IBM Integration Bus V10.0

IIB Docker Container Deployment

Part A - Obtaining and Installing the IIB v10 Studio

Part B - Creating, Deploying and Testing Integration Services

V1.0 February 2015

Table of Contents

Overview

Description

The steps in this lab will take you through connecting the IIB Studio to the IIB runtime docker container.

You will complete the following steps. If you already have the IIB Studio downloaded you can jump to Part 2.

1. Part 1 – Obtain, install and connect the IIB Studio
    a. Download the IIB open beta development studio
    b. Install and start the IIB studio
    c. Connect to the IIB runtime in the Docker container
2. Part 2 – Create, Deploy and Test your first IIB Integration Service
    a. Web Service enablement of a backend system
    b. Exercise IIB SOAP, Graphical Mapping and File I/O capabilities

Pre-requisites

None

Part A – Download, install and connect the IIB v10 Open Beta Studio

Download the IIB open beta development studio

Download the IBM Integration Bus v10 Open Beta to your local computer so that you can explore the very latest IIB driver, run the samples, and develop your own solutions and skills.

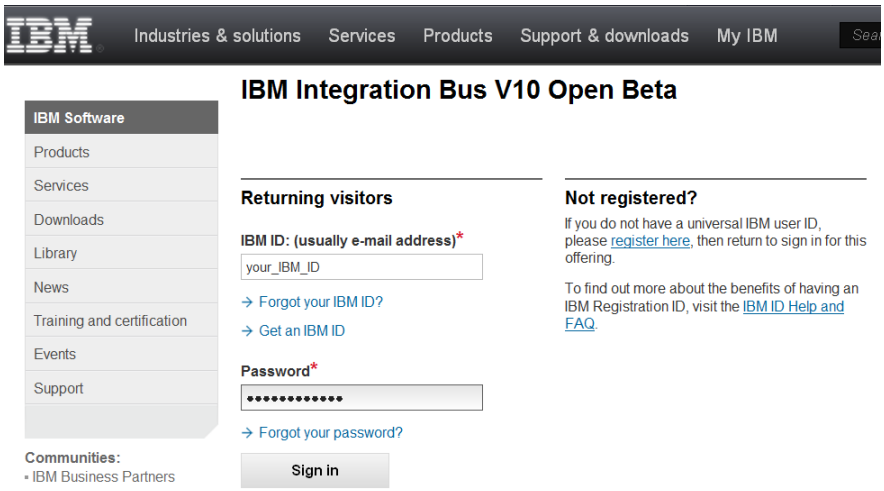
The IBM Integration Bus V10 Open Beta driver has been published for Windows 64-bit and Linux/Intel 64-bit.

To access the download site, you must log in with your IBM ID. If you don't have an IBM ID, obtain one for free here:

https://www.ibm.com/account/profile/us?page=reg&okURL=https%3A%2F%2Fwww14.software.ibm.com%2Fwebapp%2Fiwm%2Fweb%2Freg%2Fpick.do%3Fsource%3Dswg-beta-IIBVNOB%26lang%3Den_US&required=fname+lname+dphn+oaddr+

Download IIB V10 Open Beta from here:

https://ibm.biz/iibopenbeta



Read and agree to the license conditions and hit the button “I confirm”

Select from a Windows or Linux install and download the product

IBM Integration Bus V.Next for Windows, 64-bit (2 February 2015) IIBSetup10.0.640.0.exe (1.2 GB)	Download now
IBM Integration Bus V.Next for Linux/Intel, 64-bit (2 February 2015) iib-10.0.640.0.tar.gz (1.2 GB)	Download now

Save to your local system

Whilst the download is completing review Linux or Windows installation videos below, or follow the instructions in this document to install the product

Install and start the IIB studio

Watch the installation videos or follow (cut n paste) the instructions for Linux below.

Watch the Linux install video (0 – 3 mins):

https://www.youtube.com/watch?v=Num615zf64s

Watch the Windows install video (0 – 2 mins):

https://ibm.biz/iibopenbetavideo

Linux installation instructions

I saved the iib-10-*betabuildnumber*.tar.gz file to my Downloads directory

```
davearno@ubuntu:~$ ls
Desktop    Downloads         Music     Public     Videos
Documents  examples.desktop  Pictures  Templates
davearno@ubuntu:~$ cd Downloads
davearno@ubuntu:~/Downloads$ ls
iib-10.0.646.0.tar.gz
```

Change to the Downloads directory and run the untar command to install the product (your beta build number may be different)

```
$ tar -zxvf iib-10.0.646.0.tar.gz
```

```
./iib-10.0.646.0/tools/icon.xpm
./iib-10.0.646.0/tools/libcairo-swt.so
./iib-10.0.646.0/tools/mqsicreatebar
./iib-10.0.646.0/tools/mqsicreatemsgdefs
./iib-10.0.646.0/tools/mqsicreatemsgdefsfromwsdl
./iib-10.0.646.0/tools/notice.html
./iib-10.0.646.0/iib
davearno@ubuntu:~/Downloads$
```

The untar (install) takes about a minute. When is has finished change to the iib-10.*buildnumber* directory and you'll find the "iib" command

```
davearno@ubuntu:~/Downloads$ ls
iib-10.0.646.0
davearno@ubuntu:~/Downloads$ cd iib-10.0.646.0
davearno@ubuntu:~/Downloads/iib-10.0.646.0$ ls
common  extensions  iib  license  properties  readmes  server  tools
davearno@ubuntu:~/Downloads/iib-10.0.646.0$ ./iib
```

Run the iib command with the accept license silently option

```
$ ./iib accept license silently
```

```
NOTE: If you running as root use this alternative command
```

```
./iib make registry global accept license silently./iib make registry
global accept license silently
```

```
davearno@ubuntu:~/Downloads/iib-10.0.646.0$ ./iib accept license silently
License accepted and local registry created in /home/davearno/iibconfig, to begi
n type:    iib toolkit
```

We can now start the IIB Studio (or Toolkit, the Development/Administration GUI)

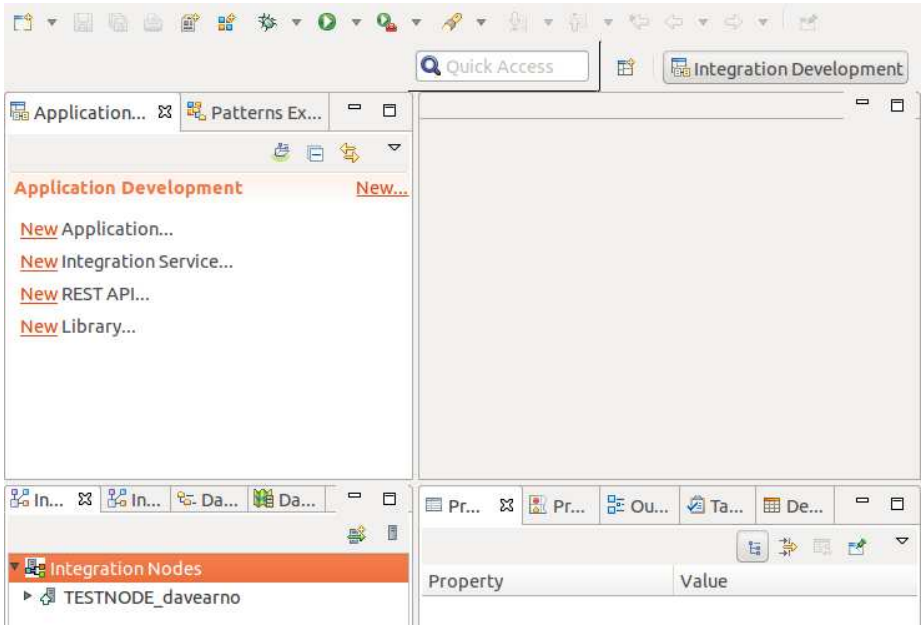
```
$ ./iib toolkit
```

```
davearno@ubuntu:~/Downloads/iib-10.0.646.0$ ./iib toolkit
Starting IIB Toolkit ...
davearno@ubuntu:~/Downloads/iib-10.0.646.0$
```

Start the IIB v10 Studio and disable the built in test node

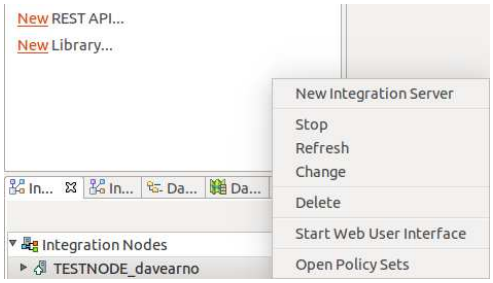
The IIB v10 Open Beta Studio includes a built in test runtime. We will not use this runtime in favour of connecting to the IIB runtime Docker container.

