Pymma Consulting Papers

IT Architecture

PYMMA

Tutorials JBI & Open-ESB

An easy way for intermediation

Paul Perez & Bruno Sinkovic

Pymma (2008)

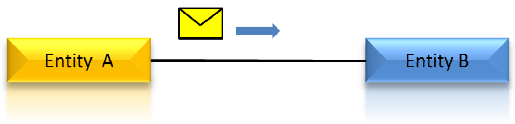
To the readers *: Since my native tongue is French and my English is far from perfect, I had the choice between writing papers in an excellent French understandable by French speakers or in a very bad English understandable by a great majority of IT people. Even if I look foolish, I choose the second option to be understandable by the maximum of people. (Paul Perez)*

Abstract

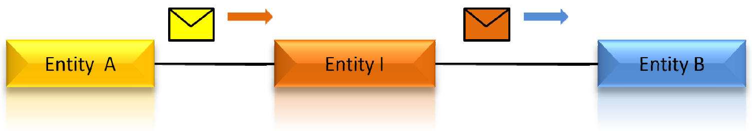
In this paper we propose a simple method to set Open-Esb applications as intermediate between provider and consumer of services with a minimal disturbance for the two tiers.

Intermediation definition

Let's suppose two entities A and B communicating together. In a regular way, A sends message to B and B receives message from A.



Intermediation is the case where a third entity named I, exists between A and B and intercept the messages. As a result, A and B don't communicate directly anymore. A sends messages to I, then I forwards the messages to B. between the two steps, I modify, enhance, upgrade or reroute the messages if needed.



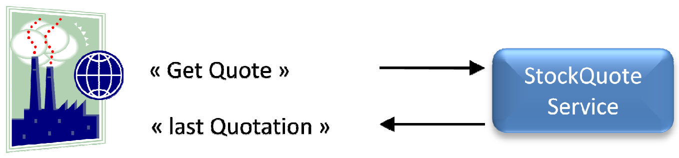
Intermediation is a very common pattern in IT development. It is used to control, manage and enhance communication between two entities or to create loose coupling and indirection. There are many other purposes for intermediation but this topic is beyond the scope of this paper.

Tutorial principles and agenda

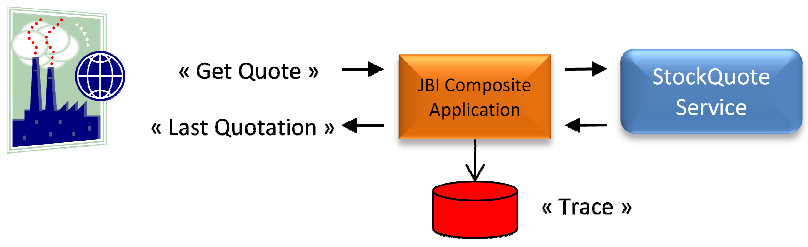
As example in this tutorial, we will use a simple public web service "stockQuote", provided by "webservicex.net". It returns the last quote of a stock. This web service can be found on internet: http://www.webservicex.net/stockquote.asmx?wsdl .

Use case

A company named "XXcompany" regularly invokes "StockQuote service" and for internal statistic reports,



"XXCompany" management wants to trace all the responses returned by "stockQuote".

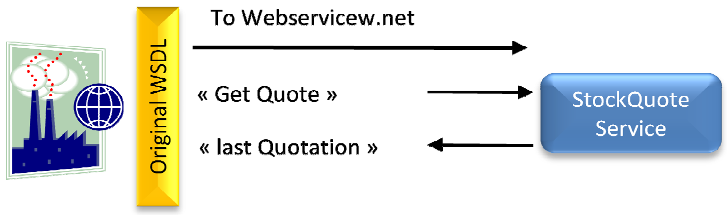


In order to achieve this requirement, we will create a JBI Application (as intermediate) that traces “stockQuote” responses in a disk file. The application must be as transparent as possible for consumer and producer and must not involve modifications in Web services signatures.

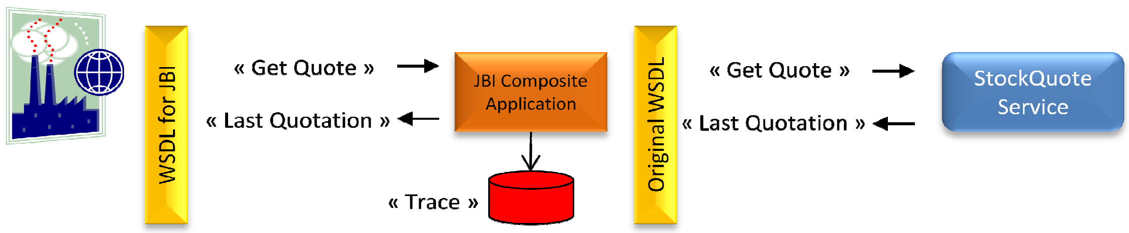
Technique for Intermediation

Tutorial principle is easy to understand. Each web service is defined by a WSDL document which is made up of two distinct parts named “abstract part” and “concrete part”. At 30000 feet, “Abstract part” could be seen as a service definition (methods, operations, messages, parameters types...) and the “concrete part” locates the service and specifies which protocols can be used to invoke it.

In the original configuration (without intermediate), the consumer uses a WSDL document provided by WebServiceX.net wherein soap address points at “http://www.webservicex.net/stockquote.asmx”. By using the original configuration, , “XXCompany” invokes webservicex.net server to get a quote.

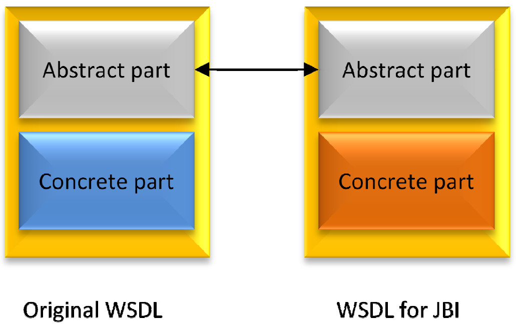


in the second configuration “with JBI as Intermediate”, “XXCompany” uses a new WSDL document wherein Soap address points at JBI Application. It replaces the original WSDL that will be use by JBI from now on.



WSDL copy and modification

Since we don’t want to change the interfaces of the service, the “original WSDL” and the “WSDL for JBI” will have the same abstract parts. But in order to achieve the requirements, the WSLDs concrete parts will be different. To define the “WSDL for JBI”, we will copy the original WSDL and change the concrete part to redirect the service requests to JBI application. An additional minor modification is required in the “WSDL for JBI” namespaces, if the two WSDLs would have the same namespace, the BPEL engine could mix up “Original WSDL” and “WSDL for JBI” parameters.



Tutorial agenda

The table below lists the different steps needed to set open-ESB as intermediate.

Step 1	Import the original Web service
Step 2	Create a WSDL copy for JBI
Step 3	Create a WSDL for Tracking
Step 4	Create BPEL Scenario
Step 5	Create Composite application
Step 6	Deploy and Test

Prerequisites

When we wrote this tutorial we supposed that:

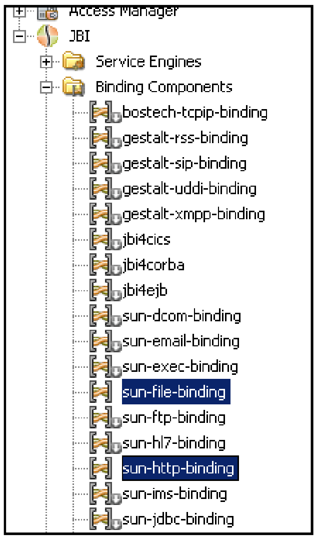
- Your machine has an Internet access
- You already install NetBeans development 2008030100002
- You are familiar with JBI and Open-ESB concepts
- You are able to read and understand XML Schemas, WSDL and BPEL documents.
- You already worked with Open-ESB

Note: If you are a novice in JBI, we propose you to read and work on Open ESB Tutorial before (http://www.netbeans.org/kb/trails/soa.html).

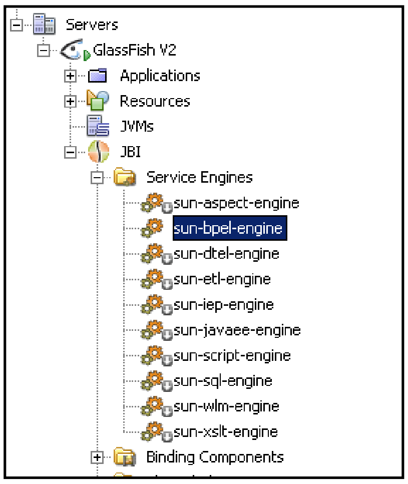
Before starting our tutorial

Before starting the tutorial:

1. Start Open-ESB
2. Check if Glassfish is started on
3. Check if BPEL Service is started on
4. Check if Http Binding and File Binding components are started on



```
Access Manager
JBI
  Service Engines
  Binding Components
    bostech-tcpip-binding
    gestalt-rss-binding
    gestalt-sip-binding
    gestalt-uddi-binding
    gestalt-xmpp-binding
    jbi4cics
    jbi4corba
    jbi4ejb
    sun-dcom-binding
    sun-email-binding
    sun-exec-binding
    sun-file-binding
    sun-ftp-binding
    sun-hl7-binding
    sun-http-binding
    sun-ims-binding
    sun-jdbc-binding
```

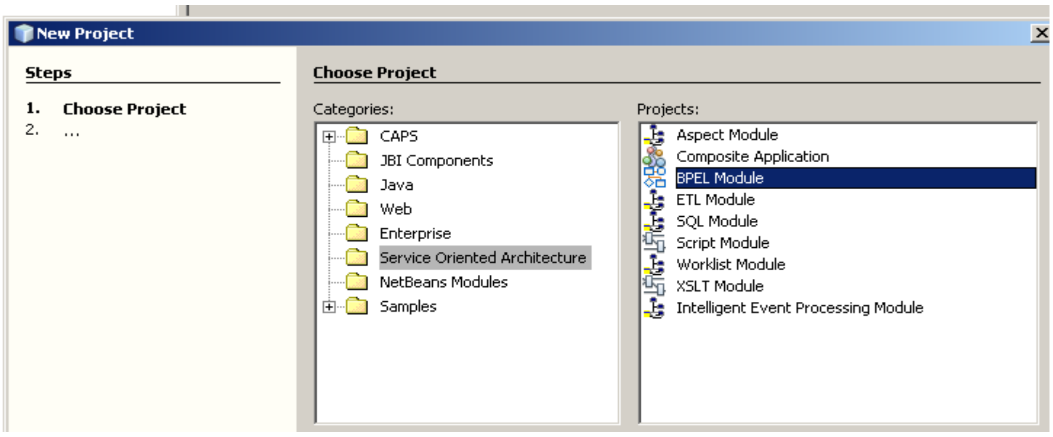


```
Servers
  GlassFish V2
    Applications
    Resources
    JVMs
    JBI
      Service Engines
        sun-aspect-engine
        sun-bpel-engine
        sun-dtel-engine
        sun-etl-engine
        sun-iep-engine
        sun-javaee-engine
        sun-script-engine
        sun-sql-engine
        sun-wlm-engine
        sun-xslt-engine
      Binding Components
```

If you experience difficulties to start the Glassfish server or JBI components, please refer to Sun's documentation.

