OpenESB Standalone Edition V3.0
Hello World

Document identifier:

Pymma document: 770-002

Location:

www.pymma.com

Editor:

Pymma Services: contact@pymma.com

Abstract:

This document provides a short guide to create a simple “Hello world” application with OpenESB Standalone EditionV3.0. You will learn how to install shared libraries, components and deploy composite applications, connect OE Studio and OE Instance, create BPEL and a composite application project. Then you will understand how to deploy and test your project.

Status:

This Document is in a beta state

ABOUT PYMMA CONSULTING

Pymma Services is a technical architect bureau founded in 1999 and headquartered in London, United Kingdom . It provides expertise in service oriented integration systems design and implementation. Leader of OpenESB project, Pymma is recognised as one of the main actors in the integration landscape. It deeply invests in open source projects such as Drools rules engine. Pymma is a European company based in London with regional offices in France, Belgium and Canada. (contact@pymma.com or visit our website on www.pymma.com)

Contents

1 Introduction

When we start learning about a product, it is customary to start with a ‘hello word exercise’. Through this simple exercise easily understood by anyone, we develop a first application, and get our first thrill and enjoyment by getting Hello + Name.
We go a bit further in this tutorial by explaining how to install OpenESB shared libraries and components and how to connect OpenESB Studio with an OpenESB instance. At the end of this tutorial you will be able to deploy and test your application without leaving OE-Studio.

OpenESB Standalone Edition (OE SE) offers the lightest and most efficient service oriented integration tools on the market. Powerful, prompt, scalable with a very low memory footprint, OE SE is perfect for virtualisation and cloud deployment.

1.1 Before starting

We suppose you have already installed OpenESB on your machine. If not have a look at our document: **770-001-OE SE Installation Guide**.

1.2 What you will learn in this document

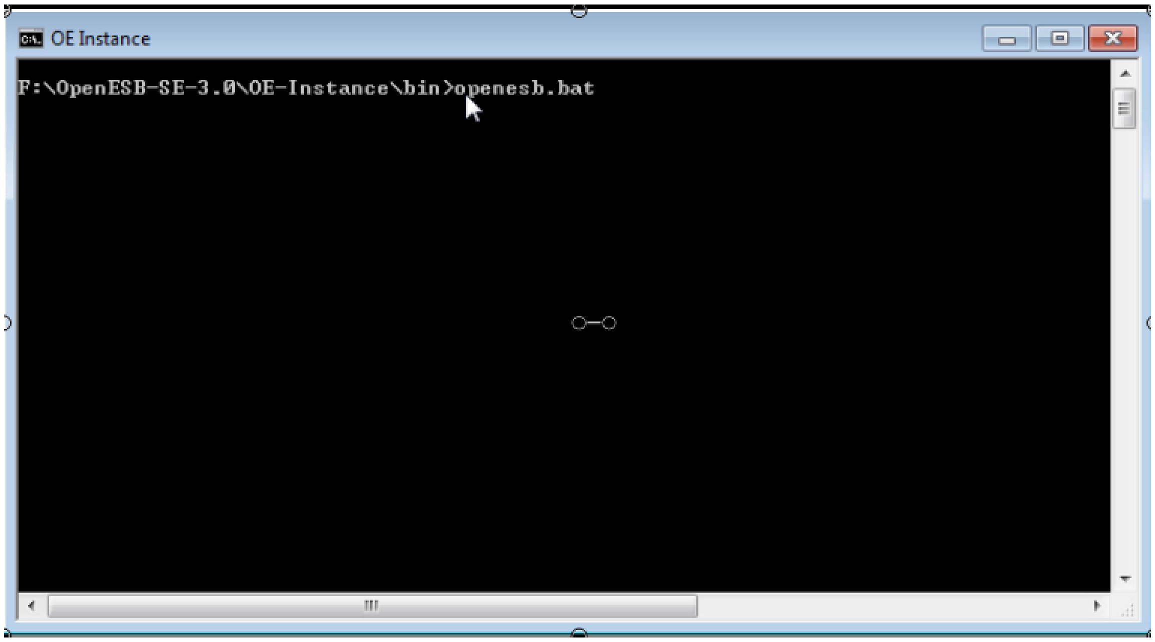
This tutorial agenda is:

- Install OE shared libraries and components
- Connect OE-Studio with an OE-Instance
- Create a BPEL project
- Create a composite application project
- Deploy a project
- Test a project

As much as possible we try to be clear and concise in our tutorial. Feel free to contact us to give us your feedback (contact@pymma.com).

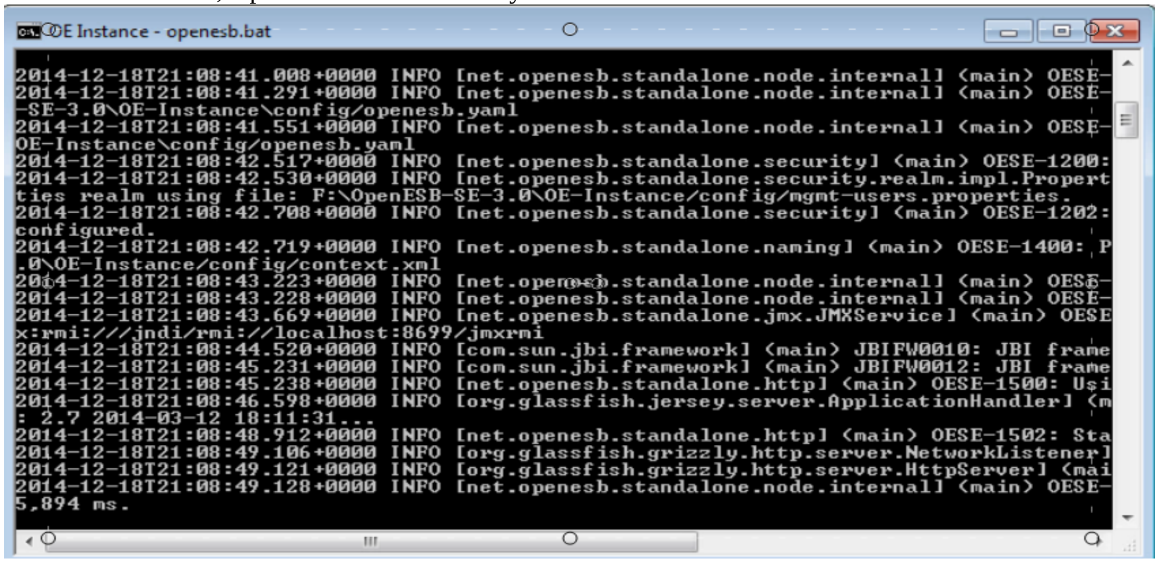
2 Start OpenESB instance

First Open a console and go to the directory **…\OpenESB-SE-3.0\OE-Instance\bin** and run "oeadmin.bat" on Windows and "oeadmin.sh" on Linux or UNIX.



```
F:\OpenESB-SE-3.0\OE-Instance\bin>openesb.bat
```

After a few seconds, OpenESB Instance is ready to run.

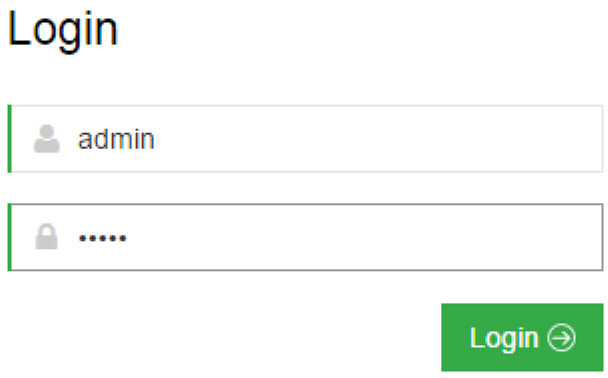


```
2014-12-18T21:08:41.008+0000 INFO [net.openesb.standalone.node.internal] <main> OESE-
2014-12-18T21:08:41.291+0000 INFO [net.openesb.standalone.node.internal] <main> OESE-
-SE-3.0\OE-Instance\config/openesb.yaml
2014-12-18T21:08:41.551+0000 INFO [net.openesb.standalone.node.internal] <main> OESE-
OE-Instance\config/openesb.yaml
2014-12-18T21:08:42.517+0000 INFO [net.openesb.standalone.security] <main> OESE-1200:
2014-12-18T21:08:42.530+0000 INFO [net.openesb.standalone.security.realm.impl.Propert
ties realm using file: F:\OpenESB-SE-3.0\OE-Instance/config/mgmt-users.properties.
2014-12-18T21:08:42.708+0000 INFO [net.openesb.standalone.security] <main> OESE-1202:
configured.
2014-12-18T21:08:42.719+0000 INFO [net.openesb.standalone.naming] <main> OESE-1400: P
.0\OE-Instance/config/context.xml
2014-12-18T21:08:43.223+0000 INFO [net.openesb.standalone.node.internal] <main> OESE-
2014-12-18T21:08:43.228+0000 INFO [net.openesb.standalone.node.internal] <main> OESE-
2014-12-18T21:08:43.669+0000 INFO [net.openesb.standalone.jmx.JMXService] <main> OESE
x:rmi:///jndi/rmi://localhost:8699/jmxrmi
2014-12-18T21:08:44.520+0000 INFO [com.sun.jbi.framework] <main> JBIFW0010: JBI frame
2014-12-18T21:08:45.231+0000 INFO [com.sun.jbi.framework] <main> JBIFW0012: JBI frame
2014-12-18T21:08:45.238+0000 INFO [net.openesb.standalone.http] <main> OESE-1500: Usi
2014-12-18T21:08:46.598+0000 INFO [org.glassfish.jersey.server.ApplicationHandler] <m
: 2.7 2014-03-12 18:11:31...
2014-12-18T21:08:48.912+0000 INFO [net.openesb.standalone.http] <main> OESE-1502: Sta
2014-12-18T21:08:49.106+0000 INFO [org.glassfish.grizzly.http.server.NetworkListener]
2014-12-18T21:08:49.121+0000 INFO [org.glassfish.grizzly.http.server.HttpServer] <mai
2014-12-18T21:08:49.128+0000 INFO [net.openesb.standalone.node.internal] <main> OESE-
5,894 ms.
```