Close the Welcome Screen by clicking on the X in the top left of the screen and you will presented with the Integration Development perspective.



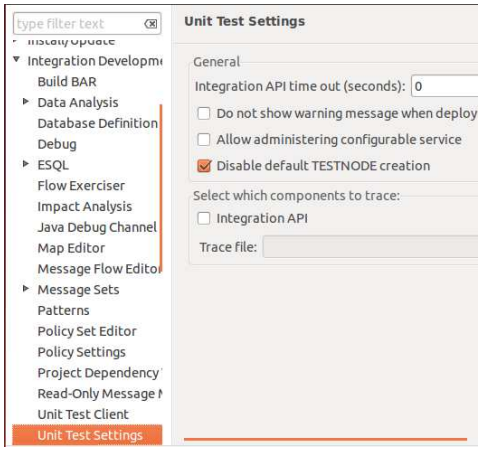
As we will not be using the built in TESTNODE_*username* in favour of our newly instantiated Docker container based IIB runtime, let's stop it.

Right click on TESTNODE_*username* and select "Stop"



Now disable TESTNODE_*username*

I. Select the menu item Window → Preferences
II. In the left hand column expand Integration Development
III. Select Unit Test Settings
IV. Tick the 'Disable default TESTNODE creation' box



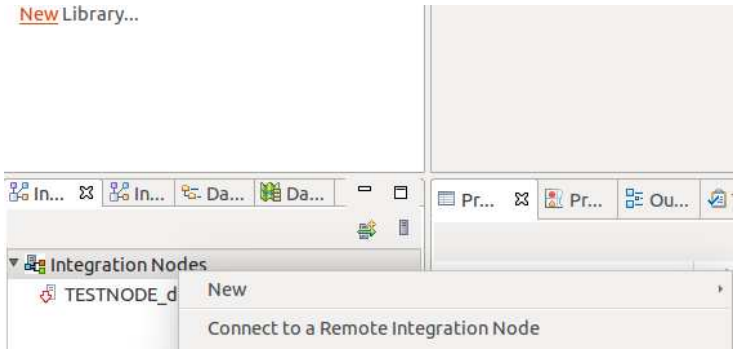
Now restart the Toolkit via File->Exit. Then ./iib toolkit from the command line.

Connect to the IIB runtime in the Docker container

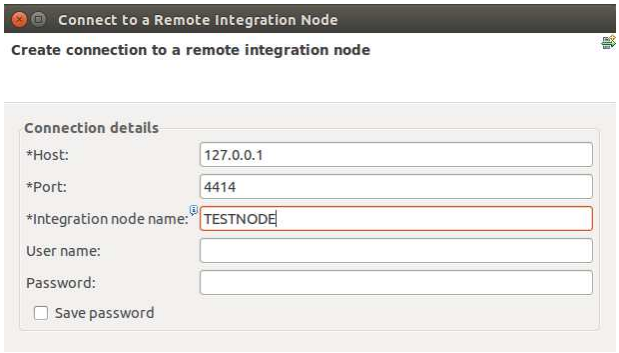
Next we will connect to the IIB TESTNODE running in the IIB Docker container.

If you have stopped the Docker container or the IIB node in the container use the commands on page 9 “Useful commands” to start them up.

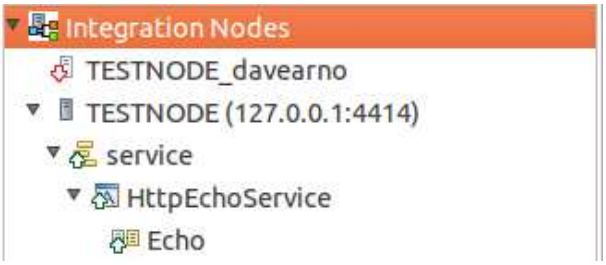
Right click on Integration Nodes and select “Connect to a Remote Integration Node”



Fill in the details for the target, Docker container based IIB Integration Node and then hit “Finish”.



You should see the TESTNODE->Integration Server and the “Echo” test service running.


Integration Nodes
TESTNODE_davearno
TESTNODE (127.0.0.1:4414)
service
HttpEchoService
Echo

Useful Commands for Starting and Stopping the IIB Container

For your reference here are useful commands for starting and stopping the container.

Stopping the running IIB node and the container

$docker exec -id dev_esb01 "/bin/bash -lc 'iib stop TESTNODE' "

$docker stop dev_esb01

```
davearno@ubuntu:~/Downloads/iib-10.0.646.0$ docker exec -id  dev_esb01  "/bin/ba
sh -lc 'iib stop TESTNODE' "
davearno@ubuntu:~/Downloads/iib-10.0.646.0$ docker stop dev_esb01
dev_esb01
davearno@ubuntu:~/Downloads/iib-10.0.646.0$
```

Re-starting the running IIB node in the container

```
$docker start dev_esb01
```

```
davearno@ubuntu:~/Downloads/iib-10.0.646.0$ docker start dev_esb01
dev_esb01
```

```
$docker exec -id dev_esb01 /bin/bash -lc 'iib start TESTNODE'
```

```
davearno@ubuntu:~/Downloads/iib-10.0.646.0$ sudo docker exec -id dev_esb01 /bin
bash -lc 'iib start TESTNODE'
[sudo] password for davearno:
davearno@ubuntu:~/Downloads/iib-10.0.646.0$
```

Part B – Creating, Deploying and Test your 1st Integration Service

Overview

In the lab you will create your 1st IIB Integration Service.

You will use a sample WSDL file to expose an IIB Integration Service as a web service.

This simple service will log customer address details to file and return a customer address object enriched with city and country data.

Pre-requisites

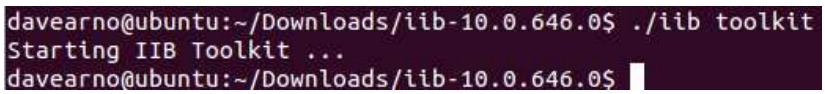
The DemoCustomerBasic.wsdl file from the Getting Started with IIB web site