Step 1 : Import the original Web service

1. With Netbeans, create a BPEL project



New Project

Steps

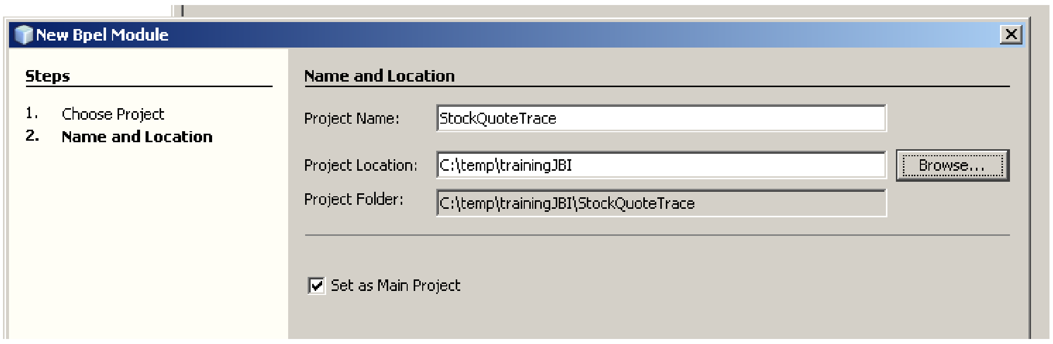
1. **Choose Project**
2. ...

Choose Project

Categories:
- CAPS
- JBI Components
- Java
- Web
- Enterprise
- Service Oriented Architecture
- NetBeans Modules
- Samples

Projects:
- Aspect Module
- Composite Application
- BPEL Module
- ETL Module
- SQL Module
- Script Module
- Worklist Module
- XSLT Module
- Intelligent Event Processing Module

2. Give a name and a location to the project



New Bpel Module

Steps

1. Choose Project
2. **Name and Location**

Name and Location

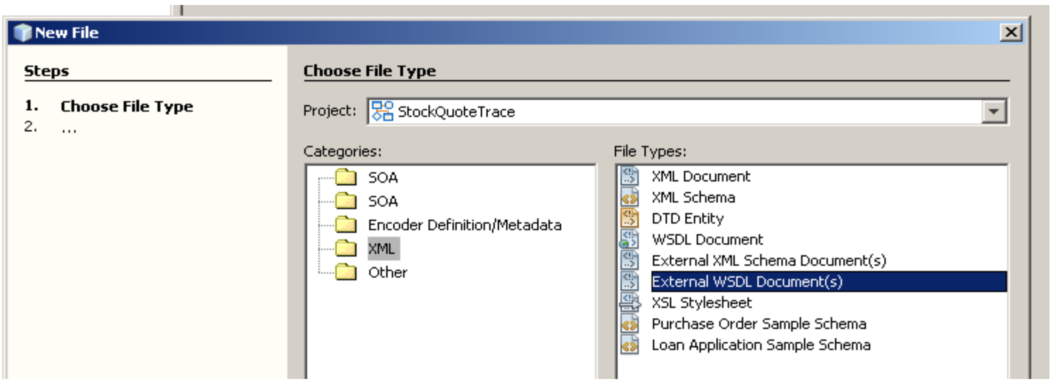
Project Name: StockQuoteTrace

Project Location: C:\temp\trainingJBI Browse...

Project Folder: C:\temp\trainingJBI\StockQuoteTrace

[✓] Set as Main Project

3. From the folder Process file import an external WSDL



New File

Steps

1. **Choose File Type**
2. ...

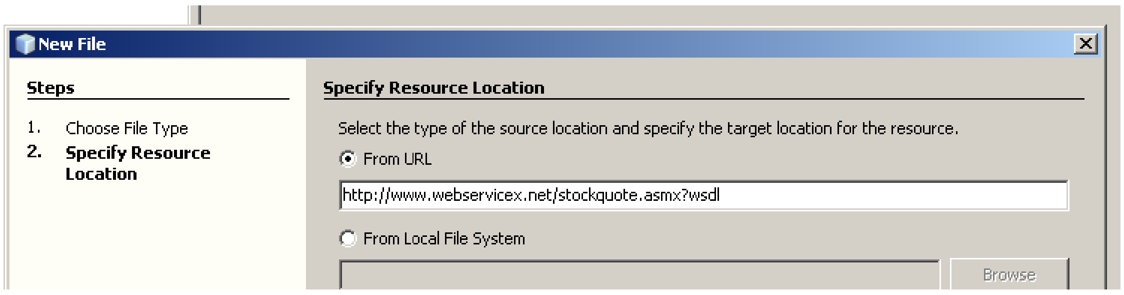
Choose File Type

Project: StockQuoteTrace

Categories:
- SOA
- SOA
- Encoder Definition/Metadata
- XML
- Other

File Types:
- XML Document
- XML Schema
- DTD Entity
- WSDL Document
- External XML Schema Document(s)
- External WSDL Document(s)
- XSL Stylesheet
- Purchase Order Sample Schema
- Loan Application Sample Schema

4. Type the URL of the WSDL

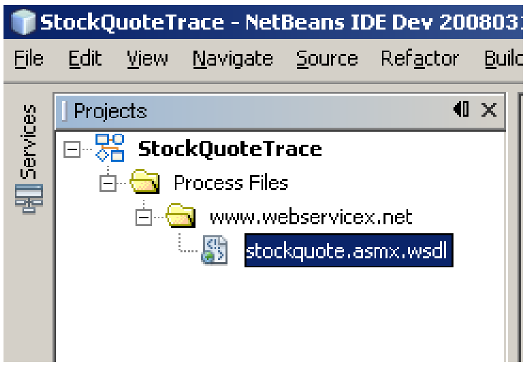


Select the type of the source location and specify the target location for the resource.

From URL: http://www.webservicex.net/stockquote.asmx?wsdl

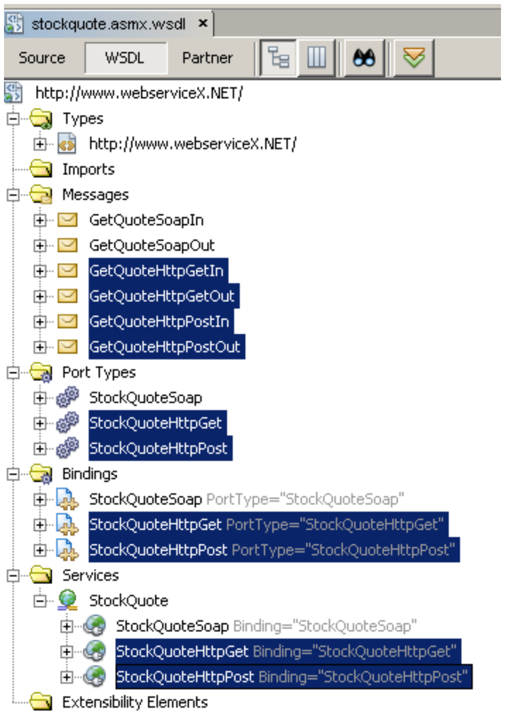
From Local File System

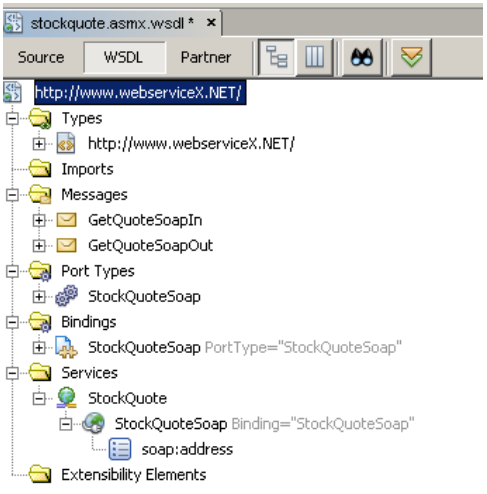
5. The imported WSDL appears in the Project Window


StockQuoteTrace - NetBeans IDE Dev 200803
File Edit View Navigate Source Refactor Build
Services
Projects
StockQuoteTrace
Process Files
www.webservicex.net
stockquote.asmx.wsdl

6. Open the WSLD Document.

In order to simplify the WSDL, we don't want to keep "Http Get" and "Http Post" Bingings. Please, suppress them and delete Messages, Port Type, Binding and Services associated with "Http Get" and "Http Put".

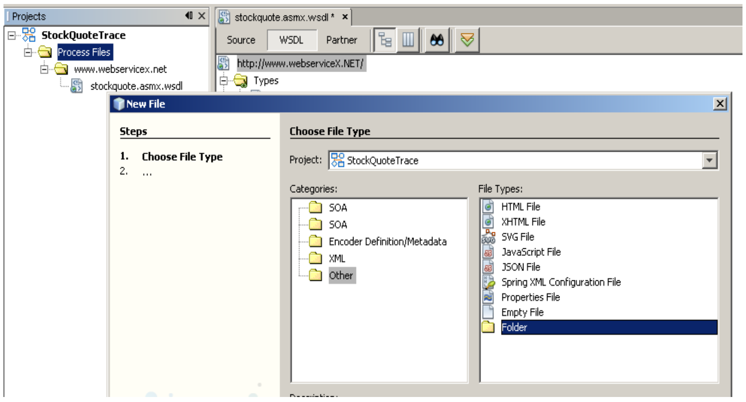

stockquote.asmx.wsdl
Source WSDL Partner
http://www.webserviceX.NET/
Types
http://www.webserviceX.NET/
Imports
Messages
GetQuoteSoapIn
GetQuoteSoapOut
GetQuoteHttpGetIn
GetQuoteHttpGetOut
GetQuoteHttpPostIn
GetQuoteHttpPostOut
Port Types
StockQuoteSoap
StockQuoteHttpGet
StockQuoteHttpPost
Bindings
StockQuoteSoap PortType="StockQuoteSoap"
StockQuoteHttpGet PortType="StockQuoteHttpGet"
StockQuoteHttpPost PortType="StockQuoteHttpPost"
Services
StockQuote
StockQuoteSoap Binding="StockQuoteSoap"
StockQuoteHttpGet Binding="StockQuoteHttpGet"
StockQuoteHttpPost Binding="StockQuoteHttpPost"
Extensibility Elements