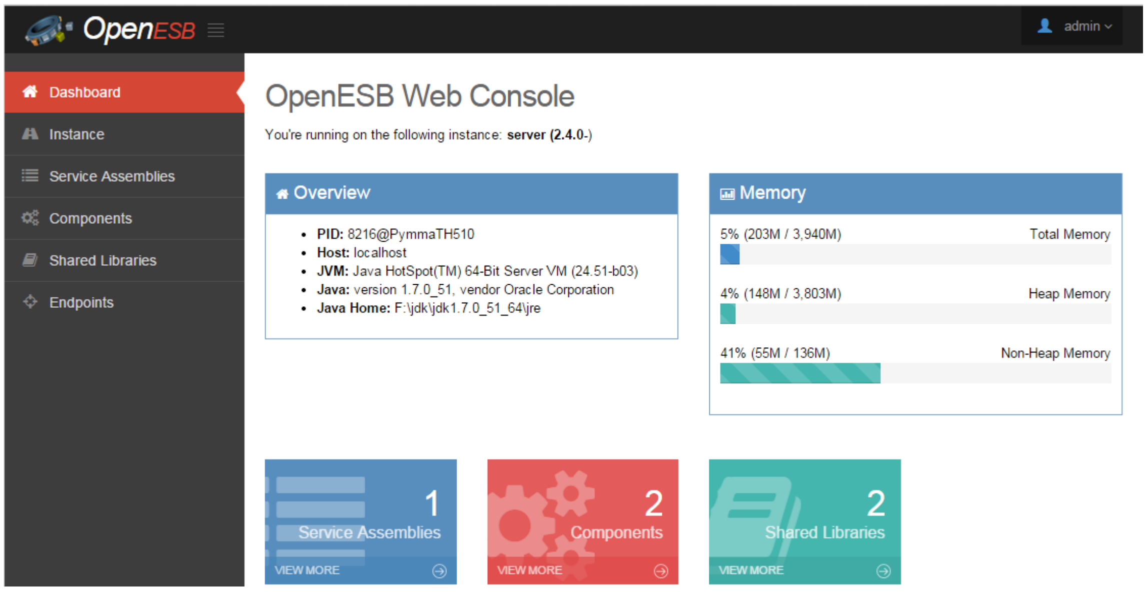
Open a browser (Firefox, Chrome or Else) and type the address: http://localhost:4848/webui.

OpenESB web admin console allows you to install shared libraries and components and deploy your project. It provides monitoring facilities as well.

Before starting you have to enter a login and password. Default values are “admin” and “admin”.



Then the web admin console opens.



You're running on the following instance: **server (2.4.0-)**

Overview

- **PID:** 8216@PymmaTH510
- **Host:** localhost
- **JVM:** Java HotSpot(TM) 64-Bit Server VM (24.51-b03)
- **Java:** version 1.7.0_51, vendor Oracle Corporation
- **Java Home:** F:\jdk\jdk1.7.0_51_64\jre

Memory

5% (203M / 3,940M) Total Memory
4% (148M / 3,803M) Heap Memory
41% (55M / 136M) Non-Heap Memory

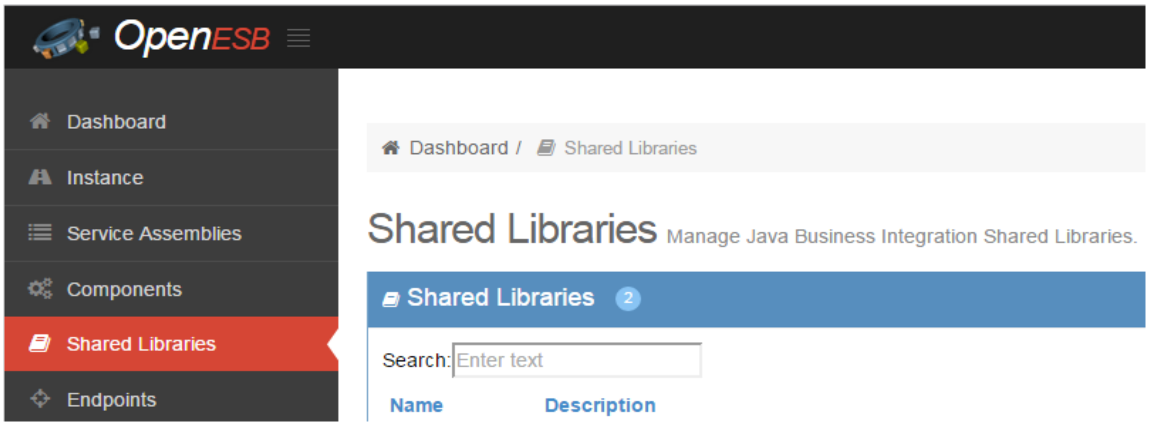
OpenESB is now ready to run.

3 Install OE shared libraries and component

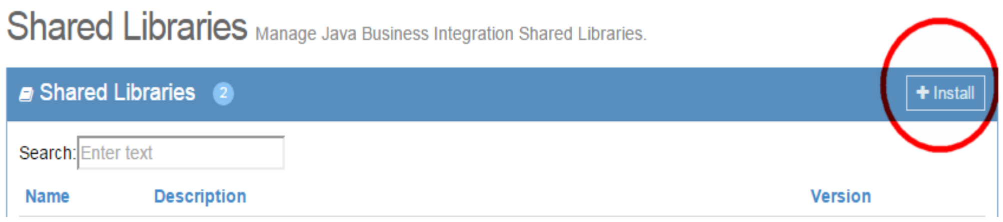
OpenESB standalone installation comes without components; it is up to you to install the shared libraries and components to run your applications. This installation is much lighter and takes up less memory.
Shared libraries are jar files shared by components in OpenESB. There are three shared libraries in OpenESB: **wsdlsl.jar**, **wsdlextlib.jar** and **encoderlib.jar**. The purpose of each library is beyond the scope of this tutorial.

wsdlsl.jar is already installed by default with an OE Instance. So you don't have to install it, let's install the two other ones.

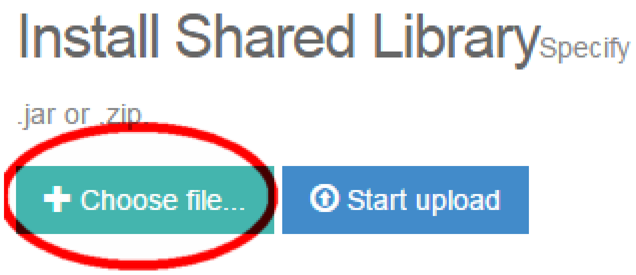
Select "**Shared Libraries**" in the main menu.



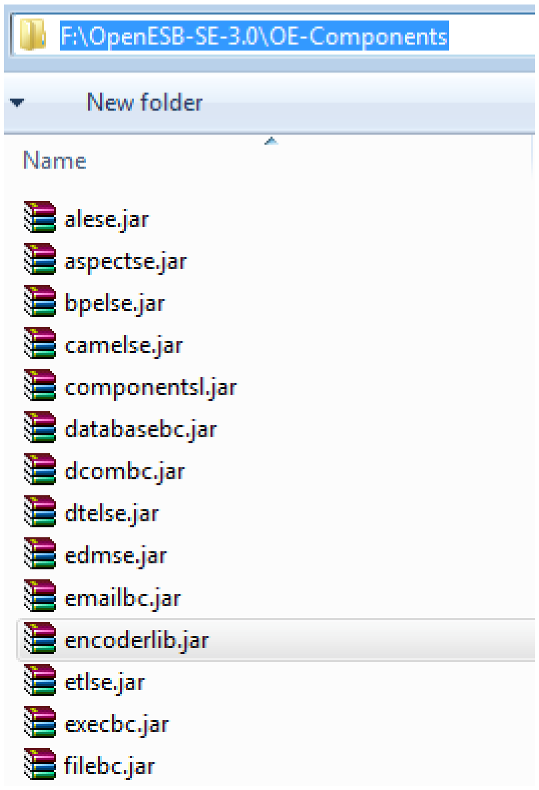
Click on the Install button on the left of the console.


Shared Libraries Manage Java Business Integration Shared Libraries.
Shared Libraries 2
+ Install
Search: Enter text
Name
Description
Version

Click on the button "**Choose file**"



Where are the components located? Shared libraries and components are located in …/OpenESB-SE-3.0\components.
Go to this directory and select the file "**encoderlib.jar**".



