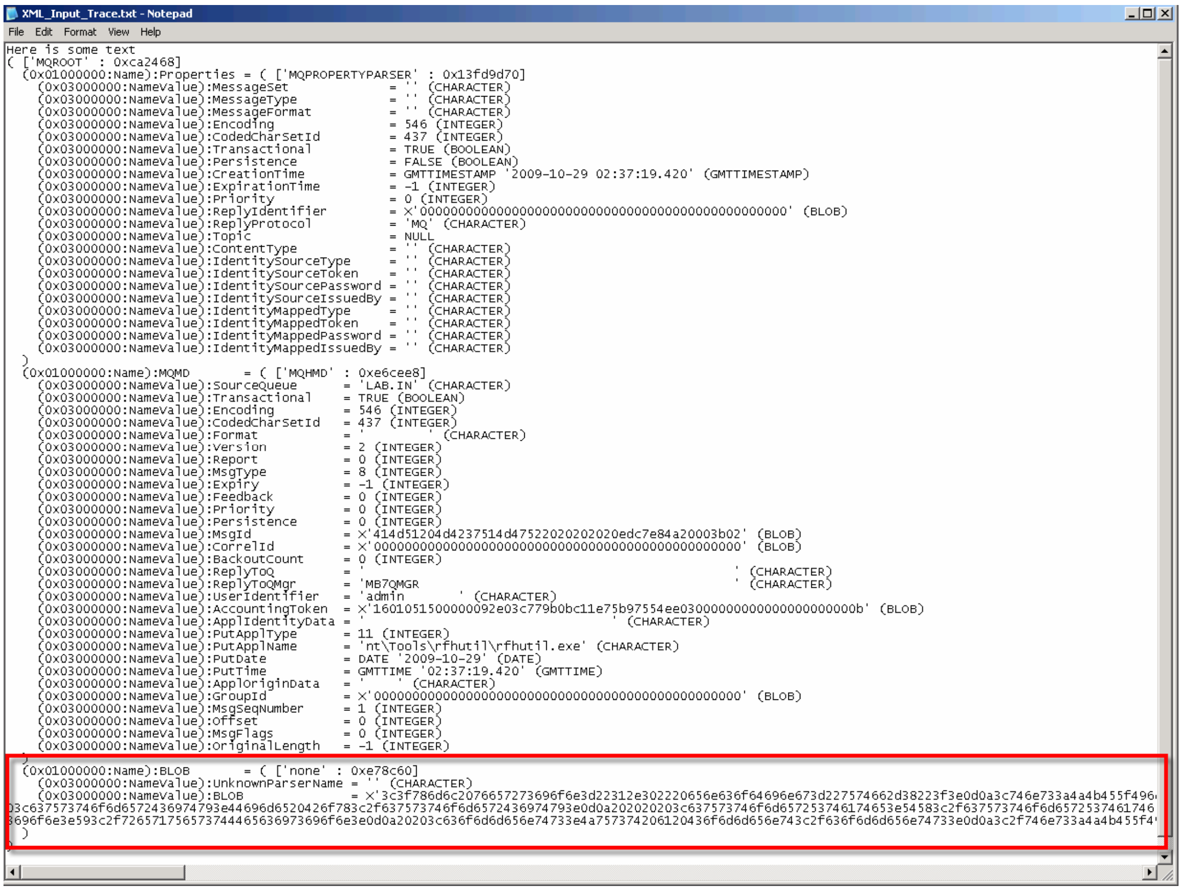
Name	Date modified	Type	Size
BGInfo	6/30/2010 12:23 PM	File folder	
DB2	7/31/2012 1:06 PM	File folder	
DSRDBM01	7/31/2012 3:08 PM	File folder	
Firefox	10/3/2012 6:06 PM	File folder	
IBM	7/8/2013 4:22 PM	File folder	
idsinstinfo	7/31/2012 3:01 PM	File folder	
idsslapd-dsrdbm01	7/31/2012 3:07 PM	File folder	
PerfLogs	7/13/2009 11:20 PM	File folder	
Program Files	5/16/2013 12:59 PM	File folder	
Program Files (x86)	7/18/2013 4:01 AM	File folder	
student	7/30/2013 6:37 PM	File folder	
temp	5/16/2013 1:03 PM	File folder	
Users	7/31/2012 12:57 PM	File folder	
Windows	7/9/2013 1:07 PM	File folder	
WMQv75-POT	4/28/2013 2:09 AM	File folder	
VS_EXPBSLN_x64_enu	3/19/2010 7:55 PM	Cabinet File	2,026 KB
VS_EXPBSLN_x64_enu	3/19/2010 7:58 PM	Windows Installer ...	539 KB
XML_Input_Trace	7/31/2013 1:28 PM	Text Document	7 KB

18 items

Viewing the contents of the trace file you can see the line of raw text you configured for the Trace node to display, as well as some detailed information about the message.