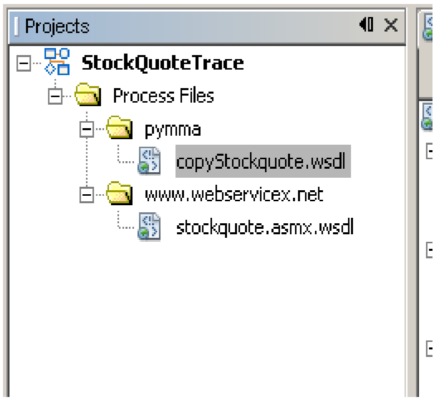


7. Save the imported WSDL

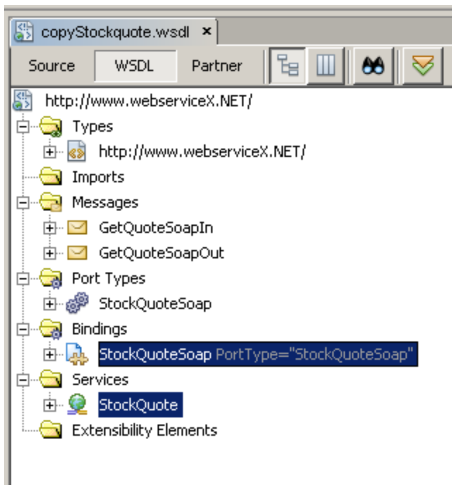
Step 2 : Create a WSDL “copy for JBI”

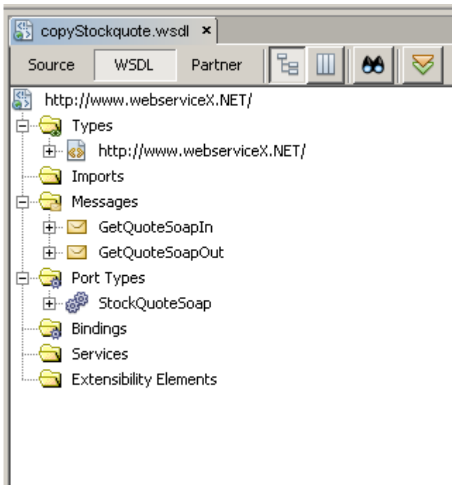
1. Create a new directory named Pymma and copy the original WSDL inside and rename it “copyStockQuote.wsdl”





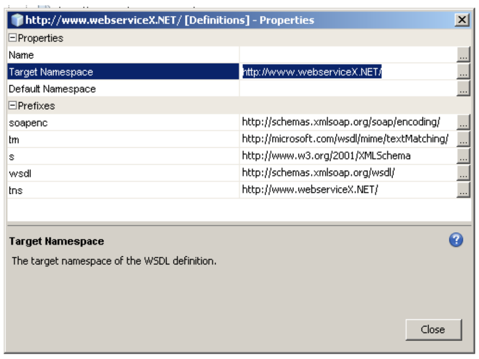
2. Open “copyStockQuote.wsdl”.
   First, suppress the Service then the Binding. WSDL concrete part is empty now. It will be specified latter.





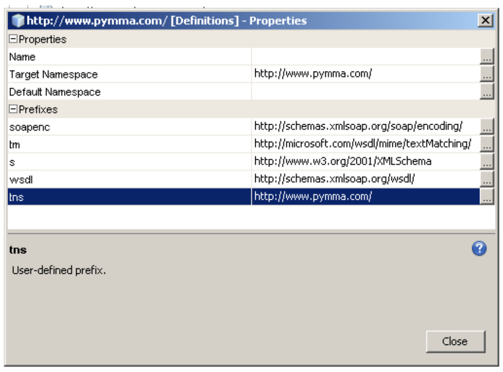
3. Right click on the root of the wsdl and select properties. Properties editor opens.
   Replace the Target namespace by “http://www.pymma.com/”.
   Note: the tns value changes automatically

.


http://www.webserviceX.NET/ [Definitions] - Properties

Properties	
Name	
Target Namespace	http://www.webserviceX.NET/
Default Namespace	
Prefixes	
soapenc	http://schemas.xmlsoap.org/soap/encoding/
tm	http://microsoft.com/wsdl/mime/textMatching/
s	http://www.w3.org/2001/XMLSchema
wsdl	http://schemas.xmlsoap.org/wsdl/
tns	http://www.webserviceX.NET/

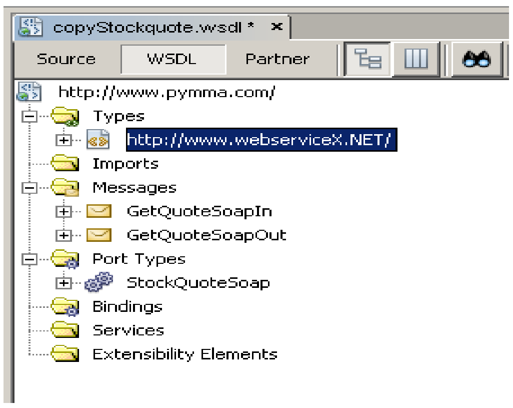
Target Namespace
The target namespace of the WSDL definition.


http://www.pymma.com/ [Definitions] - Properties

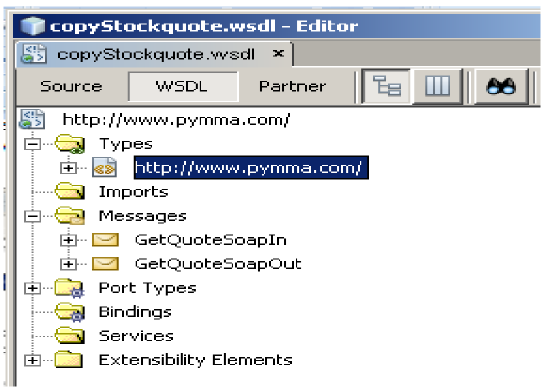
Properties	
Name	
Target Namespace	http://www.pymma.com/
Default Namespace	
Prefixes	
soapenc	http://schemas.xmlsoap.org/soap/encoding/
tm	http://microsoft.com/wsdl/mime/textMatching/
s	http://www.w3.org/2001/XMLSchema
wsdl	http://schemas.xmlsoap.org/wsdl/
tns	http://www.pymma.com/

tns
User-defined prefix.

Close the properties editor. In the WSDL editor extend the type node. Right click on the type namespace and change it to “http://pymma.com/”. Then you obtain the following WSDL

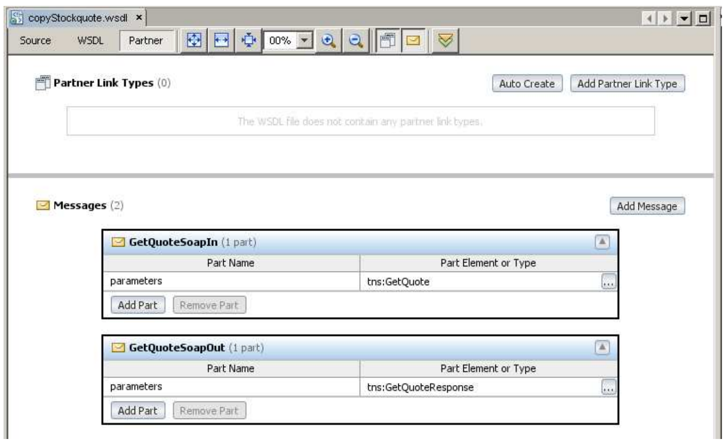


```
http://www.pymma.com/
  Types
    http://www.webserviceX.NET/
  Imports
  Messages
    GetQuoteSoapIn
    GetQuoteSoapOut
  Port Types
    StockQuoteSoap
  Bindings
  Services
  Extensibility Elements
```



```
http://www.pymma.com/
  Types
    http://www.pymma.com/
  Imports
  Messages
    GetQuoteSoapIn
    GetQuoteSoapOut
  Port Types
  Bindings
  Services
  Extensibility Elements
```

4. Click on Partner Button. Partners view editor opens



Partner Link Types (0)

The WSDL file does not contain any partner link types.

Messages (2)

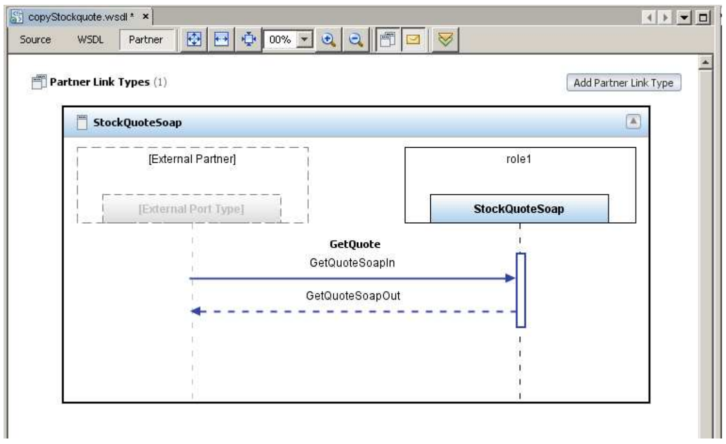
GetQuoteSoapIn (1 part)

Part Name	Part Element or Type
parameters	tns:GetQuote

GetQuoteSoapOut (1 part)

Part Name	Part Element or Type
parameters	tns:GetQuoteResponse

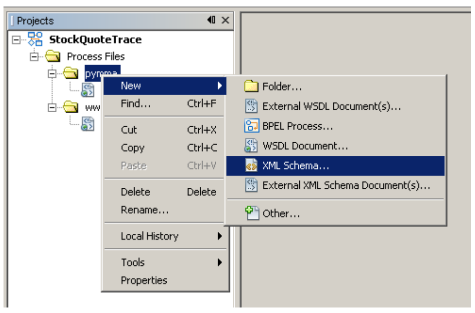
5. Click on Auto Create Button

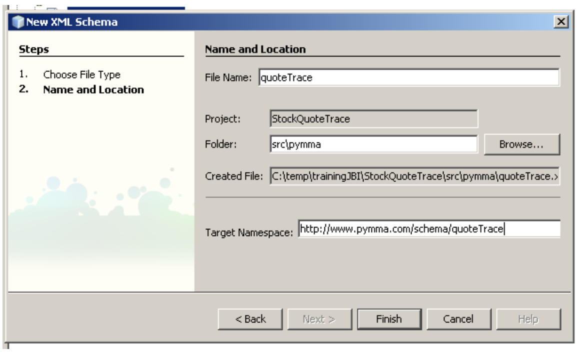


6. Save and close the WSDL

Step3: Create a WSDL for Tracking

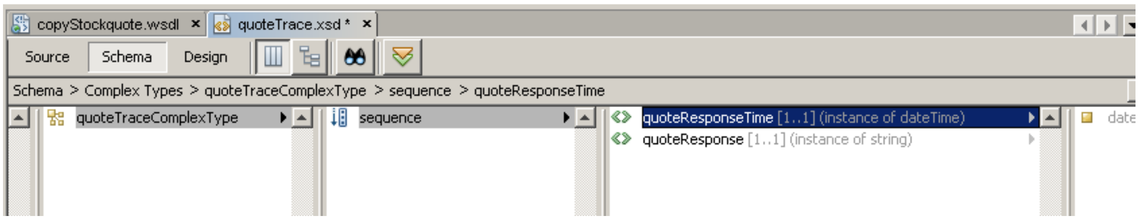
1. From the Pymma directory create a new XML Schema named "quoteTrace.xsl"