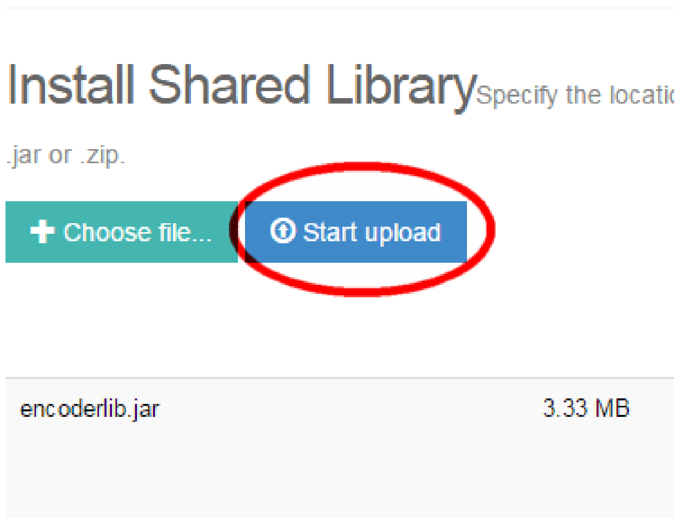
F:\OpenESB-SE-3.0\OE-Components

New folder

Name

alese.jar
aspectse.jar
bpelse.jar
camelse.jar
componentsl.jar
databasebc.jar
dcombc.jar
dtelse.jar
edmse.jar
emailbc.jar
encoderlib.jar
etlse.jar
execbc.jar
filebc.jar

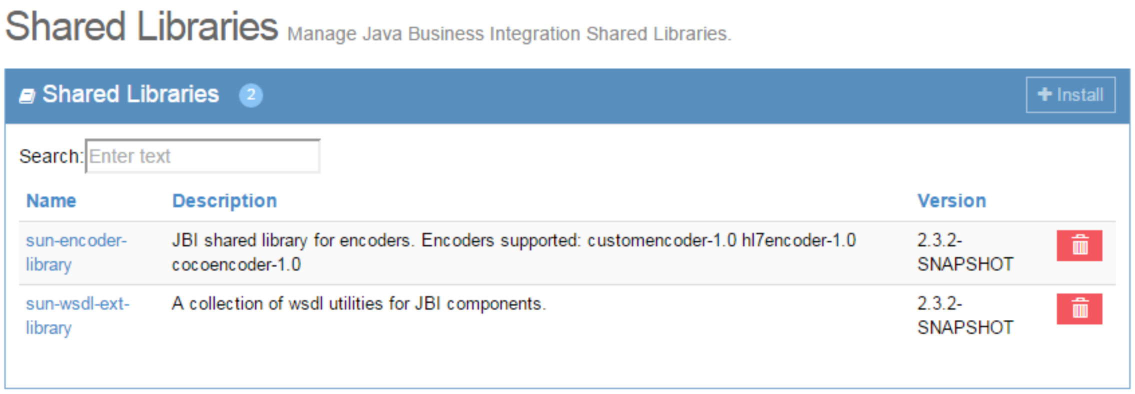
Then select the button "**Start Upload".**



encoderlib.jar 3.33 MB

After a few seconds the library opens.

Redo the same process with **wsdlextlib.jar.** You must get almost the same screen.


Shared Libraries Manage Java Business Integration Shared Libraries.
Shared Libraries 2
Install
Search: Enter text

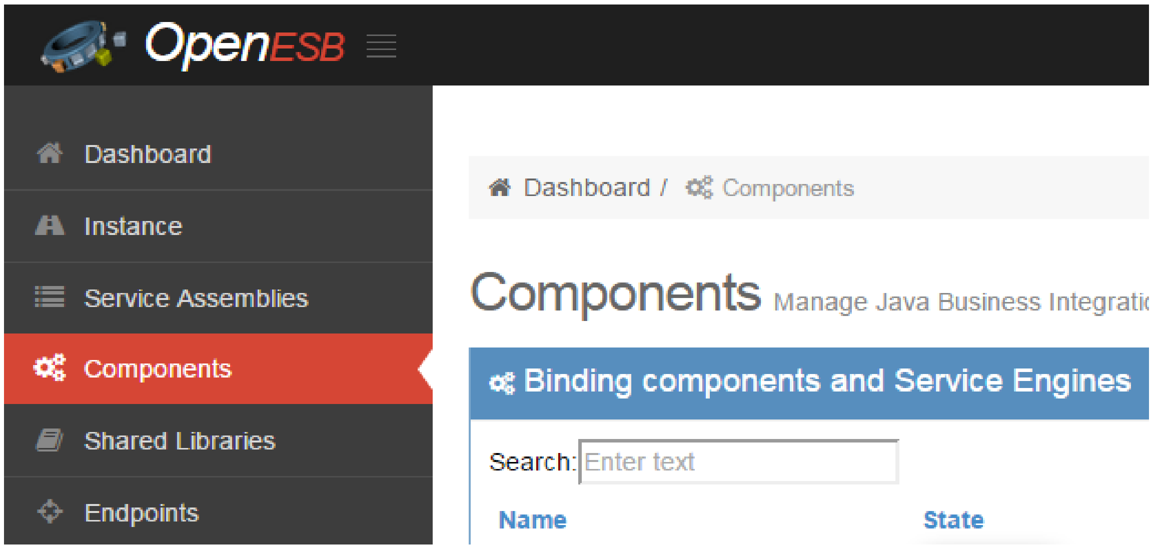
Name	Description	Version
sun-encoder-library	JBI shared library for encoders. Encoders supported: customencoder-1.0 hl7encoder-1.0 cocoencoder-1.0	2.3.2-SNAPSHOT
sun-wsdl-ext-library	A collection of wsdl utilities for JBI components.	2.3.2-SNAPSHOT

Once the shared libraries are installed we can install the components.

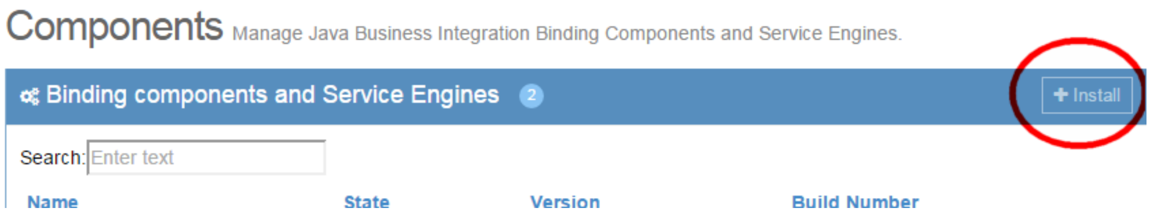
4 Install the components

Shared libraries and components installations are similar. Nevertheless, component installation requires an additional step. By default, an OE component is installed in the state shutdown; to run it must be set up in the state Start. For more information on component life cycles please read JBI specifications. For this tutorial, we will install the HTTP and BPEL components:

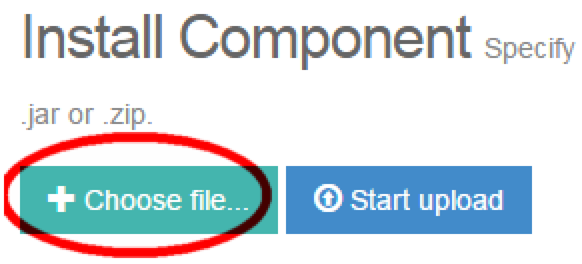
On the main menu, click on Components.



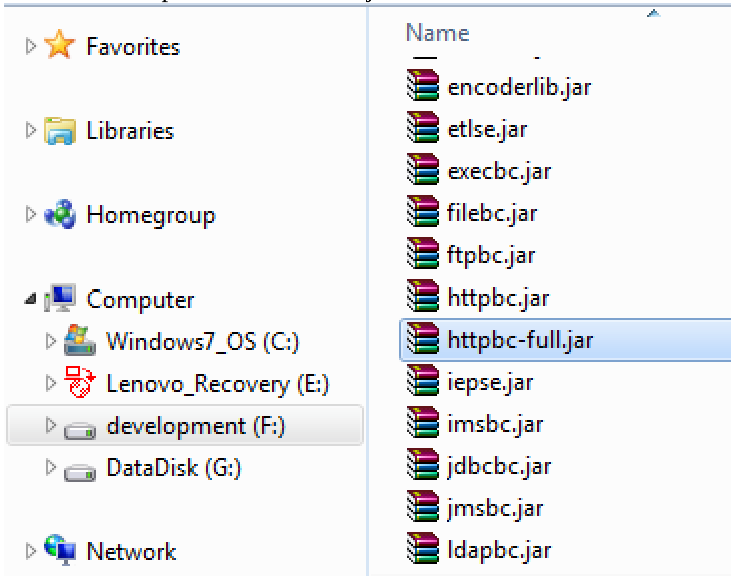
Click on the Install button on the left of the console.


Components Manage Java Business Integration Binding Components and Service Engines.
Binding components and Service Engines 2
+ Install
Search: Enter text
Name
State
Version
Build Number

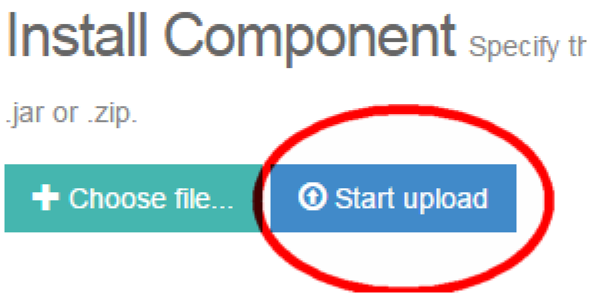
Click on the button "**Choose file**"



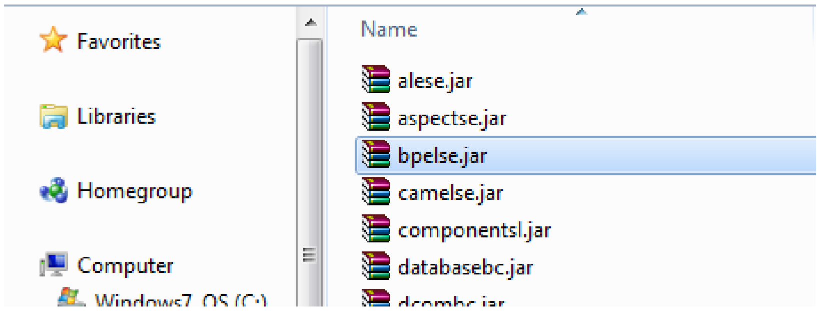
Select the component HTTP-Full.jar.