@DA@ link here

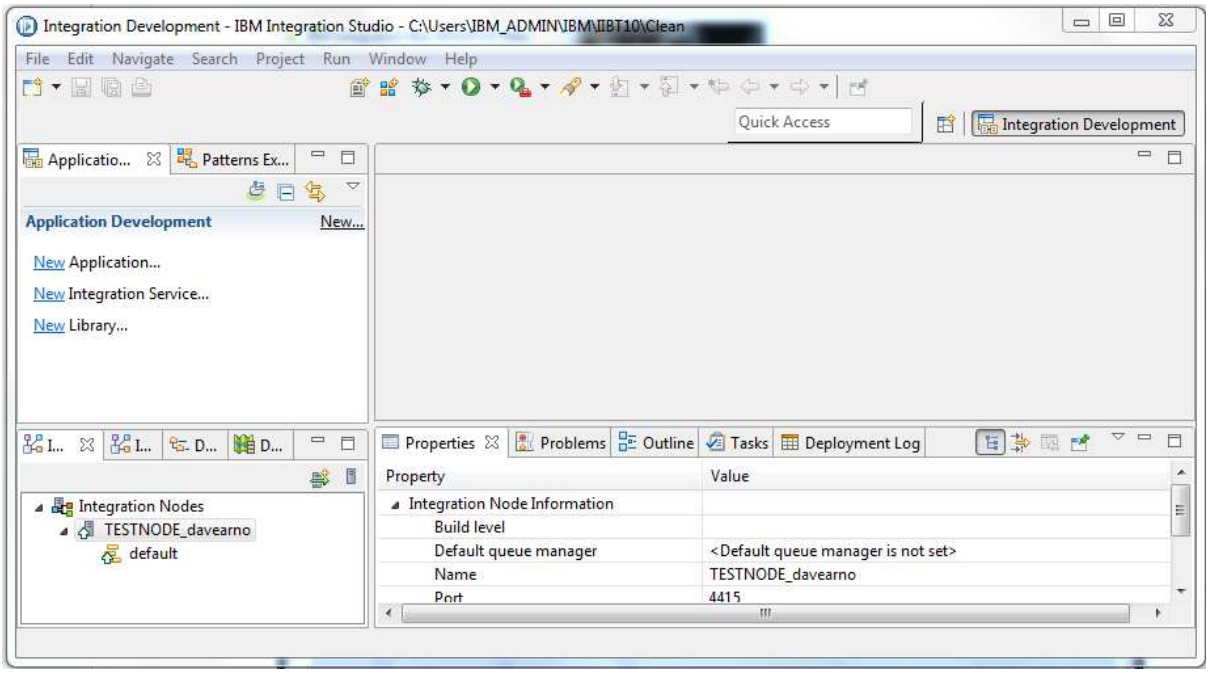
Constructing the IIB Integration Service in the IIB Studio

1. If not already open, open the IBM Integration Studio/Toolkit.

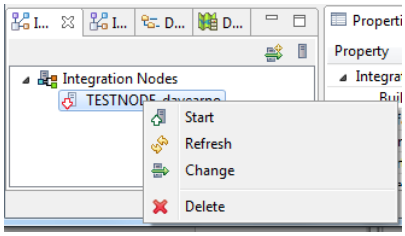
```
$ ./iib toolkit
```



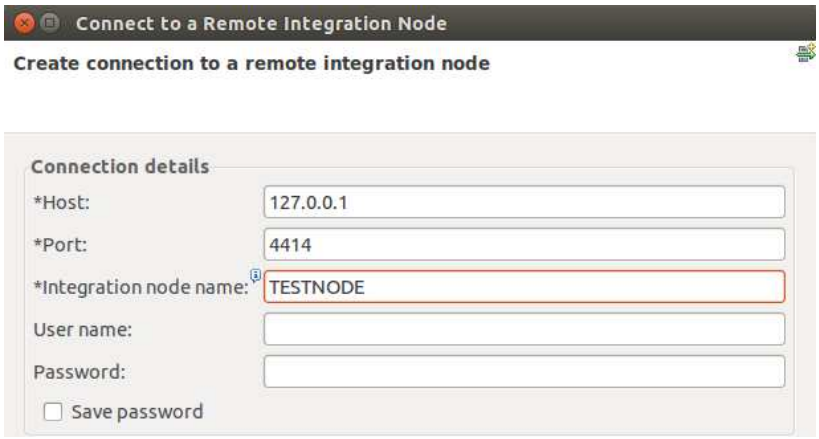
Close the Welcome screen (if necessary) and you will be presented with the development and testing workspace



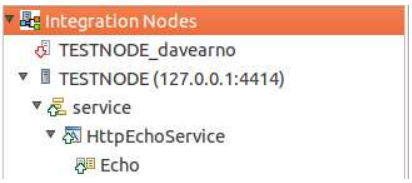
Ensure the built in TESTNODE_*username* is stopped and connect to the IIB Docker container based TESTNODE.



Right click on Integration Nodes and select “Connect to a remote integration node”

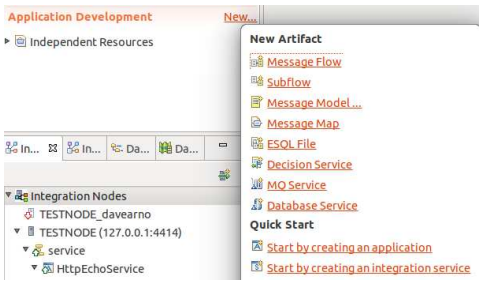


You should see the “*Echo*” test service running in the “*HttpEchoService”* Application under the “*service”* integration server on the TESTNODE



2. Start by using the New Integration Service wizard to import your WSDL file and generate the integration service framework (skeleton artefacts).

Click on New Integration Service…. Or New and Start by creating an integration service

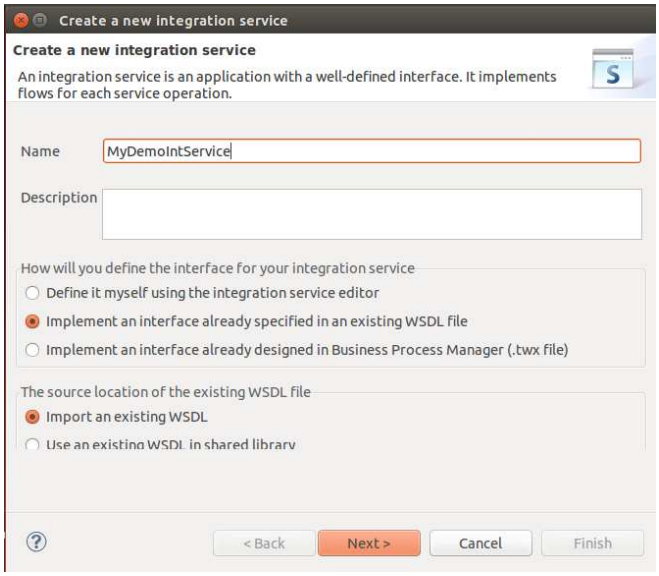


Give the service a name.

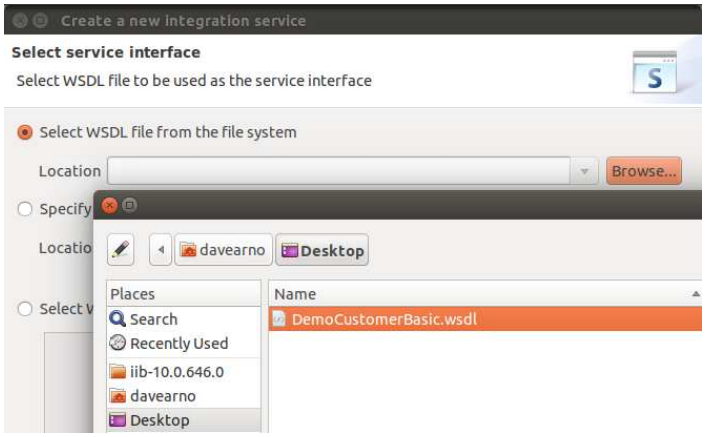
Select “Implement an interface already defined in a WSDL file”

Select “Import an existing WSDL file”

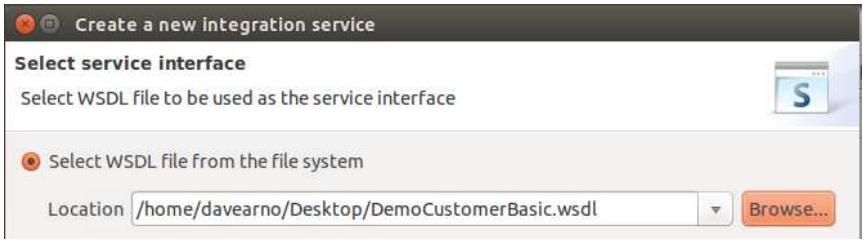
Let the remaining radio buttons default as shown below then click Next



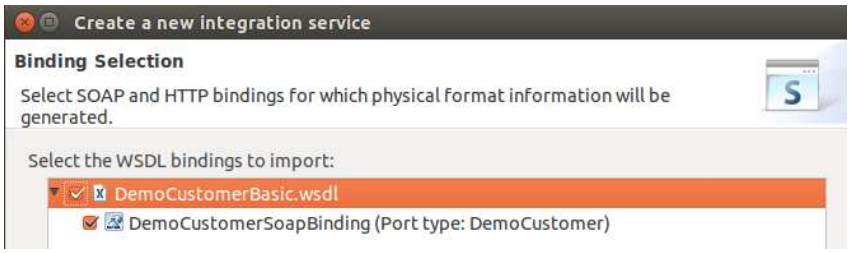
Select the sample WSDL file DemoCustomerBasic.wsdl by clicking Browse and then opening the file.