File Name: quoteTrace
Project: StockQuoteTrace
Folder: src\pymma
Created File: C:\temp\trainingJBI\StockQuoteTrace\src\pymma\quoteTrace.x
Target Namespace: http://www.pymma.com/schema/quoteTrace

2. Create a complex Type named "quoteTraceComplexType"


copyStockquote.wsdl
quoteTrace.xsd *
Source
Schema
Design
Schema > Complex Types > quoteTraceComplexType > sequence > quoteResponseTime
quoteTraceComplexType
sequence
quoteResponseTime [1..1] (instance of dateTime)
quoteResponse [1..1] (instance of string)
date

In the complex Type Sequence add the elements detailed in the table below:

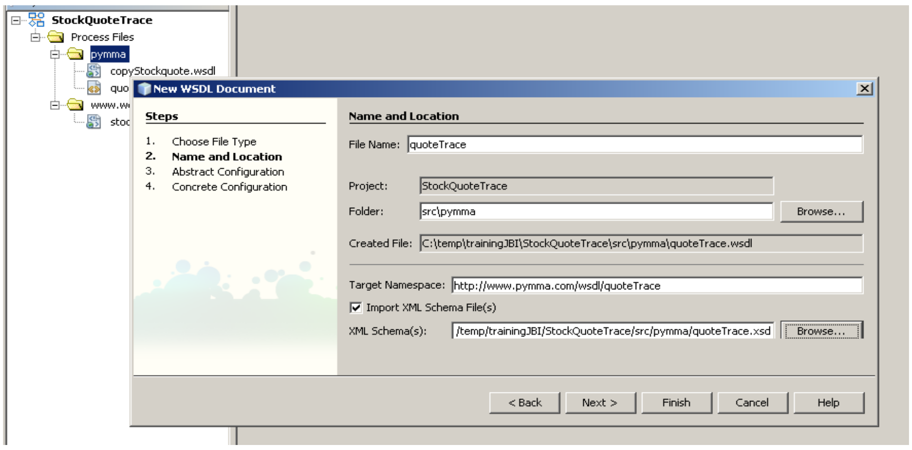
Element name	Type
quoteResponseTime	dateTime
quoteResponse	String

3. Create an Element named quoteResponse with the type "quoteTraceComplexeType".

This element is the data type that will be stored on the disk.

4. Save and close the XSL document.

5. From the Pymma directory create a new WSDL named it "quoteTrace.wsld"
   Select the option "Import XML Schema File(s)". Then select the XML Schema we just create
   .



Name and Location

File Name: quoteTrace

Project: StockQuoteTrace

Folder: src\pymma

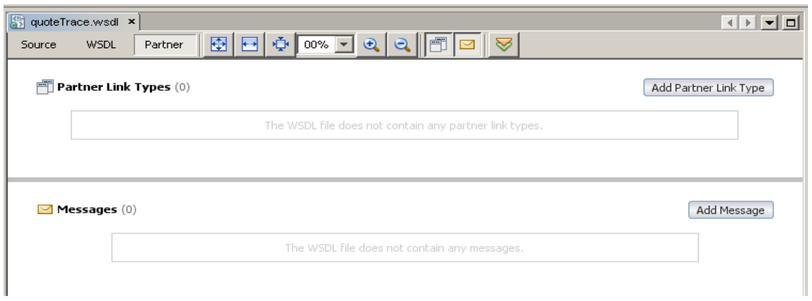
Created File: C:\temp\trainingJBI\StockQuoteTrace\src\pymma\quoteTrace.wsdl

Target Namespace: http://www.pymma.com/wsdl/quoteTrace

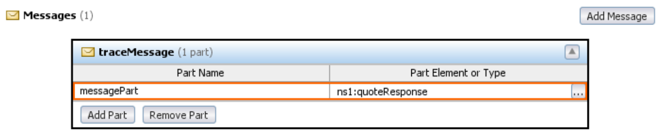
☑ Import XML Schema File(s)

XML Schema(s): /temp/trainingJBI/StockQuoteTrace/src/pymma/quoteTrace.xsd

6. Click the Finish button (not the Next Button) and the WSDL Editor opens. Click on the partner button.

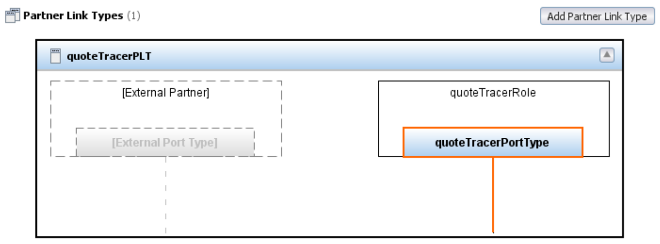


7. Click Add message Button and complete as follows
   For the "Part Element or Type" value select "quoteResponse".

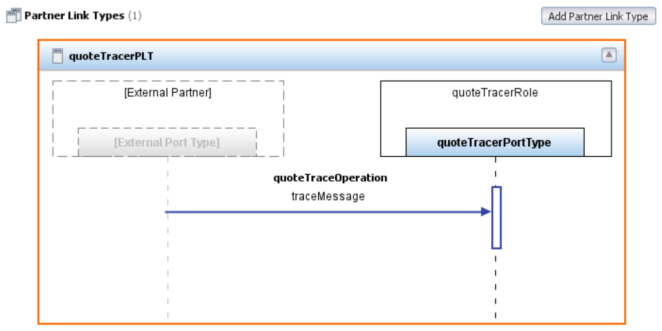


Part Name	Part Element or Type
messagePart	ns1:quoteResponse

8. Click on Add Partner Link Type button and complete the values as follows

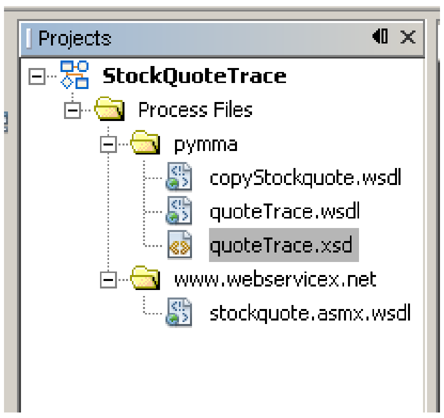


9. Open the palette windows, drag the one-way icon and drop it on the orange line under "quoteTracerPortType". Rename Operation1 into quoteTraceoperation and select "traceMessage" as message.



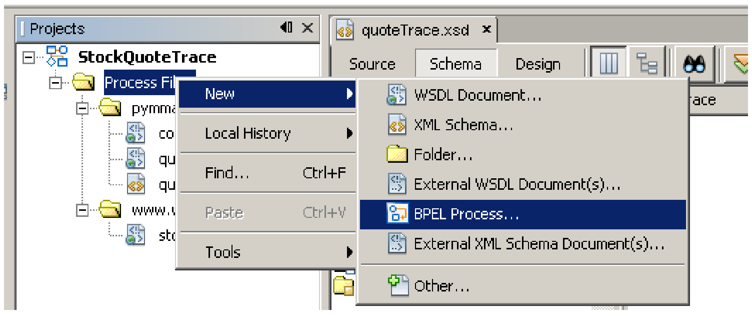
10. Save and close the WSDL editor.

Now the Project window appears as follows:



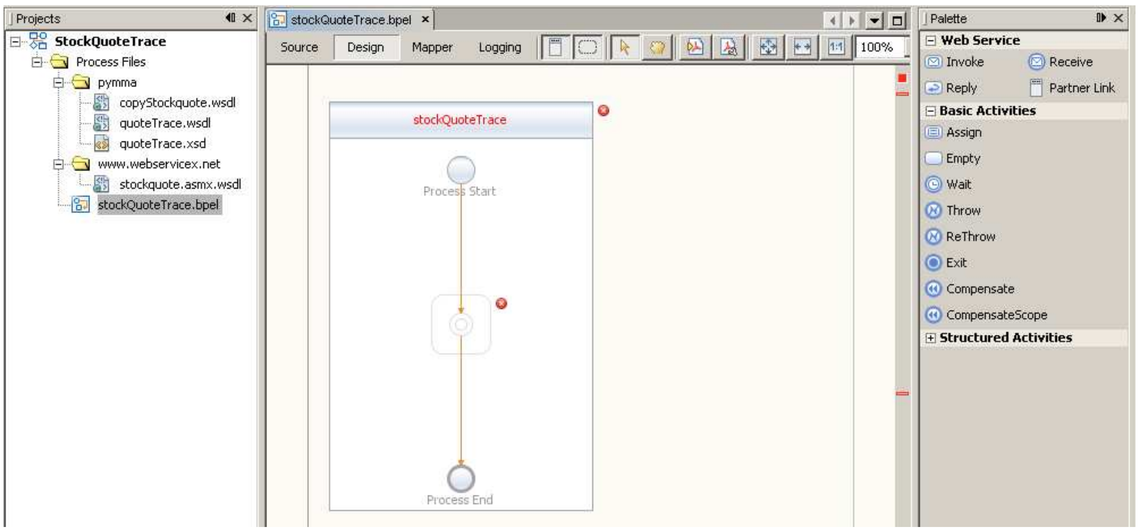
Step 4 : Create a BPEL Scenario

1. From the Process Files folder create a BPEL document:

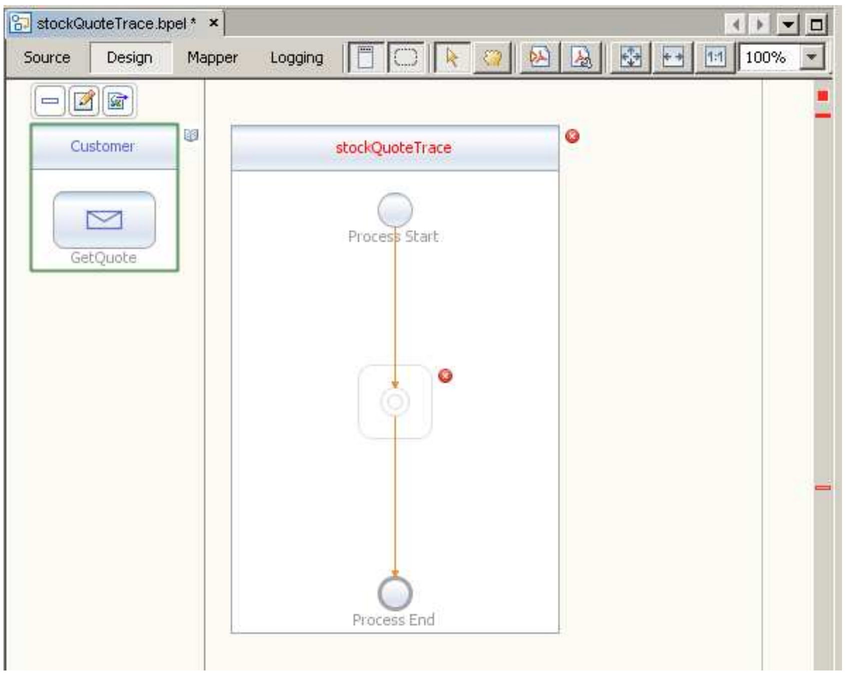


And name it “stockQuoteTrace”.