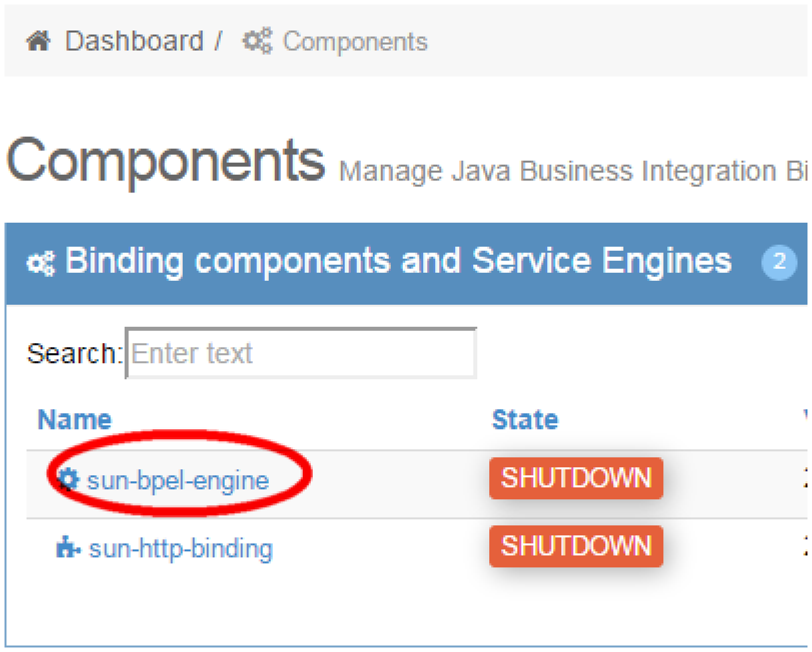
Http jar is a legacy component and should not be used with the OpenESB standalone edition. Then install the component.



Follow the same process to install the component bpelse.jar.

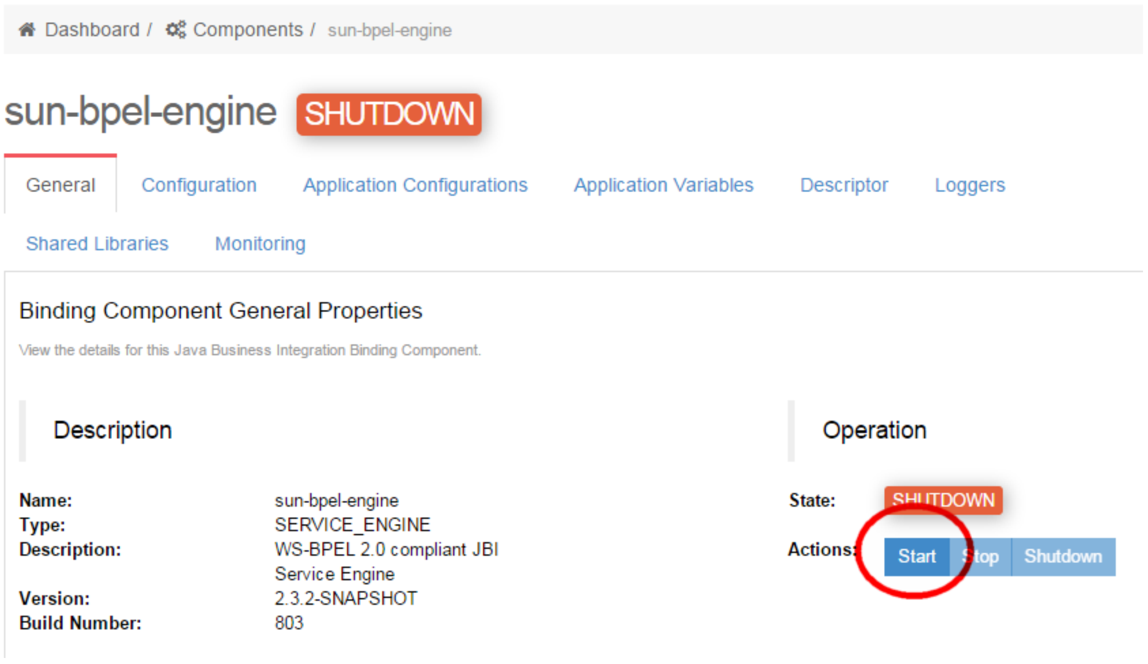


When a component is uploaded its state is set up to "**Shutdown"**. A component must be in the start state to process new message.


Dashboard / Components
Components Manage Java Business Integration Bi
Binding components and Service Engines 2
Search: Enter text

Name	State
sun-bpel-engine	SHUTDOWN
sun-http-binding	SHUTDOWN

Click on the component name, select the general tab and select "**Start**"



Dashboard / Components / sun-bpel-engine

sun-bpel-engine SHUTDOWN

General | Configuration | Application Configurations | Application Variables | Descriptor | Loggers | Shared Libraries | Monitoring

Binding Component General Properties

View the details for this Java Business Integration Binding Component.

Description

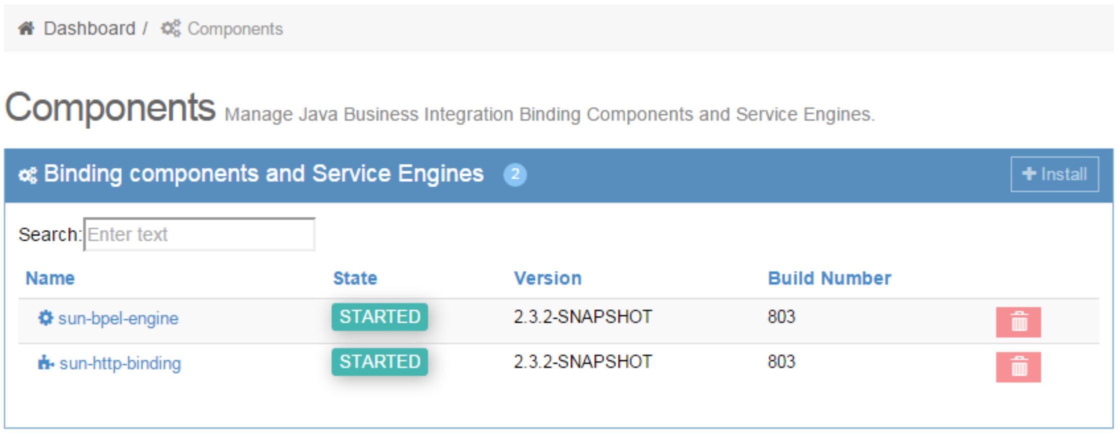
Name:	sun-bpel-engine
Type:	SERVICE_ENGINE
Description:	WS-BPEL 2.0 compliant JBI Service Engine
Version:	2.3.2-SNAPSHOT
Build Number:	803

Operation

State: SHUTDOWN

Actions: Start Stop Shutdown

After a few seconds, the component starts. Start BPEL and HTTP components to continue the tutorial.



Dashboard / Components

Components Manage Java Business Integration Binding Components and Service Engines.

Binding components and Service Engines 2 Install

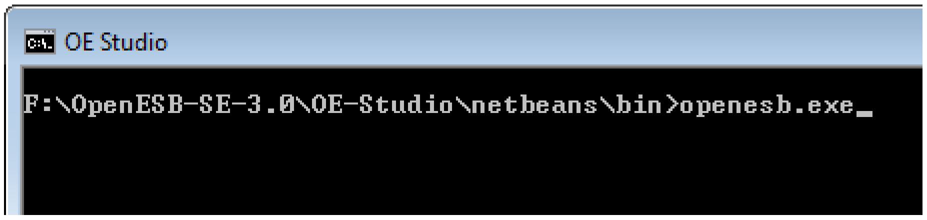
Search: Enter text

Name	State	Version	Build Number
sun-bpel-engine	STARTED	2.3.2-SNAPSHOT	803
sun-http-binding	STARTED	2.3.2-SNAPSHOT	803

Shared libraries and components are installed. You are able now to develop and deploy your application.

5 Connect OE-Studio with an OE-Instance

In this chapter, we will connect OpenESB studio and the OpenESB instance we started in the previous chapter. First open a new console and go to the repository …/OpenESB-SE-3.0/OE-Studio/NetBeans/bin. Run openesb.exe for Windows or openesb for Linux.