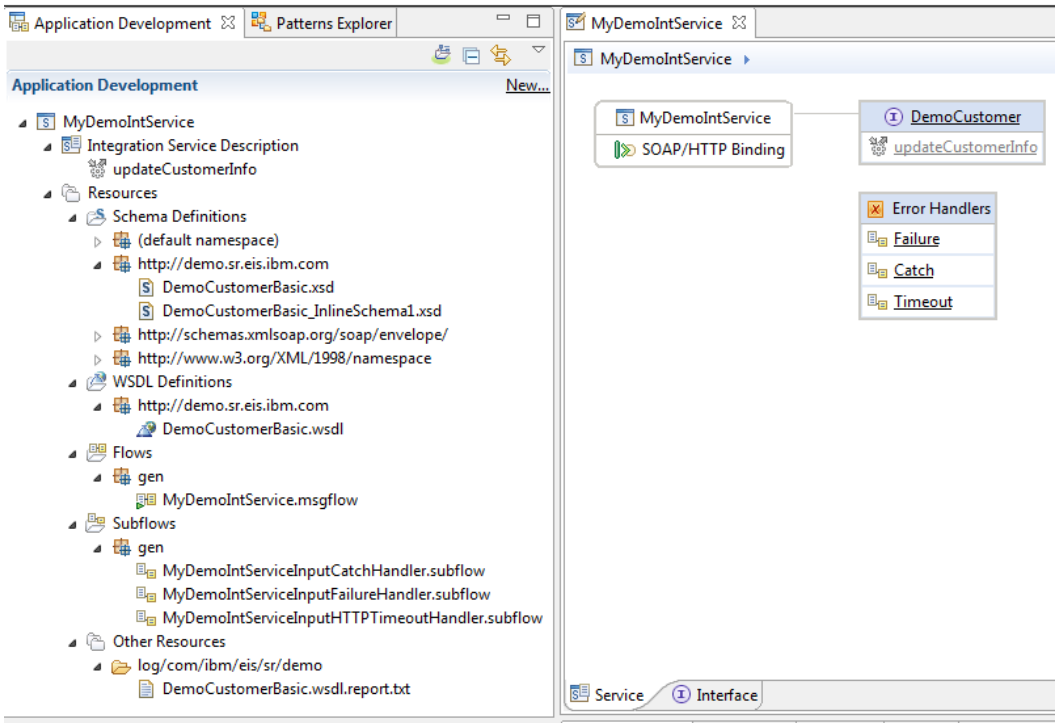
Click OK and then Next



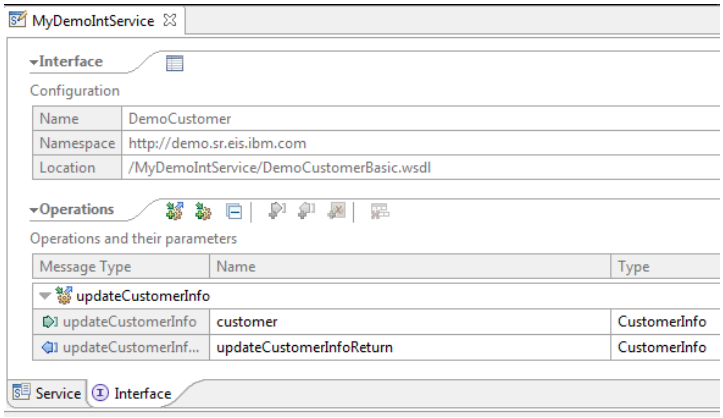
Select the binding to import (there is only 1) then click Finish and the IIB Studio will build the skeleton service for you.



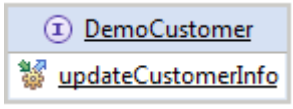
Explore the generated artefacts – the MyDemoIntService Integration Service Description is open in the right hand pane in the Service view.



Click on Interface to review the interface definition

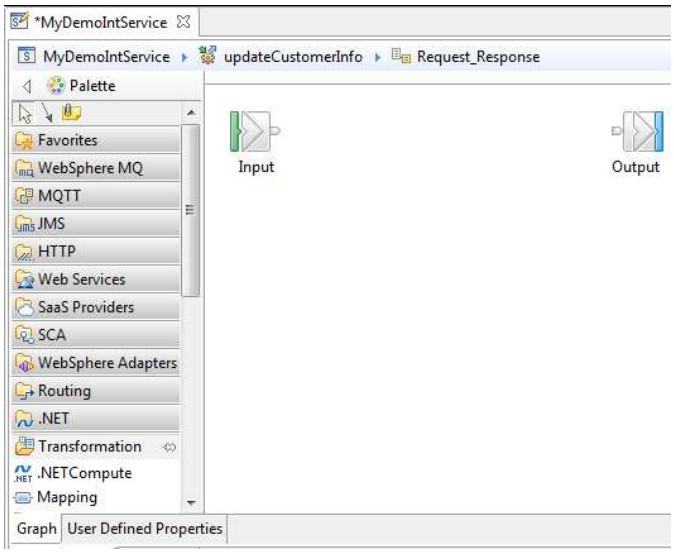


3. Build the integration flow behind the service façade

Returning to the Service view of the MyDemoIntService Integration Service Description click on the updateCustomerInfo link 

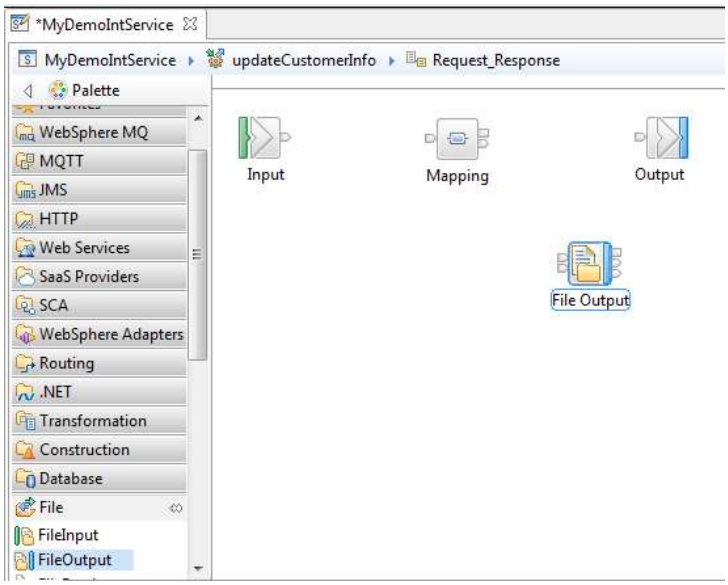
to open an IIB Message Flow editor in which you will build out the functionality for this service.

A Request_Response subflow is generated and you place your IIB functional nodes between the Input and Output terminals.



Select a Mapping node from the Transformation folder in the palette (we will use this to build the updateCustomerInfoResponse based on the updateCustomerInfo request data plus some hard coded values) and place it between the Input and Output terminals.