2. The project window is on the left side and the palette window on the right side. BPEL designer is in the middle. The designer is divided in three parts separated by thin vertical lines.

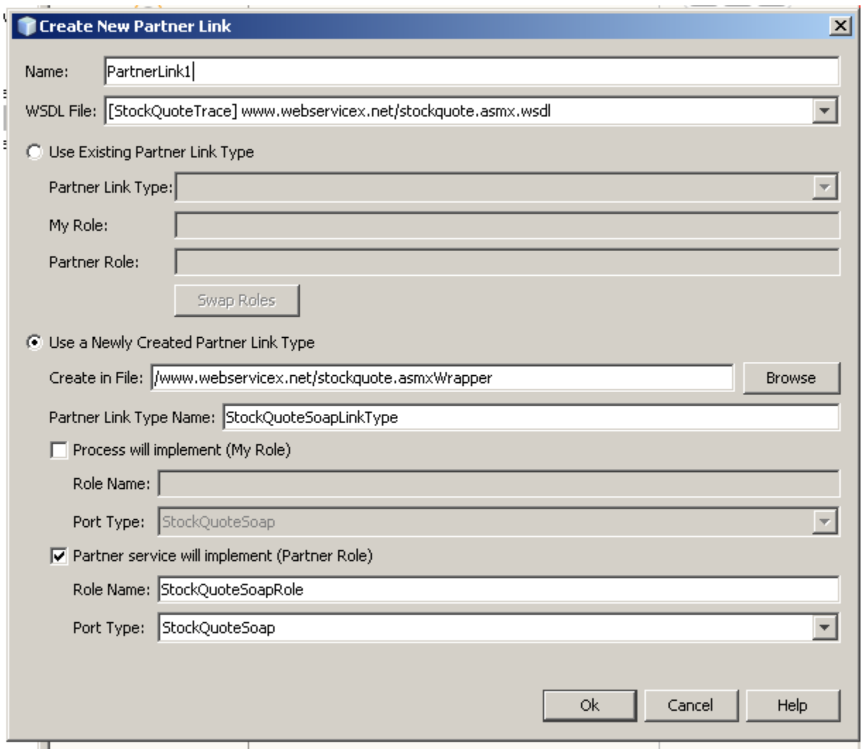


Click on the “copyStockQuote.wsdl” node, drag it and drop it on the left side of the BPEL editor. (Where an orange circle appears)
Double click on the text PartnerLink1 and rename the partner link into Customer



3. Drag the node “quoteTrace.wsdl” and drop it (on the orange circle) on the right side of the BPEL Editor. Double click on the text PartnerLink1 and rename the partner link “Tracer”

4. Drag the node “stockquote.asmx.wsdl” and drop it (on the orange circle) on the right side of the BPEL Editor. A Create new Partner Link window opens.



Create New Partner Link

Name: PartnerLink1

WSDL File: [StockQuoteTrace] www.webservicex.net/stockquote.asmx.wsdl

( ) Use Existing Partner Link Type

Partner Link Type:

My Role:

Partner Role:

(•) Use a Newly Created Partner Link Type

Create in File: /www.webservicex.net/stockquote.asmxWrapper

Partner Link Type Name: StockQuoteSoapLinkType

[ ] Process will implement (My Role)

Role Name:

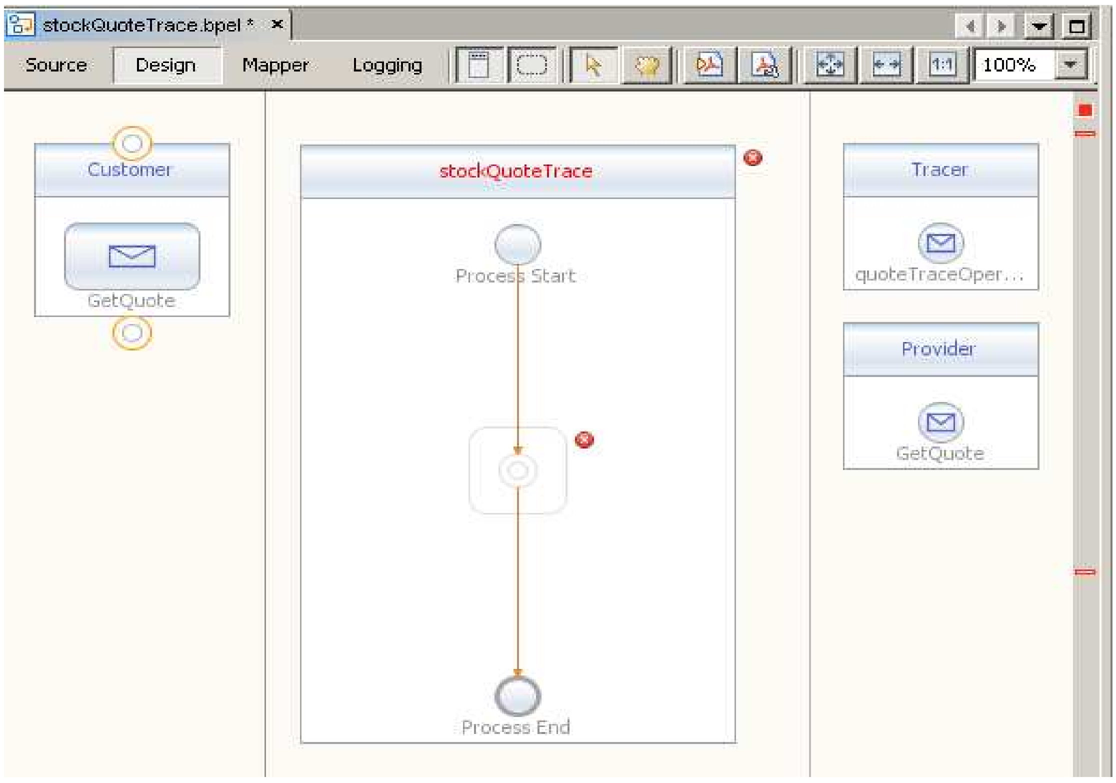
Port Type: StockQuoteSoap

[✓] Partner service will implement (Partner Role)

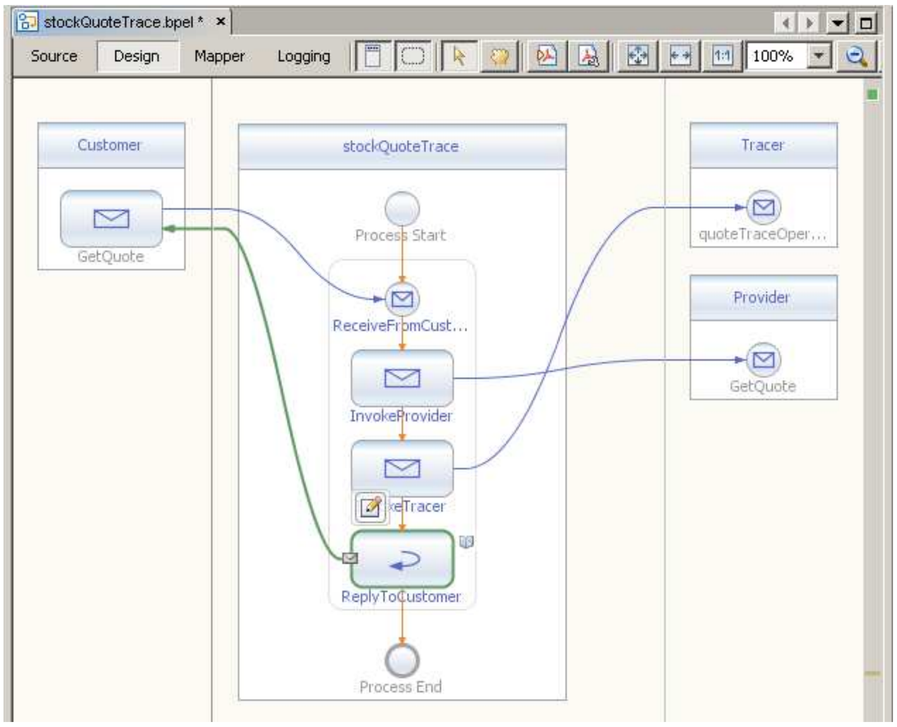
Role Name: StockQuoteSoapRole

Port Type: StockQuoteSoap

Replace PartnerLink1 with “Provider” and click ok.



5. Now create the following process



Name	ReceiveFromCustomer		
Partner Link	Customer		
Operation	GetQuote		
InputVariable	Name	GetQuoteIn	
	Type	http://www.pymma.com/:GetQuoteSoapIn	
	Scope	stockQuoteTrace	
Create Instance	Selected		

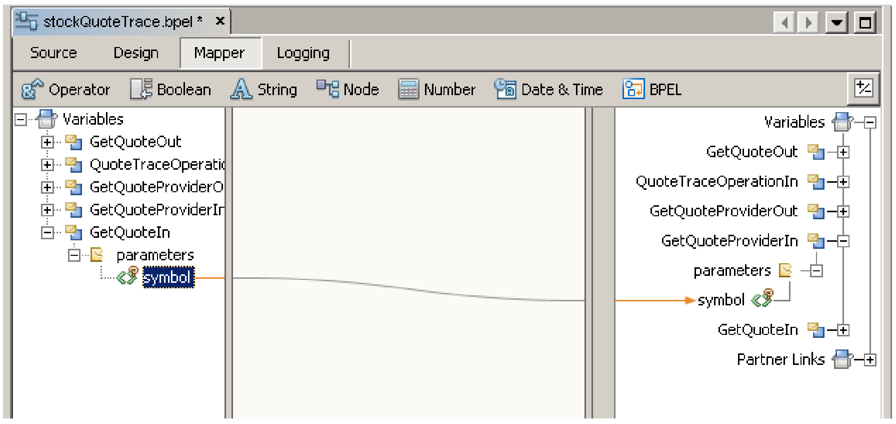
Name	InvokeTracer	
Partner Link	Provider	
Operation	GetQuote	
InputVariable	Name	GetQuoteProviderIn
	Type	http://www.webserviceX.NET/:GetQuoteSoapIn
	Scope	stockQuoteTrace
InputVariable	Name	GetQuoteProviderOut
	Type	http://www.webserviceX.NET/:GetQuoteSoapIn
	Scope	stockQuoteTrace

Name	InvokeTrace
Partner Link	Tracer
Operation	quoteTraceOperation

InputVariable	Name	QuoteTraceOperationIn
	Type	http://www.pymma.com/wsdl/quoteTrace:traceMessage
	Scope	stockQuoteTrace