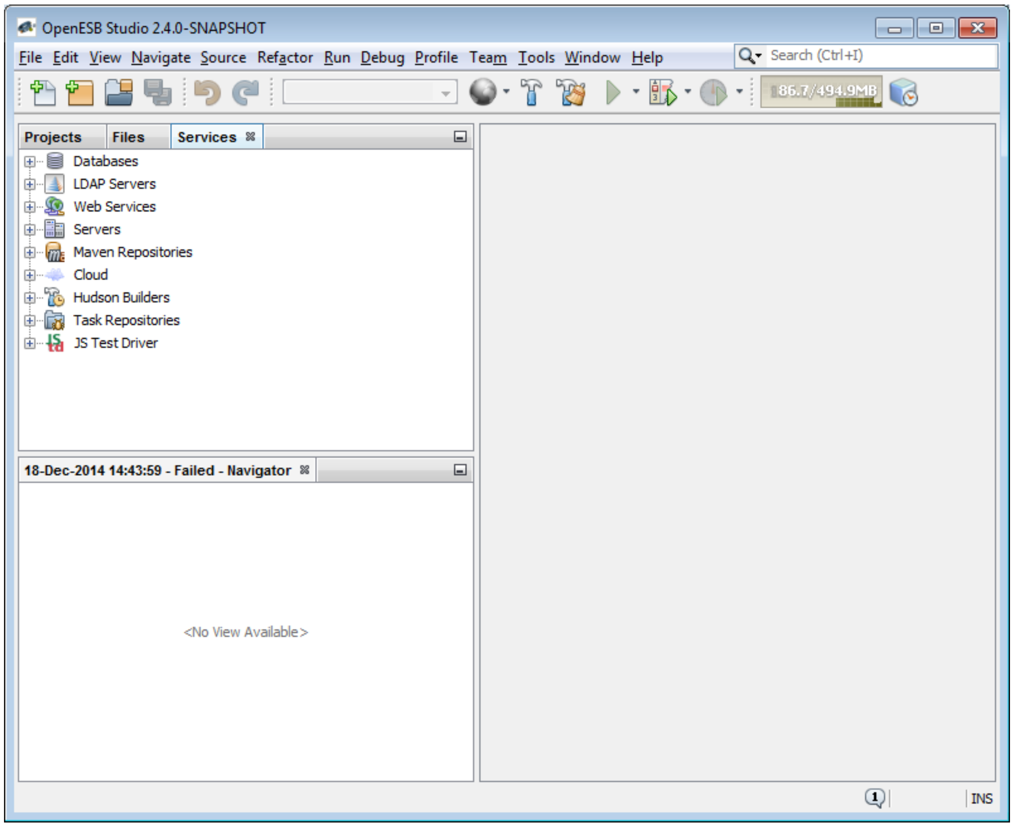


```
F:\OpenESB-SE-3.0\OE-Studio\netbeans\bin>openesb.exe_
```

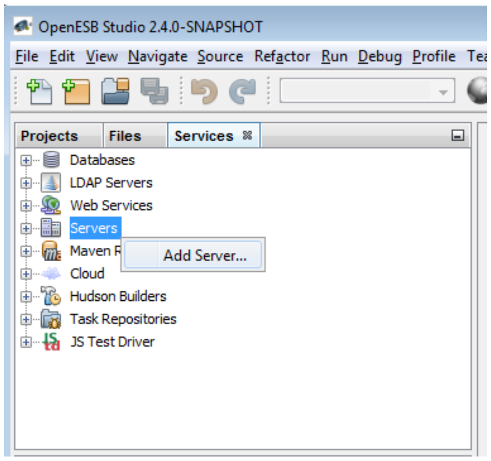
The OpenESB studio logo appears.

OpenESB
3.0
OpenESB IDE is based on Netbeans IDE and Netbeans Platform under the Common Development and Distribution License (CDDL). For more information, please visit www.open-esb.net

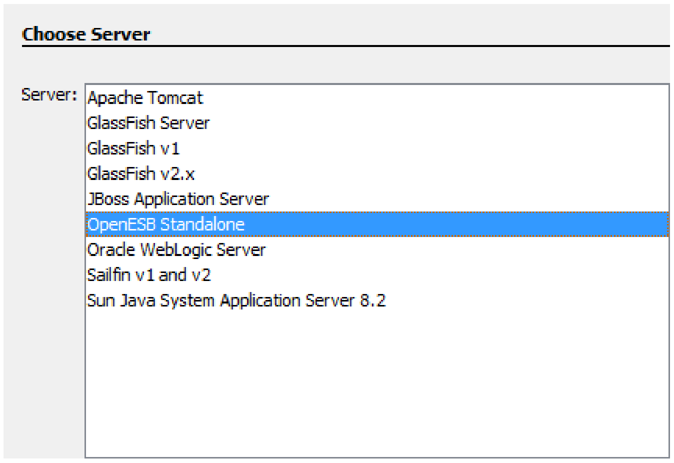
After a few moments the OpenESB studio opens.



Select the node Server, right click and select Add Server.



Select the server OpenESB standalone.

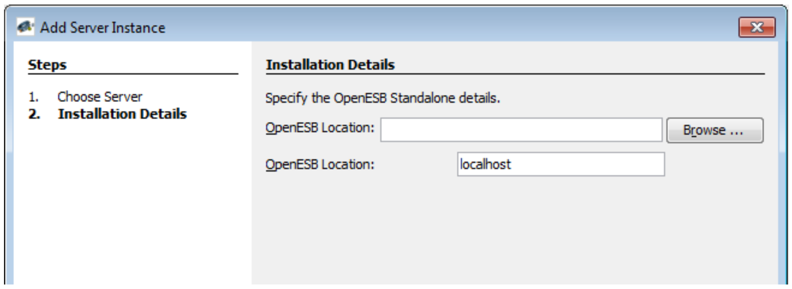


Choose Server

Server:

- Apache Tomcat
- GlassFish Server
- GlassFish v1
- GlassFish v2.x
- JBoss Application Server
- OpenESB Standalone
- Oracle WebLogic Server
- Sailfin v1 and v2
- Sun Java System Application Server 8.2

Then click on the NEXT button.



Add Server Instance

Steps

1. Choose Server
2. **Installation Details**

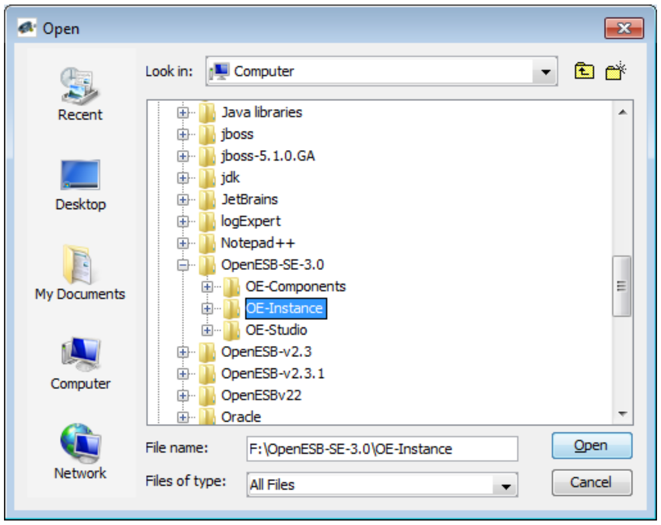
Installation Details

Specify the OpenESB Standalone details.

OpenESB Location: [Browse ...]

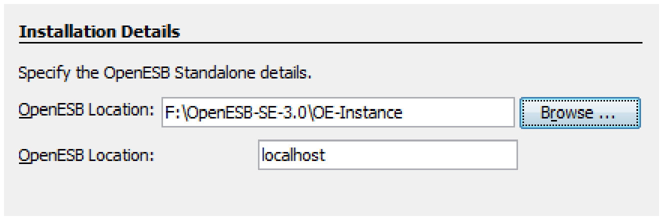
OpenESB Location: localhost

In the OpenESB Studio V3.0, the Installation screen has two fields both entitled, "OpenESB Location". The first one is used for local installations. The second one is dedicated to remote installations planned in the next OE-Studio versions.
OpenESB location is **…/OpenESB-SE-3.0/OE-Instance**. Leave the second field as it is. It should be **localhost**.



File name: F:\OpenESB-SE-3.0\OE-Instance

Files of type: All Files



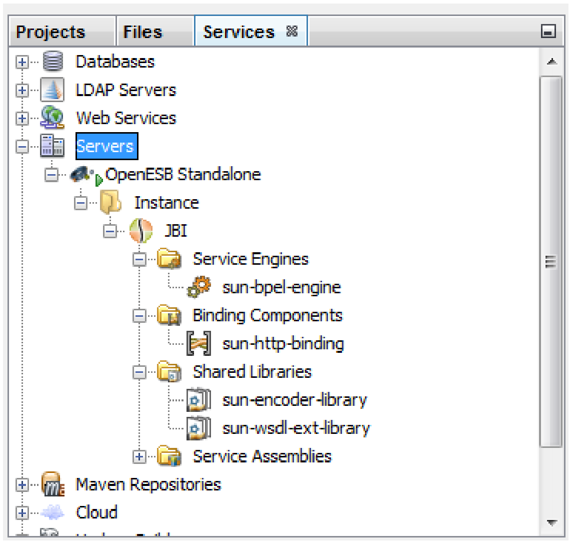
Installation Details

Specify the OpenESB Standalone details.

OpenESB Location: F:\OpenESB-SE-3.0\OE-Instance

OpenESB Location: localhost

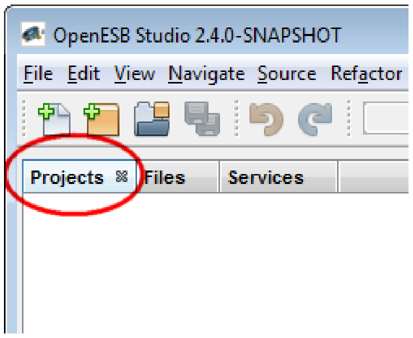
Click on the Finish button.