Select a FileOutput from the File folder (we will use this to log our customer info data to file) and place it below the Output terminal

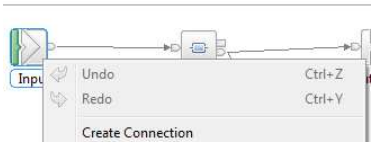


Wire the terminals and nodes together as follows:

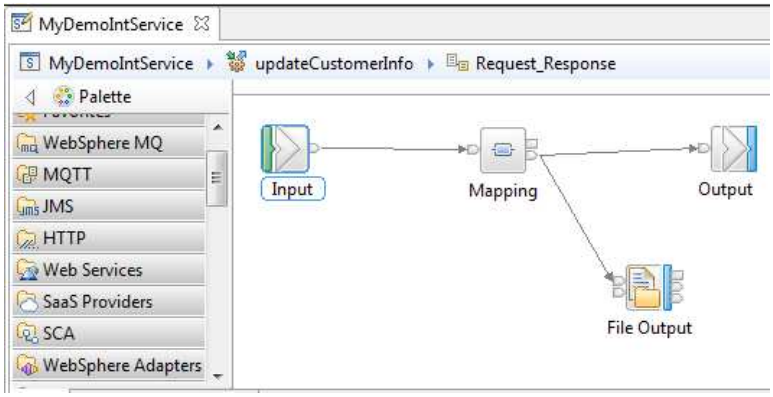
a) Input Node (Out terminal) to Mapping Node (In terminal)

b) Mapping node (Out terminal) to Output Node (In terminal)

c) Mapping node (Out terminal) to File Output Node (In terminal)

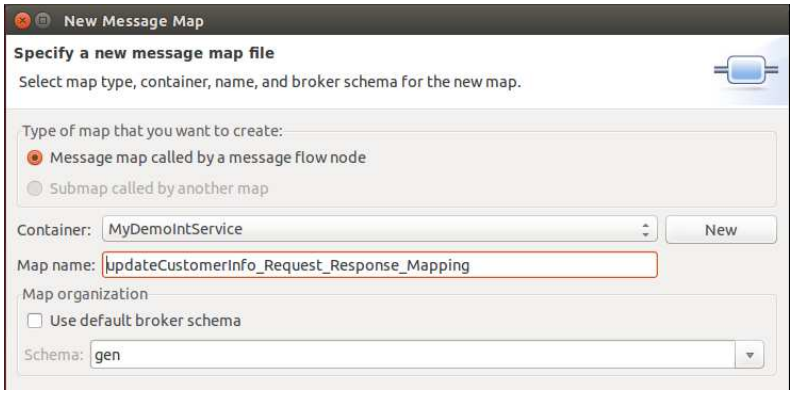
You can hover over the terminals with the mouse or right click on a node and select create connection 

The result should be as show below



4. Create the mapping to map from the Customer Request object to the Customer Response object and hard code values for city and country.

Double click on the Mapping node to open the map creation wizard, accept the defaults and click next

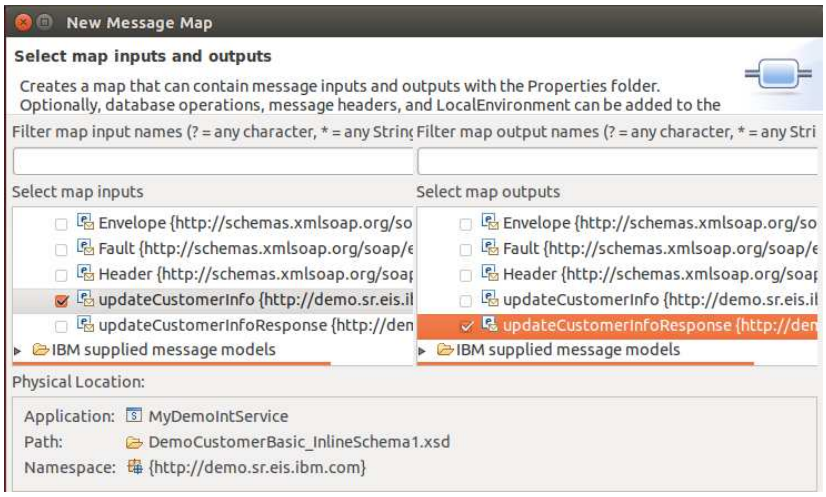


Select the map Inputs and Outputs.

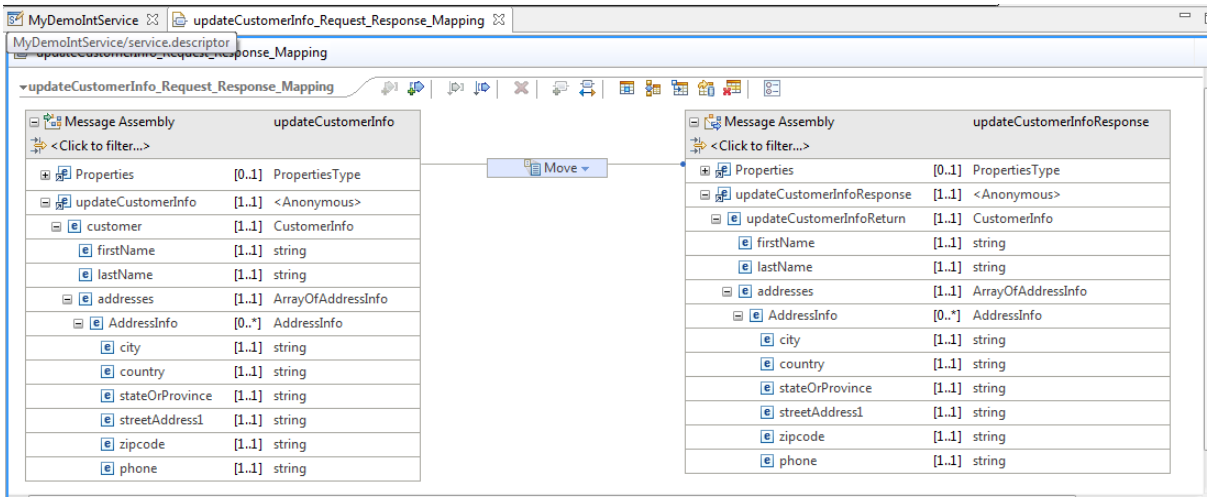
Expand the service name node then expand DFDL and XML Schemas node in each side of the mapping wizard window

Map Input = updateCustomerInfo, Map Output = updateCustomerInfoResponse

Then click finish

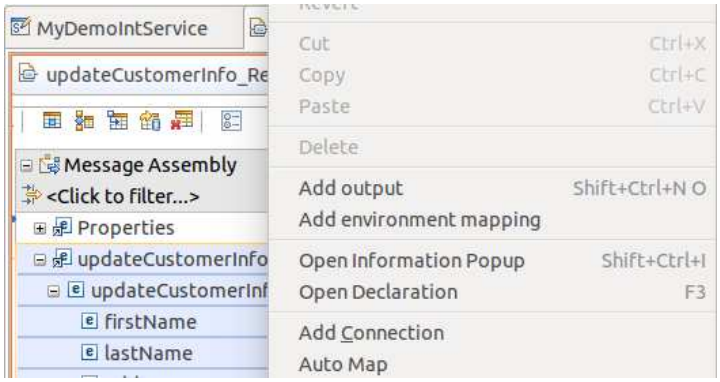


In the mapping editor, expand the input and output elements to the review the structures

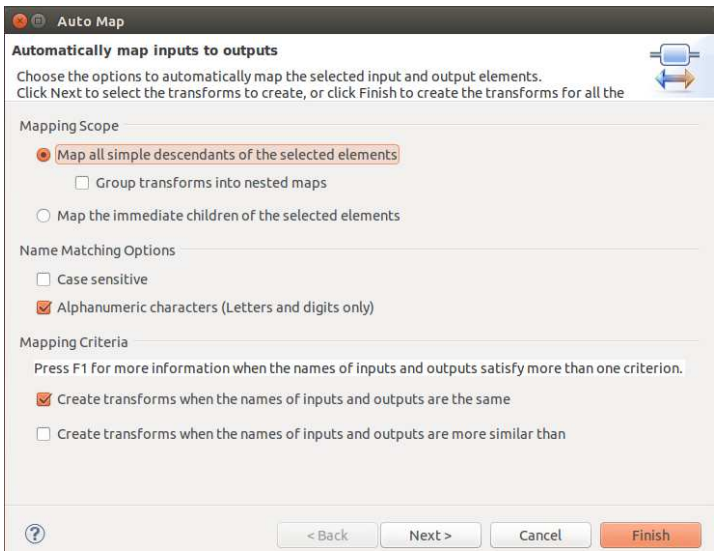


Use Auto Map to do the hard work for us.