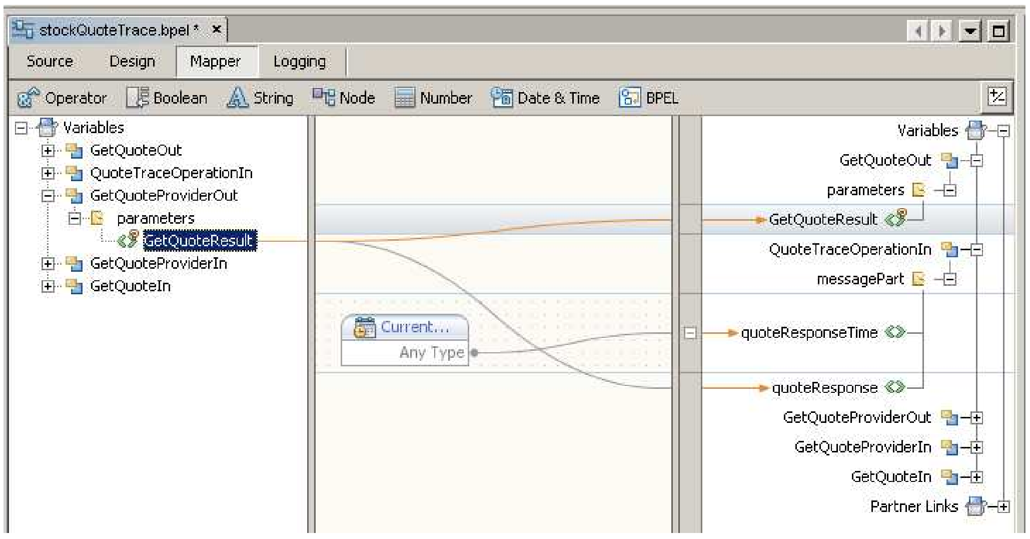
Name	ReplyToCustomer	
Partner Link	GetQuote	
Operation	quoteTraceOperation	
Normal Response	Name	GetQuoteOut
	Type	http://www.pymma.com/:GetQuoteSoapOut
	Scope	stockQuoteTrace

6. Insert an “Assign” activity between “ReceiveFromCustomer” and “invokeProvider”.
   Click on the Mapper button and map the variables as follows



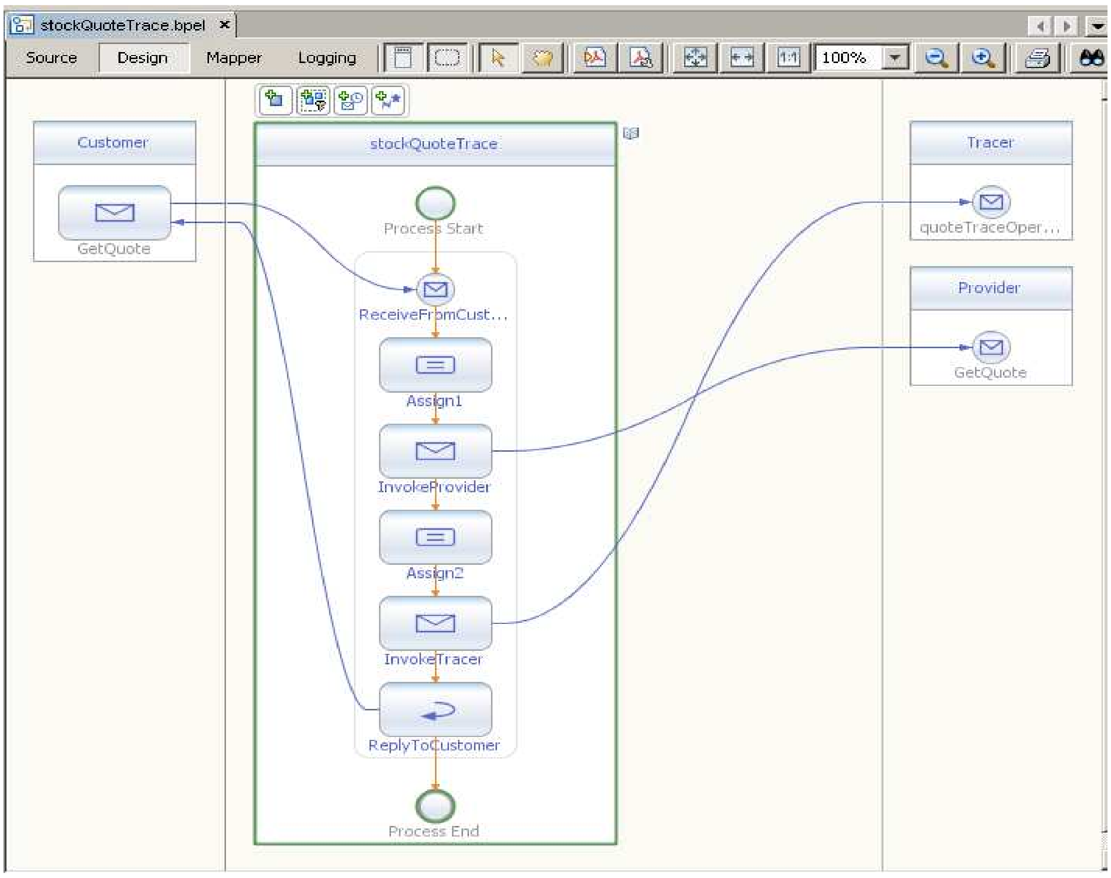
Go back on design view

7. Insert again an “Assign” activity between “invokeProvider” and “invokeTracer”.
   Click on the Mapper button and map the variables as follows



Note: QuoteResponseTime is linked to the "Current Date and Time".

8. Return to the design view. The final BPEL scenario is :

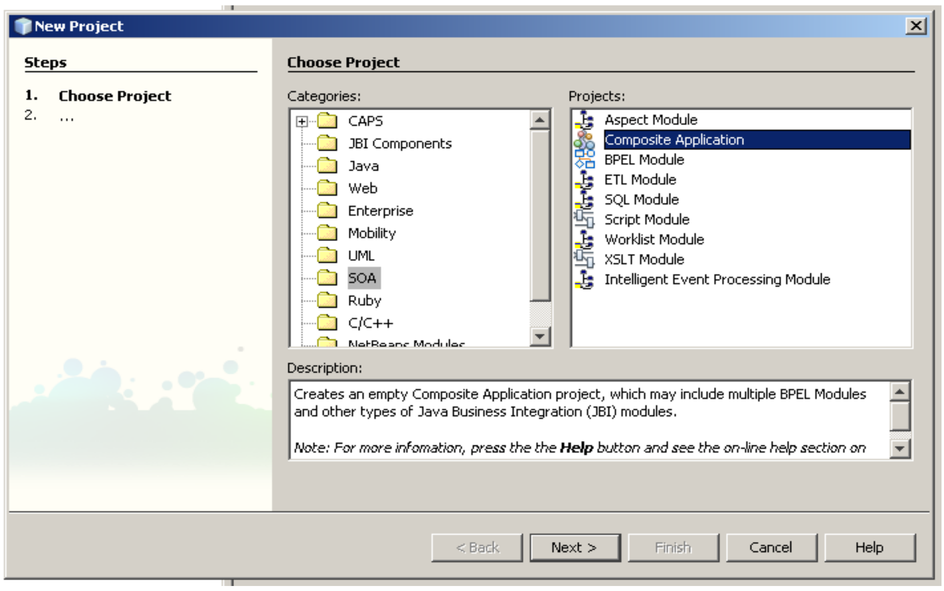


9. Go back to the design view and click on the Validate XML button.
   If an error occurs, you must fixe it before going on.

10. Save the BPEL and close it

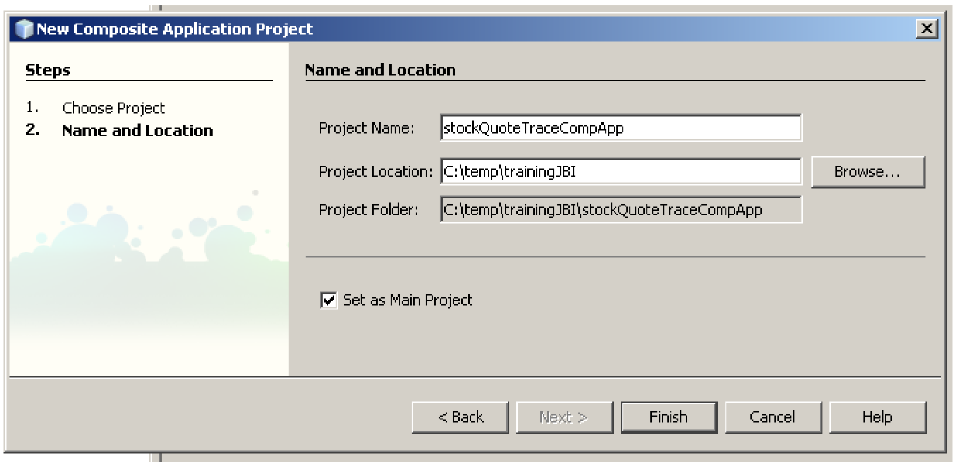
Step 5 : Create a composite application

1. Create a new composite Application Project and name it "stockQuoteTraceCompApp".



Creates an empty Composite Application project, which may include multiple BPEL Modules and other types of Java Business Integration (JBI) modules.

Note: For more infomation, press the the ***Help*** *button and see the on-line help section on*



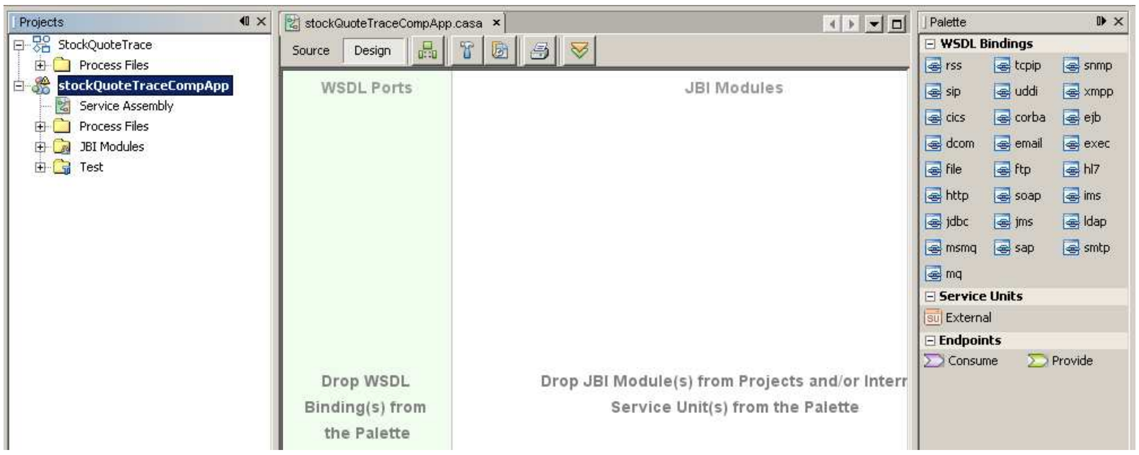
Project Name: stockQuoteTraceCompApp

Project Location: C:\temp\trainingJBI

Project Folder: C:\temp\trainingJBI\stockQuoteTraceCompApp

☑ Set as Main Project

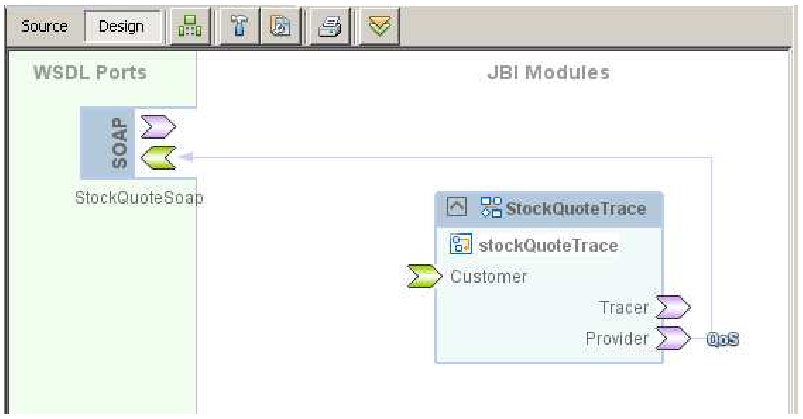
2. Click on finish. Service Assembly Editor opens.