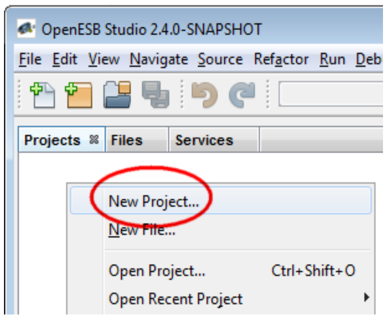
Expend the node “Servers”. You could notice that the OpenESB Studio is well connected with our OE Instance.

6 Create a BPEL project

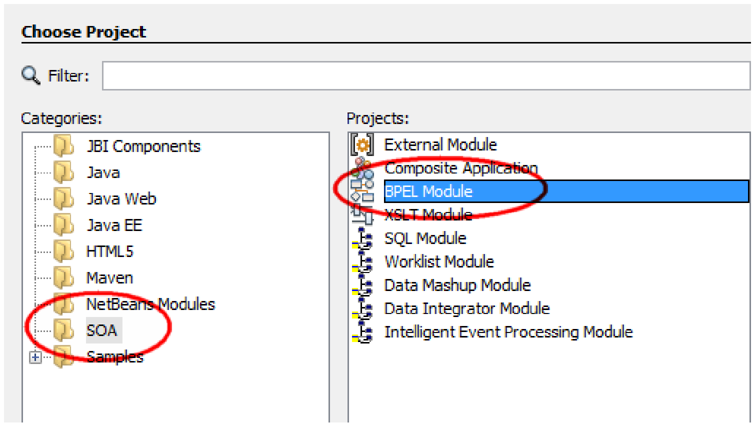
In this paragraph, we will create the “Hello World” Services. To do it we have to open the BPEL editor. Select the tab project.


OpenESB Studio 2.4.0-SNAPSHOT
File Edit View Navigate Source Refactor
Projects
Files
Services

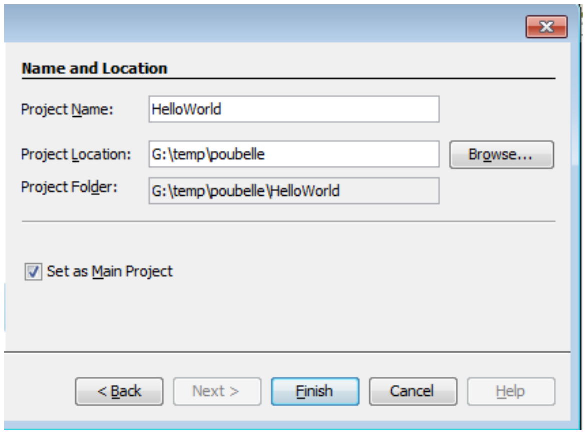
Right click on the project windows and select New Project.


OpenESB Studio 2.4.0-SNAPSHOT
File Edit View Navigate Source Refactor Run Deb
Projects
Files
Services
New Project...
New File...
Open Project... Ctrl+Shift+O
Open Recent Project

Chose the category SOA and then choose the BPEL Module.



Click on Next.


Name and Location

Project Name: HelloWorld

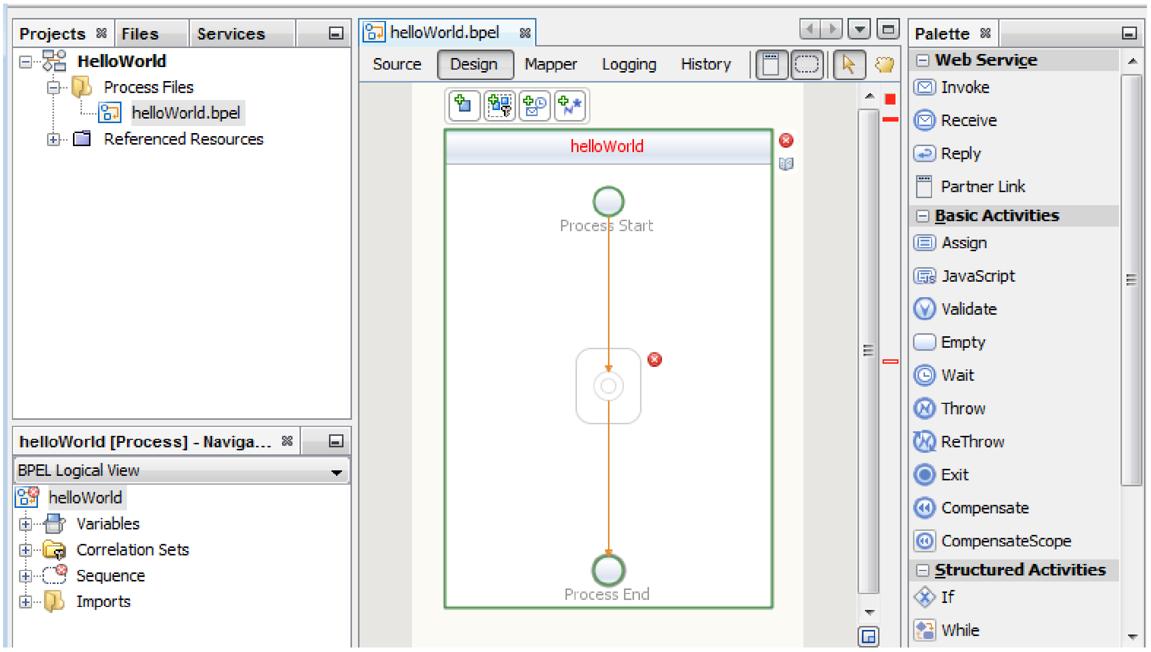
Project Location: G:\temp\poubelle

Project Folder: G:\temp\poubelle\HelloWorld

[✓] Set as Main Project

Enter the project Name and then click on Finish.

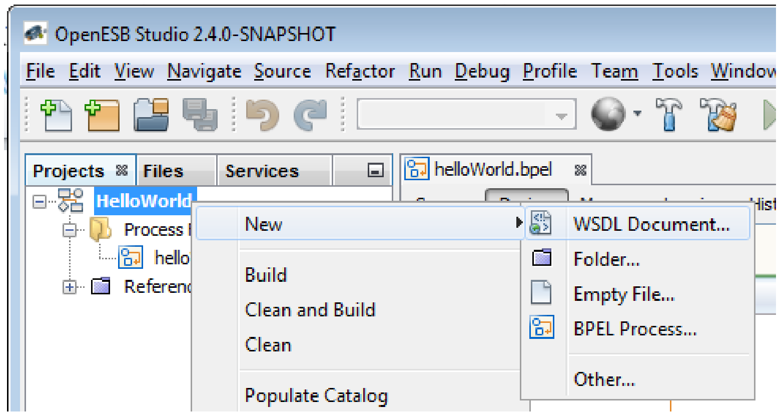
After a while, the OE-Studio screen appears as follows.



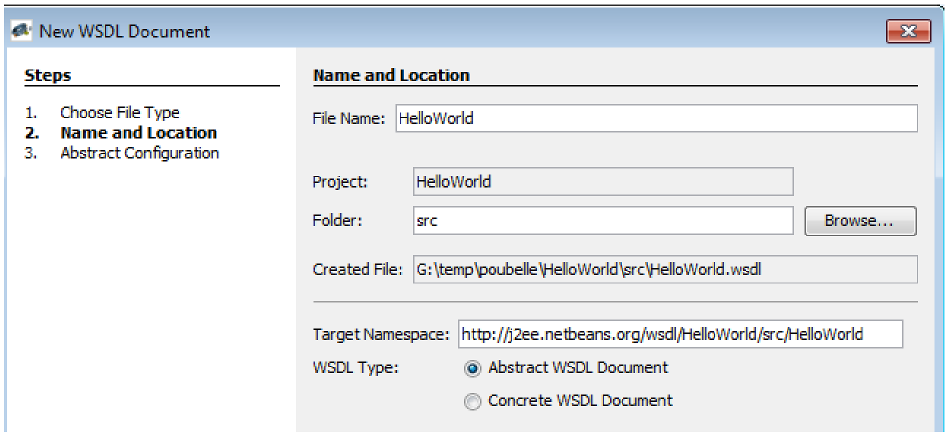
As part of the OpenESB process, before working on the orchestration, we have to define the contract of our hello world service (a WSDL). The contract defines the operation(s) and the parameter of our services. For Hello world service, the input will be a String (ex: Pymma) and the output a String as well (ex: Hello Pymma).

6.1 Create a WSDL

Right click on the project node HelloWorld and select New→ WSDL.



The WSDL wizard opens.