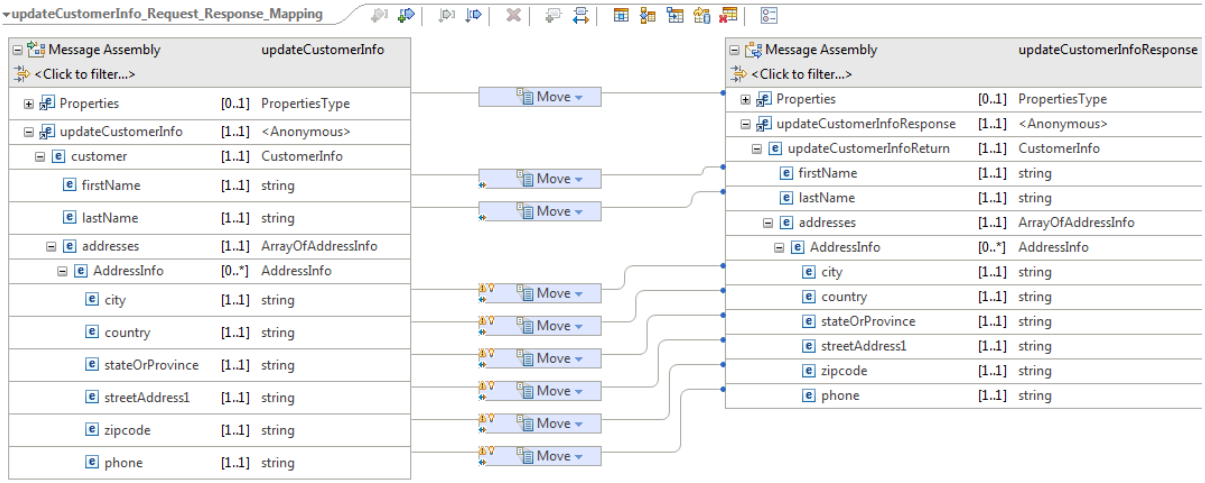
Right click on updateCustomerInfoResponse on the right side and select Auto Map.



Accept the defaults and click Finish

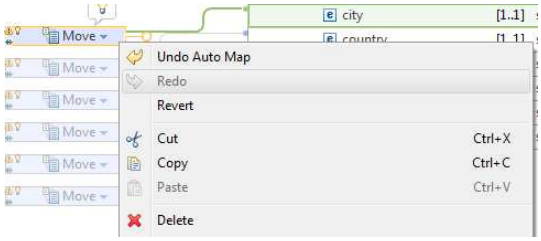


The result should be as follows

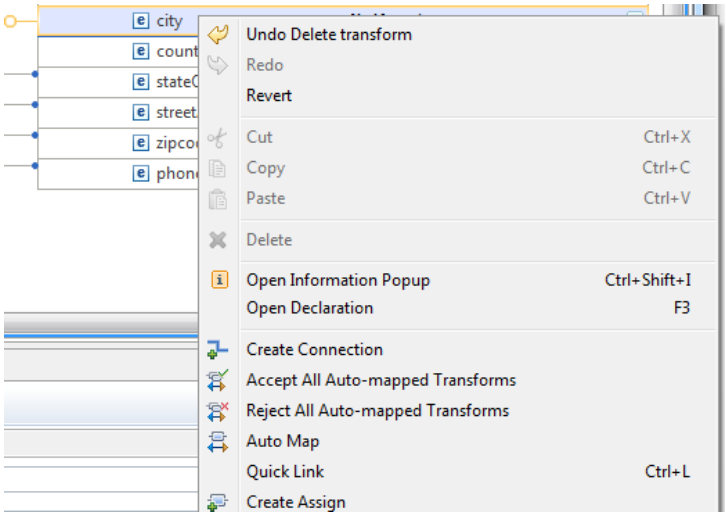


We will be hard coding the city and country so delete the Move “boxes” associated with these elements and create assigns for them.

Right click on the Move box for city and select delete, then repeat for country

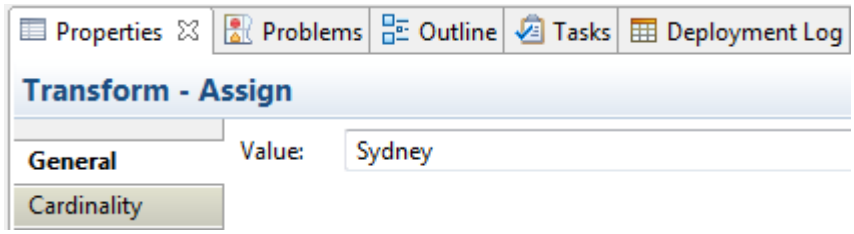


Right click on the city element on the output side and select Add Assign, then set a value

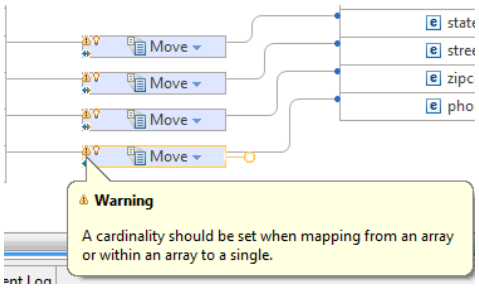


Fill in a city name in the properties.

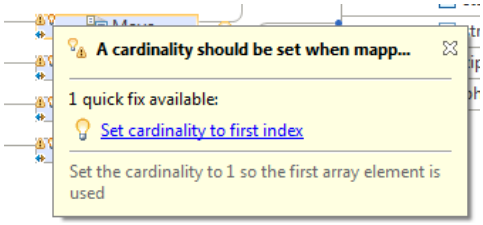
Repeat this for the country element



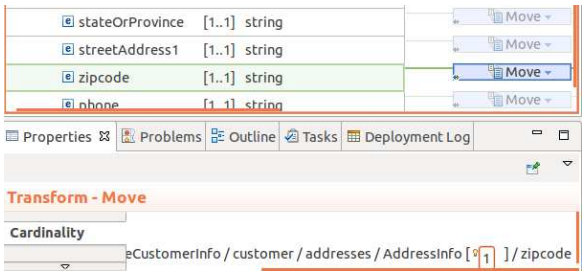
Finally, because the AddressInfo object is an array we have warnings that out Move statements do not have a cardinality. You can hover the mouse over the yellow warning sign to read the message.

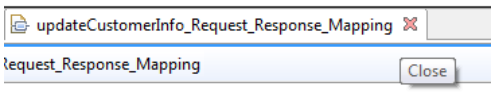


Use QuickFix to set a cardinality. Hover the mouse over the Lightbulb icon on the Move box for the state element. The click on the Set cardinality to first index link to “fix” the problem. Repeat for the other three elements.



Hit enter to accept the quick fix. This will place a 1 in the cardinality for that move.



Hit Ctrl-S to save the completed map. Close the map 

5. Configure the File Output node to log a record of the output customer info data to file.

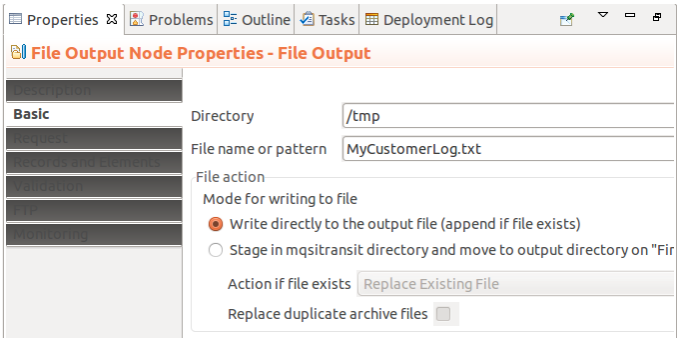
Click on the File Output Node and configure its properties.