On the left side there is the "project window" and on the right side, the "Palette window" which contains the WSDL Binding defined in your IDE. In the middle, The CASA Editor is divided in two parts. The left one is named “WSDL Ports” and the right one “JBI Module”.

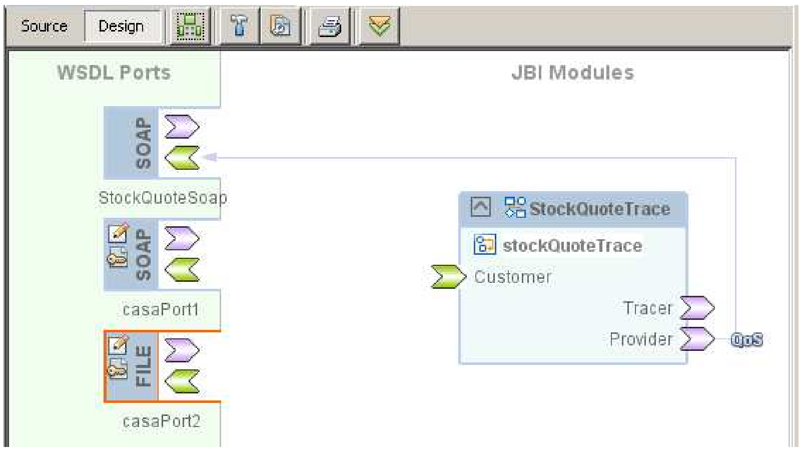
3. Click on the “StockQuoteTrace” application node in the project window. Drag it and drop it in the right part of CASA editor.

   Then click on the Build button.

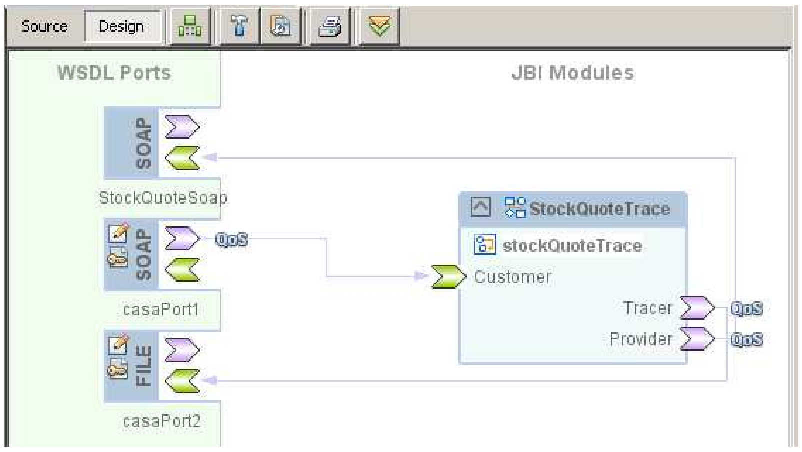


The three endpoints defined in the StockQuote application appears. Since we did not define concrete parts for “CopyStockQuote.wsdl” and “quoteTrace.WSDL”, Tracer and Customer endpoints have no WSDL Port associated.

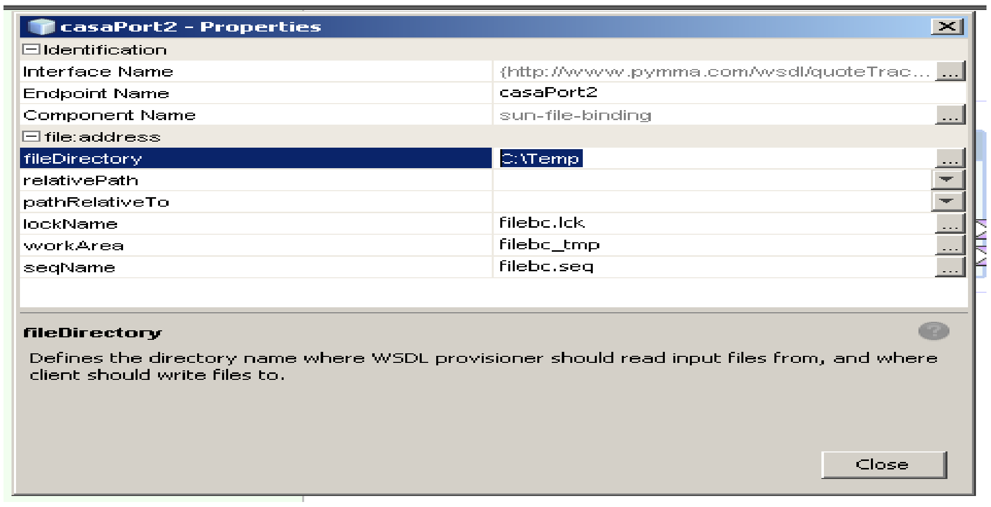
4. Add WSDL Ports dynamically
   From the palette window drag the “soap icon” and drag it in the “WSDL Ports” side of CASA Editor.
   Redo the same operation with the “file icon”.



5. Link the endpoints as follows



6. Right click on File port and select properties



casaPort2 - Properties

⊟ Identification	
Interface Name	{http://www.pymma.com/wsdl/quoteTrac…
Endpoint Name	casaPort2
Component Name	sun-file-binding
⊟ file:address	
fileDirectory	C:\Temp
relativePath	
pathRelativeTo	
lockName	filebc.lck
workArea	filebc_tmp
seqName	filebc.seq

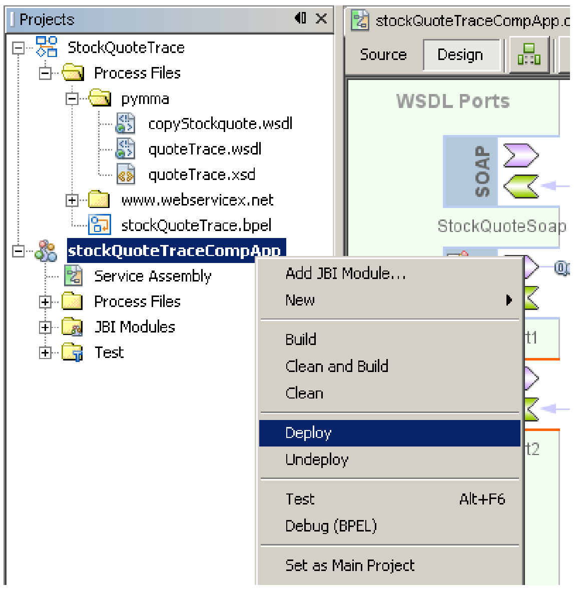
fileDirectory
Defines the directory name where WSDL provisioner should read input files from, and where client should write files to.

File Directory value indicates where Stockquote traces will be written. If you need more details on the "File binding component" and define a more sophisticated log trace, please have look on the page http://wiki.open-esb.java.net/Wiki.jsp?page=FileBC .

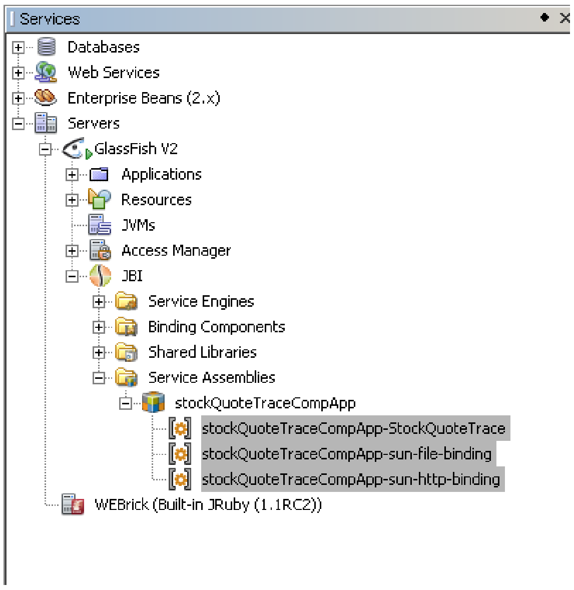
7. Close properties window and rebuild the project.

Step 6 : Deploy and Test

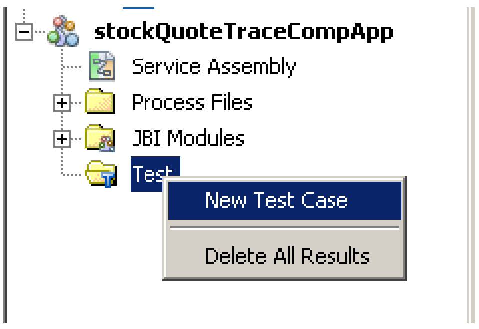
1. Deploy you application on Glassfish

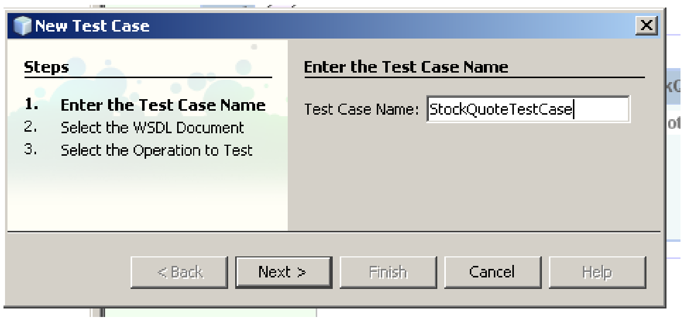


2. Open the services window. Expand the Glassfish->JBI->Service Assemblies node and check if the components you installed, have been started.



3. Test your application
Return to the project window and create a new Test in the "stockQuoteTracecompApp" project. And name it "StockQuoteTraceComApp".