File Name: HelloWorld

Project: HelloWorld

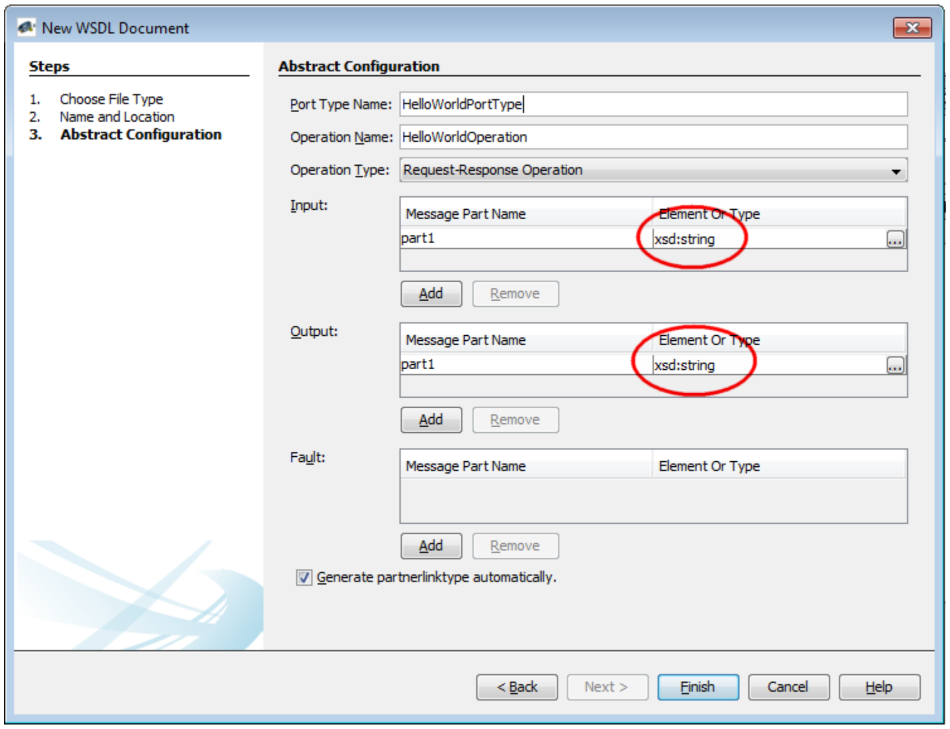
Folder: src

Created File: G:\temp\poubelle\HelloWorld\src\HelloWorld.wsdl

Target Namespace: http://j2ee.netbeans.org/wsdl/HelloWorld/src/HelloWorld

WSDL Type: [✓ Abstract WSDL Document] Concrete WSDL Document

Type HelloWorld as File Name and let the WSDL Type as Abstract. Then click on **NEXT.**
Note that Input and output element are Strings by default. Keep the default field values and click on the **Finish button**.


New WSDL Document

Steps

1. Choose File Type
2. Name and Location
3. **Abstract Configuration**

Abstract Configuration

Port Type Name: HelloWorldPortType
Operation Name: HelloWorldOperation
Operation Type: Request-Response Operation

Input:

Message Part Name	Element Or Type
part1	xsd:string

Output:

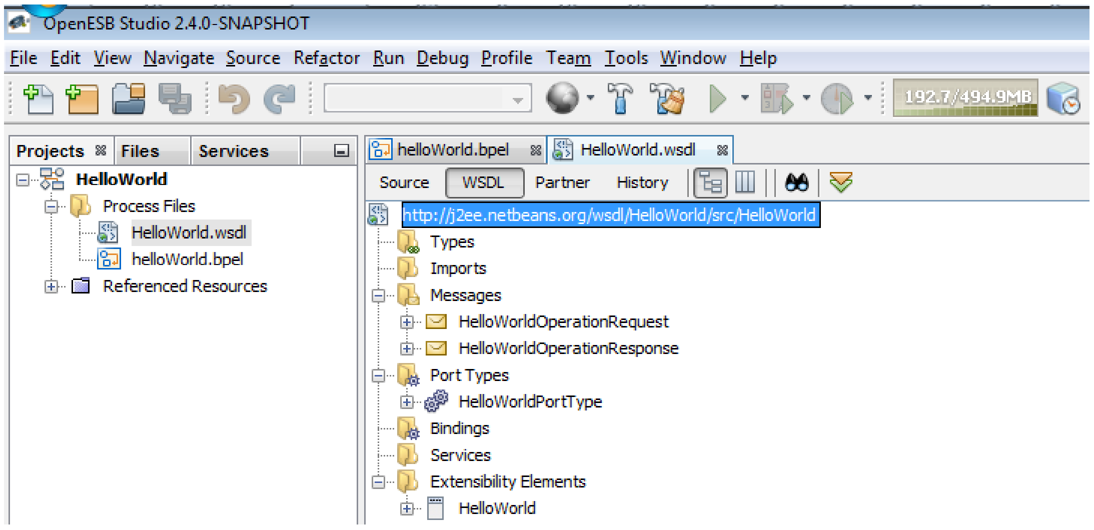
Message Part Name	Element Or Type
part1	xsd:string

Fault:

Message Part Name	Element Or Type

[✓] Generate partnerlinktype automatically.

The HelloWorld wsdl file has been added to your project.

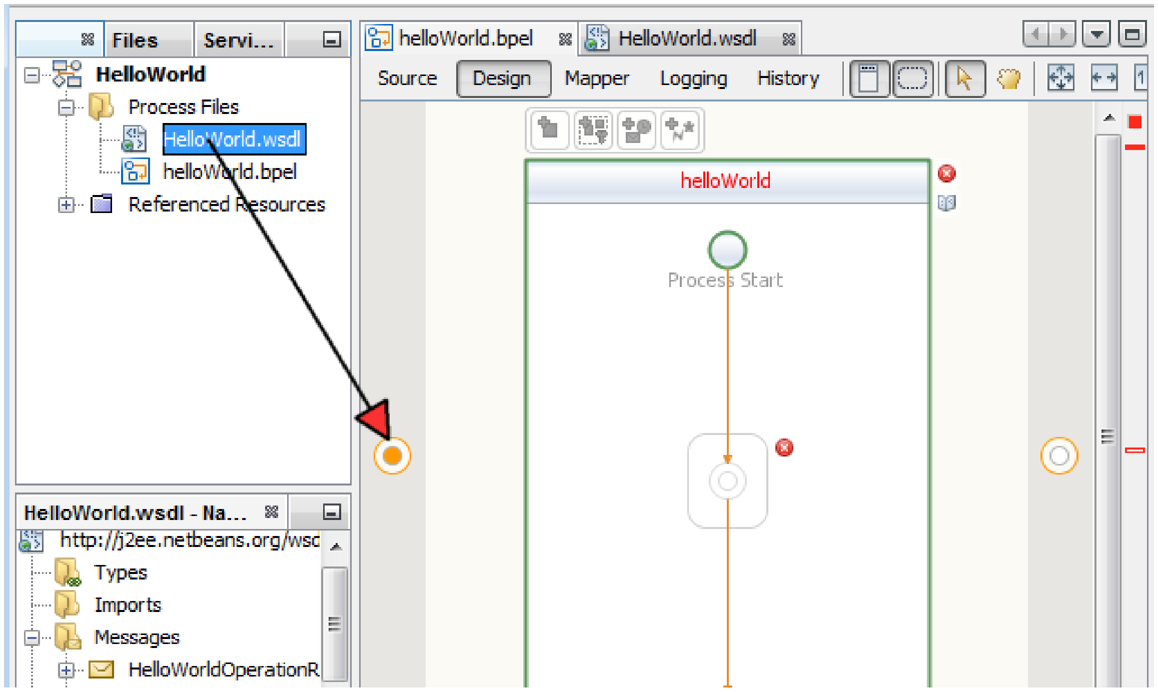

OpenESB Studio 2.4.0-SNAPSHOT
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
192.7/494.9MB
Projects
Files
Services
helloWorld.bpel
HelloWorld.wsdl
Source
WSDL
Partner
History

HelloWorld
- Process Files
  - HelloWorld.wsdl
  - helloWorld.bpel
- Referenced Resources

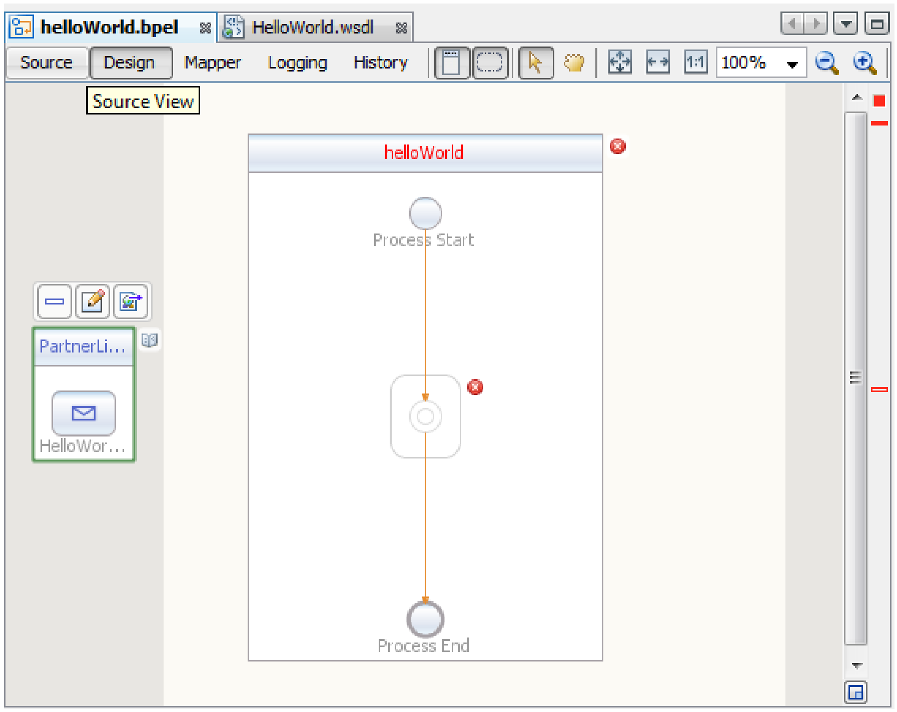
http://j2ee.netbeans.org/wsdl/HelloWorld/src/HelloWorld
- Types
- Imports
- Messages
  - HelloWorldOperationRequest
  - HelloWorldOperationResponse
- Port Types
  - HelloWorldPortType
- Bindings
- Services
- Extensibility Elements
  - HelloWorld

6.2 Create a BPEL

Click back on the tab helloWorld.bpel. Select HelloWorld.wsd and drag it onto the grey left strip of the BPEL editor.
When you position the mouse cursor in the grey left strip an orange circle appears. Drop the wsdl there.