Set a Directory for the file to reside in. Make sure this directory already exists, or create the directory in your file system. For example /tmp

Set the File name.

Select radio button for “Write directly to the output file(append if file exists)”

All other properties on all tabs can be left as default.



Hit Ctrl-S to save MyDemoIntService. The development exercise is now completed.

Deploying and testing the IIB Integration Service.

The IIB Studio has an integrated test client that can be used to:

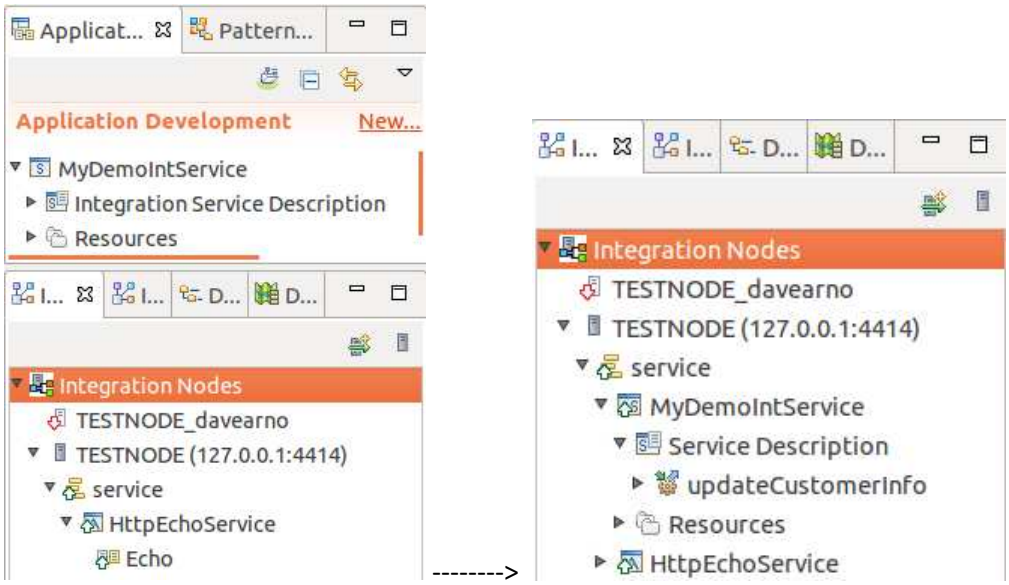
a) Deploy
b) Create test data
c) Invoke the Integration Service with the test data
d) Capture response data.

Before we use the test client, for demonstration purposes only, the next couple of steps show you how to manually deploy and remove an Integration Service from the runtime.

Deploying to the runtime.

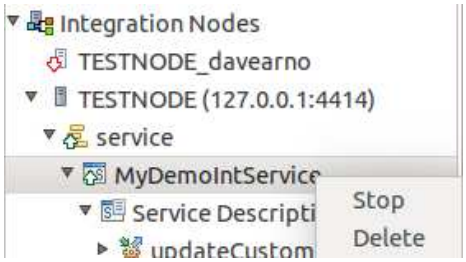
Expand the TESTNODE Node object to reveal the service integration server object

Click on the MyDemoIntService and drag and drop it onto the TESTNODE ->service integration server.



Removing artefacts from the runtime.

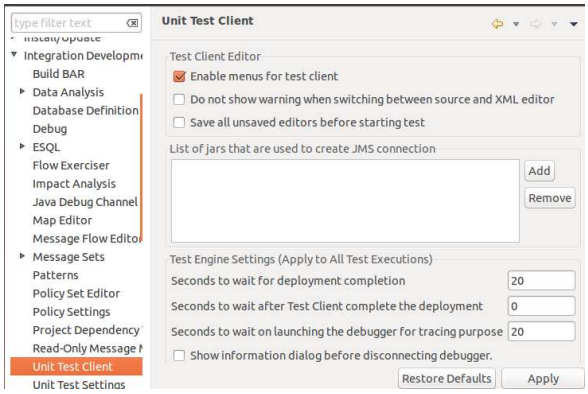
Click on the MyDemoIntService under the TESTNODE -> service integration server and select delete.



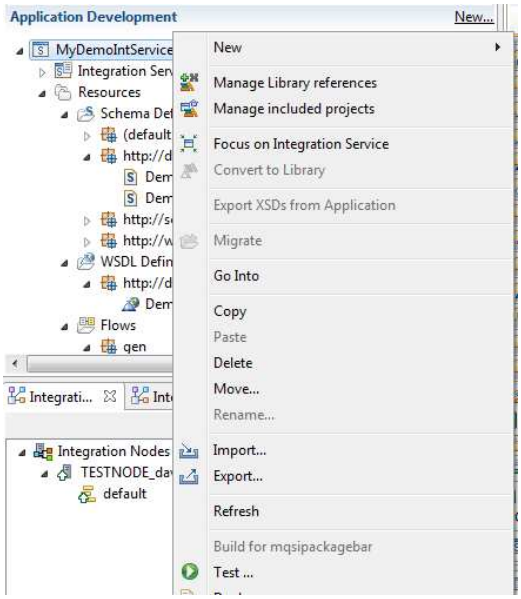
Deploying and testing using the Test Client.

Ensure that the test client facility is enabled

I. Select the menu item Window → Preferences
II. In the left hand column expand Integration Development
III. Select Unit Test Client
IV. Tick the 'Enable menus for test client' box



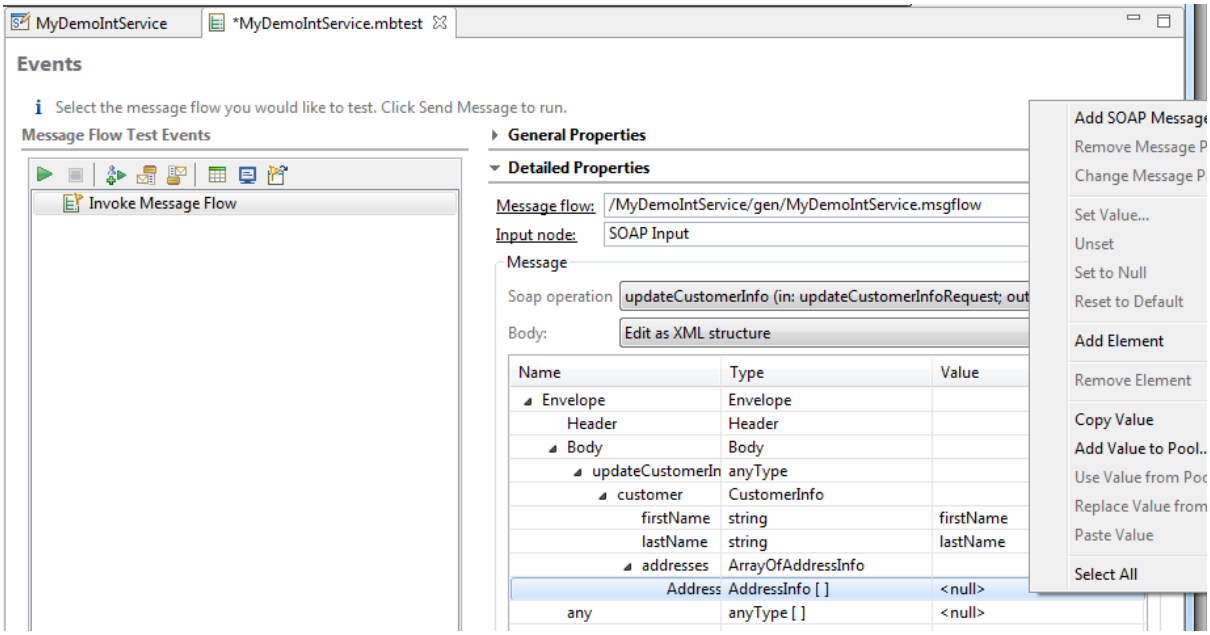
Right click on MyDemoIntService in the top left of the IIB Studio and select Test



A myDemoIntService.mbtest test client window is opened. The test client populates the test data based on the service definition of the myDemoIntService integration service. Notice it picks up the Transport type, SOAP operation and builds sample input data.