New Test Case

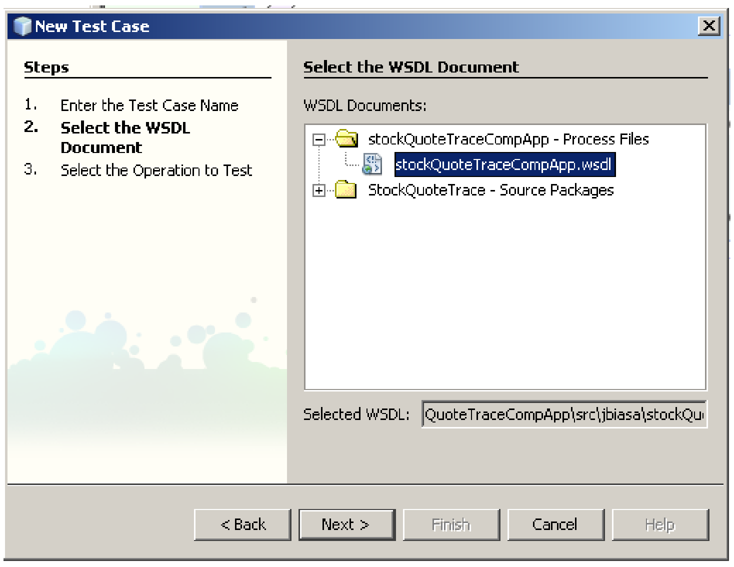
Steps

1. Enter the Test Case Name
2. Select the WSDL Document
3. Select the Operation to Test

Enter the Test Case Name

Test Case Name: StockQuoteTestCase

4. Click Next and Select the WSDL “stockQuoteTraceComApp” in the Composite application.


New Test Case

Steps

1. Enter the Test Case Name
2. Select the WSDL Document
3. Select the Operation to Test

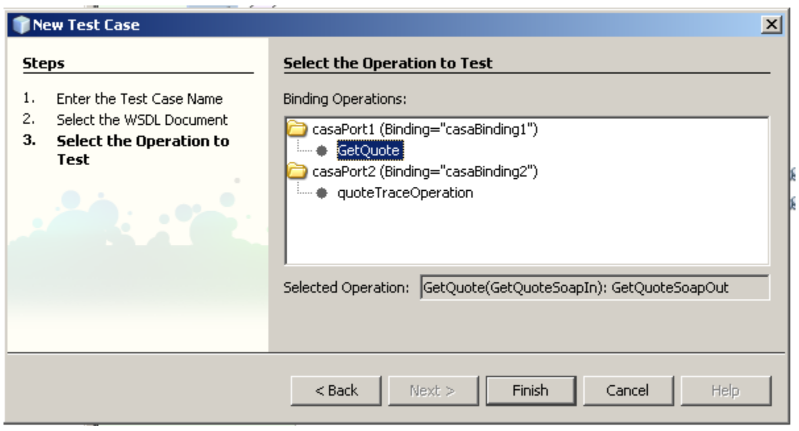
Select the WSDL Document

WSDL Documents:

stockQuoteTraceCompApp - Process Files
stockQuoteTraceCompApp.wsdl
StockQuoteTrace - Source Packages

Selected WSDL: QuoteTraceCompApp\src\jbiasa\stockQu

5. Select “GetQuote” operation
6. Click on Finish button.


New Test Case
Steps
1. Enter the Test Case Name
2. Select the WSDL Document
3. Select the Operation to Test
< Back
Next >
Finish
Cancel
Help

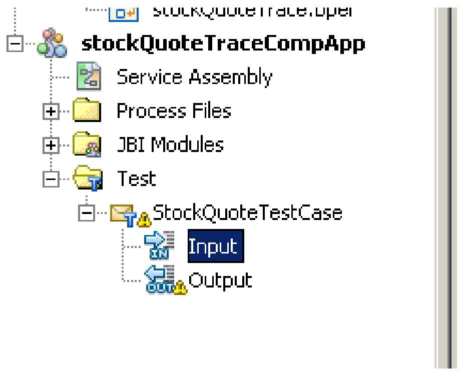
Select the Operation to Test

Binding Operations:

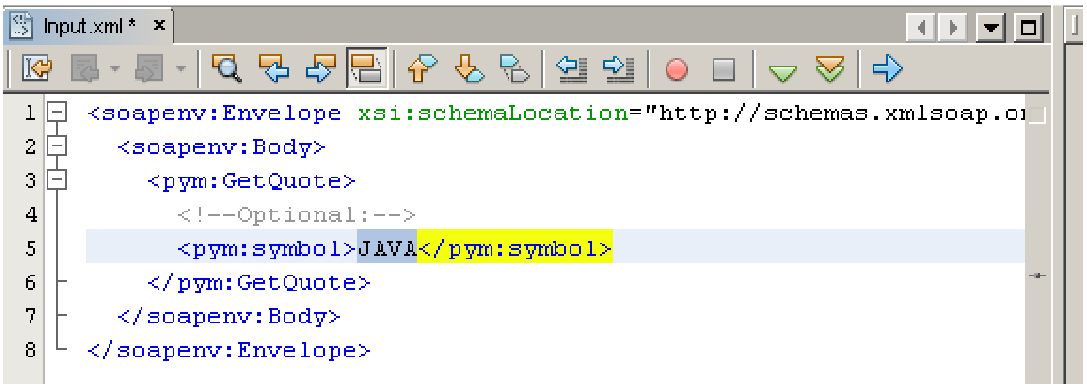
```
casaPort1 (Binding="casaBinding1")
    GetQuote
casaPort2 (Binding="casaBinding2")
    quoteTraceOperation
```

Selected Operation: GetQuote(GetQuoteSoapIn): GetQuoteSoapOut

7. Open the document "input.xml" in the test case



8. Replace pym:Symbol with "JAVA" (*Sun Microsystems Quote name*)

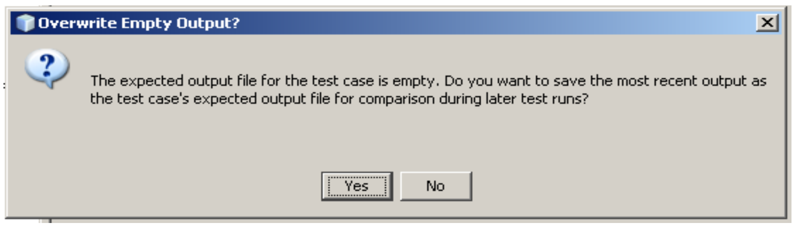


```
<soapenv:Envelope xsi:schemaLocation="http://schemas.xmlsoap.o
  <soapenv:Body>
    <pym:GetQuote>
      <!--Optional:-->
      <pym:symbol>JAVA</pym:symbol>
    </pym:GetQuote>
  </soapenv:Body>
</soapenv:Envelope>
```

9. Save and close the XML document

10. Right click on the "StockQuoteTestCase" node and select run.

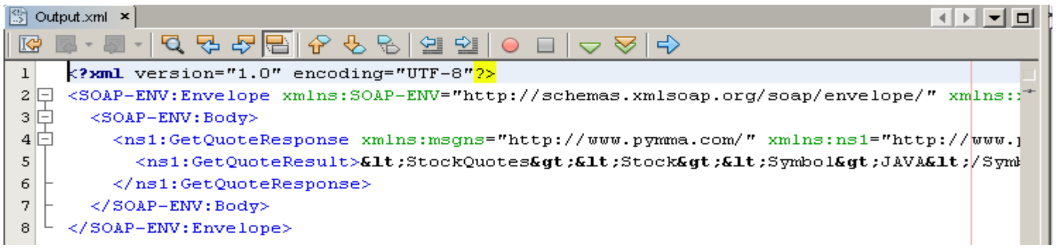
At the first test, a popup window could appear:



The expected output file for the test case is empty. Do you want to save the most recent output as the test case's expected output file for comparison during later test runs?

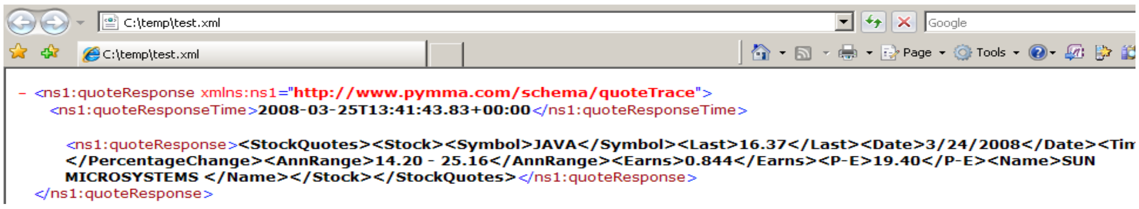
11. Reply Yes

12. Open the file “Output.xml” and read the service response



```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:
  <SOAP-ENV:Body>
    <ns1:GetQuoteResponse xmlns:msgns="http://www.pymma.com/" xmlns:ns1="http://www.
      <ns1:GetQuoteResult>&lt;StockQuotes&gt;&lt;Stock&gt;&lt;Symbol&gt;JAVA&lt;/Sym
    </ns1:GetQuoteResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

13. Open the Windows Explorer and expand the directory “c:\temp”. Open the XML file “test.xml”.



```
- <ns1:quoteResponse xmlns:ns1="http://www.pymma.com/schema/quoteTrace">
    <ns1:quoteResponseTime>2008-03-25T13:41:43.83+00:00</ns1:quoteResponseTime>

      <ns1:quoteResponse><StockQuotes><Stock><Symbol>JAVA</Symbol><Last>16.37</Last><Date>3/24/2008</Date><Tim
      </PercentageChange><AnnRange>14.20 - 25.16</AnnRange><Earns>0.844</Earns><P-E>19.40</P-E><Name>SUN
      MICROSYSTEMS </Name></Stock></StockQuotes></ns1:quoteResponse>
  </ns1:quoteResponse>
```

There are two elements in the xml file: quoteResponseTime and quoteResponse as design in the Schema.

Conclusion

Arriving at the conclusion, we hope the tutorial has been pleasant and useful. We learnt how set an Open-ESB composite application as intermediate. To do it, we used an original web service provided by an external partner, we copied and renamed it. We applied minor modifications on the namespace and redefine the concrete part. BPEL completed the job by playing an orchestrator role in the intermediation. Since we did not change the abstract part of the WSDL, we did not create disturbance for the consumer and the provider of services.

The use case in this paper is a very simple one. If you need more sophisticated intermediation for your application (ex: Services Governance), the principles expressed in this tutorial remain and you could be able to adapt them to your requirement easily.

Last remark: in the steps 2-3, we did not define any concrete parts or binding for the WSDLs. Binding have been add at the last moment (step 5). It is a flexible way to design an agile SOA application. In an upcoming tutorial, we will go deeper into this technique.

Thanks for sending your remarks and feedback at : contact@pymma.com. For further detail on our services and training on JBI and Open-ESB visit our web site www.pymma.com