__22. To see the actual payload or application data scroll down to the bottom (**Ctrl+End**).

So what's going on here? The message is a BLOB – a Binary Large Object. Just a string of bytes shown in hexadecimal that happens to be XML.



```
Here is some text
( ['MQROOT' : 0xca2468]
  (0x01000000:Name):Properties = ( ['MQPROPERTYPARSER' : 0x13fd9d70]
    (0x03000000:NameValue):MessageSet             = '' (CHARACTER)
    (0x03000000:NameValue):MessageType            = '' (CHARACTER)
    (0x03000000:NameValue):MessageFormat          = '' (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 02:37:19.420' (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):Priority         = 0 (INTEGER)
    (0x03000000:NameValue):Persistence      = 0 (INTEGER)
    (0x03000000:NameValue):MsgId            = X'414d51204d4237514d47522020202020edc7e84a20003b02' (BLOB)
    (0x03000000:NameValue):CorrelId         = X'000000000000000000000000000000000000000000000000' (BLOB)
    (0x03000000:NameValue):BackoutCount     = 0 (INTEGER)
    (0x03000000:NameValue):ReplyToQ         = '                                                ' (CHARACTER)
    (0x03000000:NameValue):ReplyToQMgr      = 'MB7QMGR                                         ' (CHARACTER)
    (0x03000000:NameValue):UserIdentifier   = 'admin       ' (CHARACTER)
    (0x03000000:NameValue):AccountingToken  = X'1601051500000092e03c779b0bc11e75b97554ee030000000000000000000000000b' (BLOB)
    (0x03000000:NameValue):ApplIdentityData = '                                ' (CHARACTER)
    (0x03000000:NameValue):PutApplType      = 11 (INTEGER)
    (0x03000000:NameValue):PutApplName      = 'nt\Tools\rfhutil\rfhutil.exe' (CHARACTER)
    (0x03000000:NameValue):PutDate          = DATE '2009-10-29' (DATE)
    (0x03000000:NameValue):PutTime          = GMTTIME '02:37:19.420' (GMTTIME)
    (0x03000000:NameValue):ApplOriginData   = '    ' (CHARACTER)
    (0x03000000:NameValue):GroupId          = X'000000000000000000000000000000000000000000000000' (BLOB)
    (0x03000000:NameValue):MsgSeqNumber     = 1 (INTEGER)
    (0x03000000:NameValue):Offset           = 0 (INTEGER)
    (0x03000000:NameValue):MsgFlags         = 0 (INTEGER)
    (0x03000000:NameValue):OriginalLength   = -1 (INTEGER)
  )
  (0x01000000:Name):BLOB       = ( ['none' : 0xe78c60]
    (0x03000000:NameValue):UnknownParserName = '' (CHARACTER)
    (0x03000000:NameValue):BLOB              = X'3c3f786d6c2076657273696f6e3d22312e302220656e636f64696e673d227574662d38223f3e0d0a3c746e733a4a4b455f496
3c637573746f6d6572436974793e44696d6520426f783c2f637573746f6d6572436974793e0d0a202020203c637573746f6d65725374617465543c2f637573746f6d6572537461746
3696f6e3e593c2f726571756573744465636973696f6e3e0d0a20203c636f6d6d656e74733e4a757374206120436f6d6d656e743c2f636f6d6d656e74733e0d0a3c2f746e733a4a4b455f4
  )
)
```

In the next lab we'll associate an XML parser with the input message.

__23. Close the Notepad window.

__24. Minimize the Windows Explorer window.

Each time that you test the message flow new data will be appended to the end of the trace file. You will need to scroll down to the end of the file to see the latest information.

This is the end of Lab 1.