We set some of the input fields with our own data. Right click on the AddressInfo element

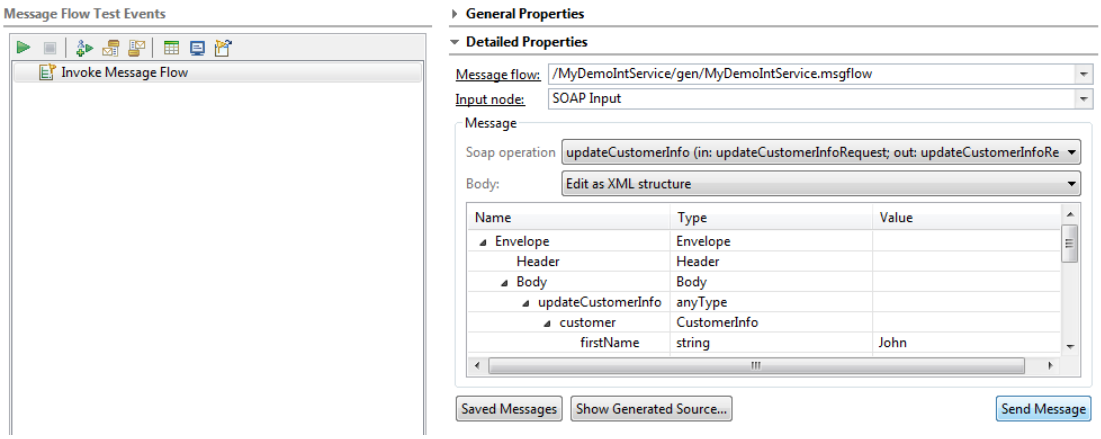
and select Add Element as shown below.



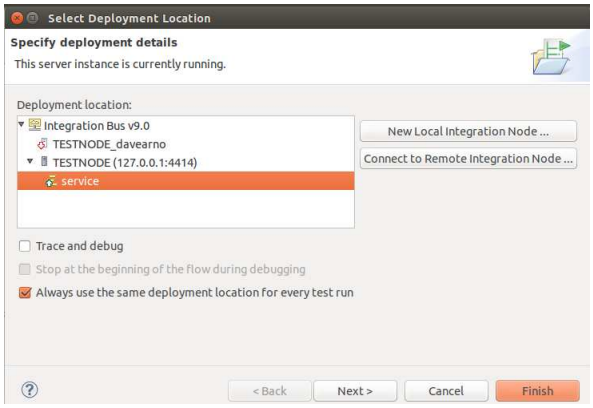
Put in some data for firstName, secondName, zipcode and phone. Leave city and country as-is

Name	Type	Value
Envelope	Envelope	
Header	Header	
Body	Body	
updateCustomerInfo	anyType	
customer	CustomerInfo	
firstName	string	John
lastName	string	Smith
addresses	ArrayOfAddressInfo	
AddressInfo	AddressInfo [ ]	
AddressInfo[0]	AddressInfo	
city	string	city
country	string	country
stateOrProvince	string	stateOrProvince
streetAddress1	string	streetAddress1
zipcode	string	2101
phone	string	12345

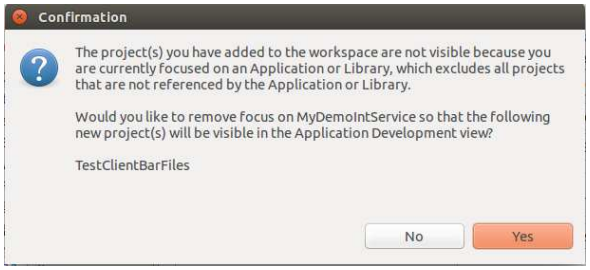
We will now deploy and test. Hit the send message button



Select the local TESTNODE->service integration server and click finish

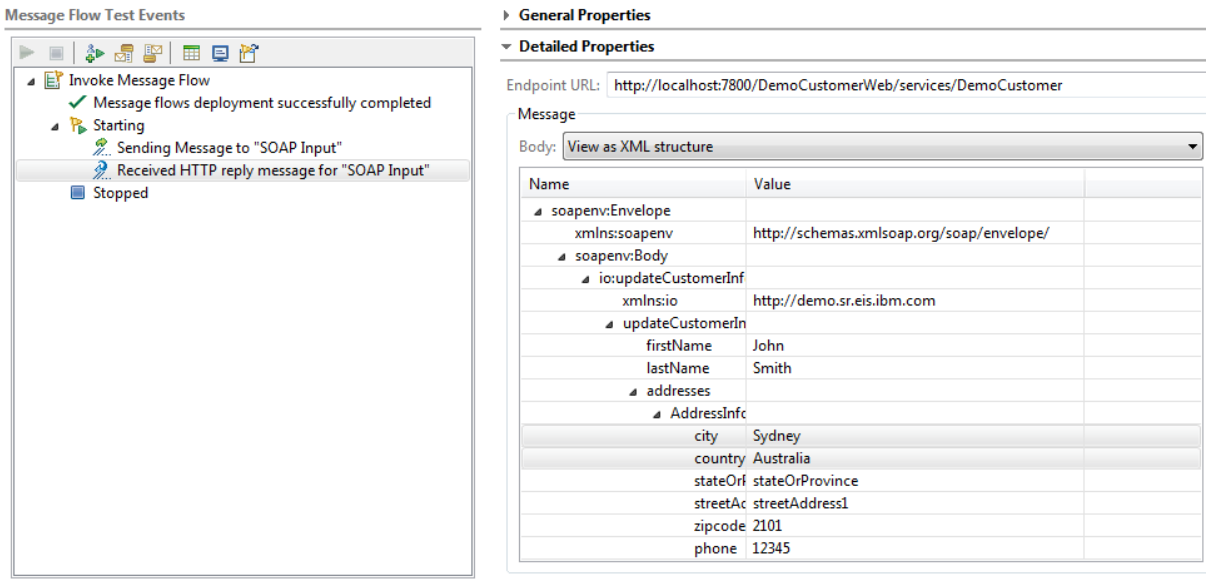


If asked for confirmation, Hit “Yes”



The test client deploys the Integration Service and runs the test.

The results should be as shown below.



Finally, let's take a look for the log file in the file system

```
$docker exec -it dev_esb01 /bin/bash -lc 'tail -f  /tmp/MyCustomerLog.txt'
```

```
davearno@ubuntu:~/Downloads/iib-10.0.646.0$ docker exec -it dev_esb01 /bin/bash
-lc 'tail -f  /tmp/MyCustomerLog.txt'
```

```
MQSI 10.0.0.0 BETA
/opt/ibm/iib-10.0.646.0/server

<io:updateCustomerInfoResponse xmlns:io="http://demo.sr.eis.ibm.com" xmlns:xsi="
http://www.w3.org/2001/XMLSchema-instance"><updateCustomerInfoReturn><firstName>
da</firstName><lastName>arn</lastName><addresses><AddressInfo><city>Sydney</city
><country>Australia</country><stateOrProvince xsi:nil="true"/><streetAddress1 xs
i:nil="true"/><zipcode xsi:nil="true"/><phone xsi:nil="true"/></AddressInfo></ad
dresses></updateCustomerInfoReturn></io:updateCustomerInfoResponse><io:updateCus
tomerInfoResponse xmlns:io="http://demo.sr.eis.ibm.com"><updateCustomerInfoRetur
n><firstName>John</firstName><lastName>Smith</lastName><addresses><AddressInfo><
city>Sydney</city><country>Australia</country><stateOrProvince>stateOrProvince</
stateOrProvince><streetAddress1>streetAddress1</streetAddress1><zipcode>2101</zi
pcode><phone>12345</phone></AddressInfo></addresses></updateCustomerInfoReturn><
/io:updateCustomerInfoResponse>
```

You have now completed your first build and test of an IBM Integration Bus service