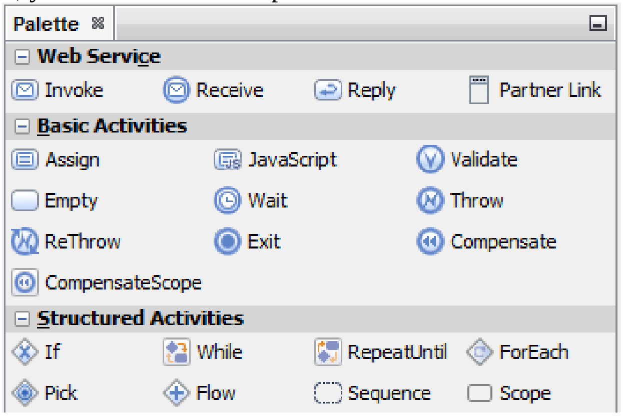


After dropping the WSDL on the orange circle, a PartnerLink appears in the BPEL Editor.



6.2.1 Add BPEL activity

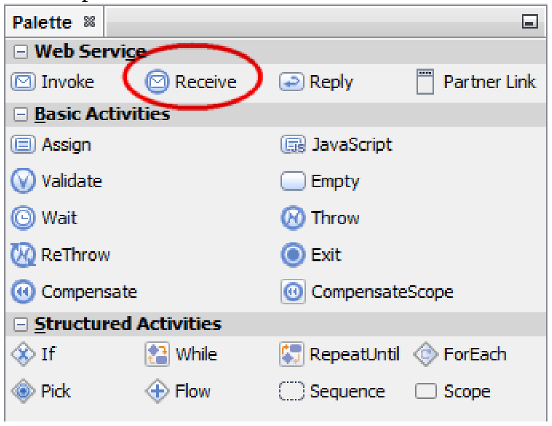
On the right side of OE-Studio, you can see the BPEL palette.



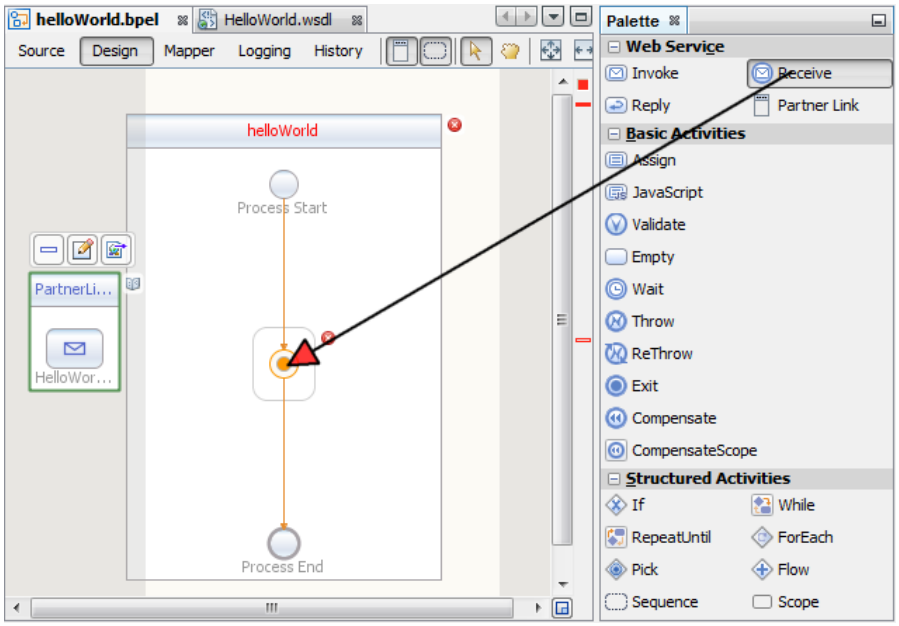
For this document, we will use 3 BPEL activities:

- Receive: used to receive the message
- Assign: map the input variable to the output variable
- Reply: return the message.

Then drag and drop the activities to design the process.
Select the Receive activity in the BPEL palette.

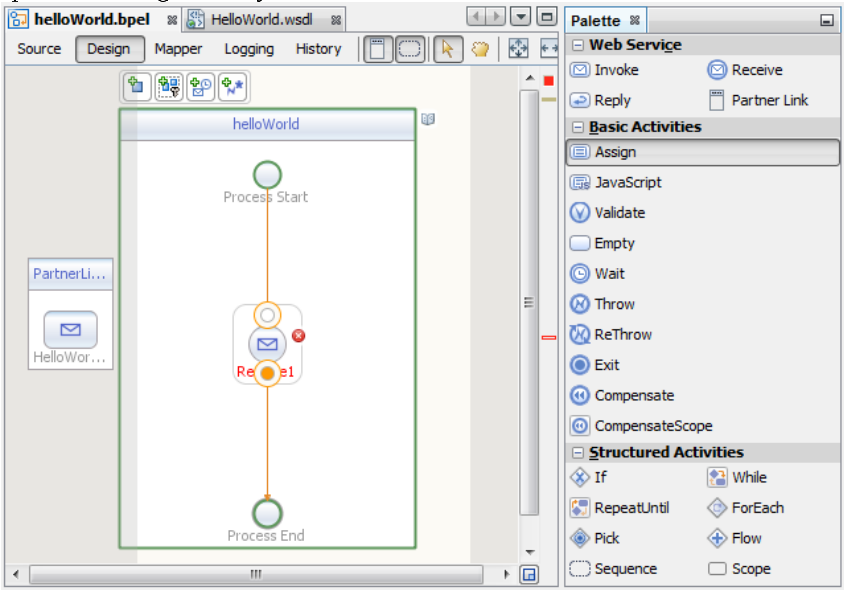


Drag the activity to the centre of the BPEL Editor.

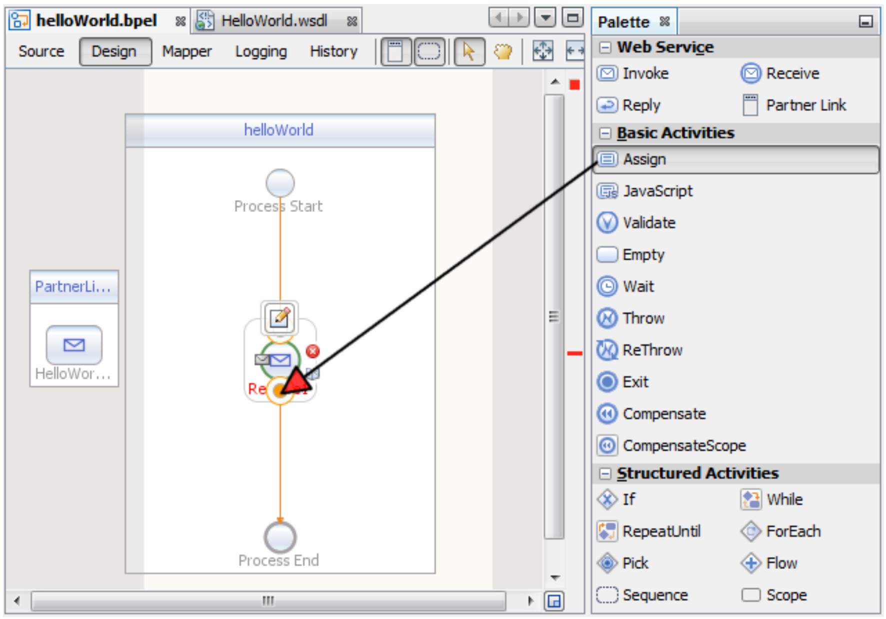


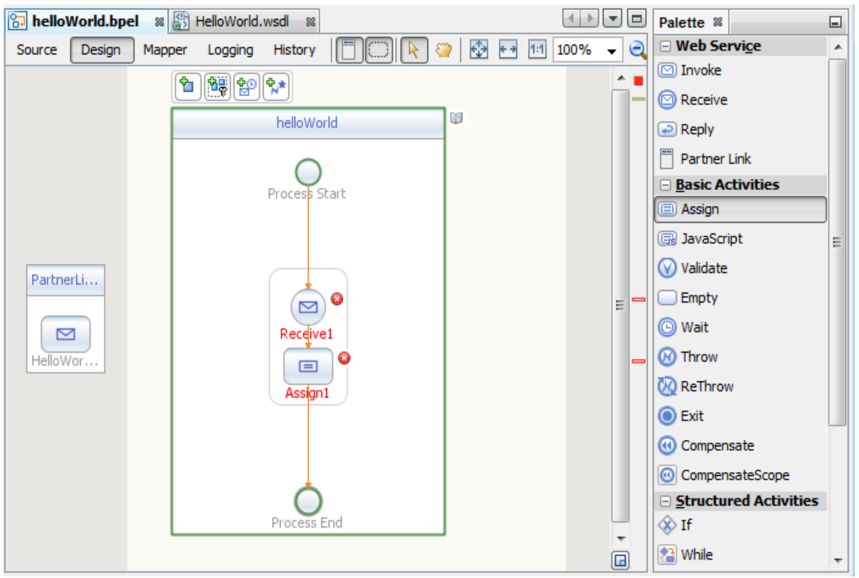
Drop the activity on the Orange circle and the Received activity appears in the BPEL Editor.

Follow the same steps to the Assign activity.

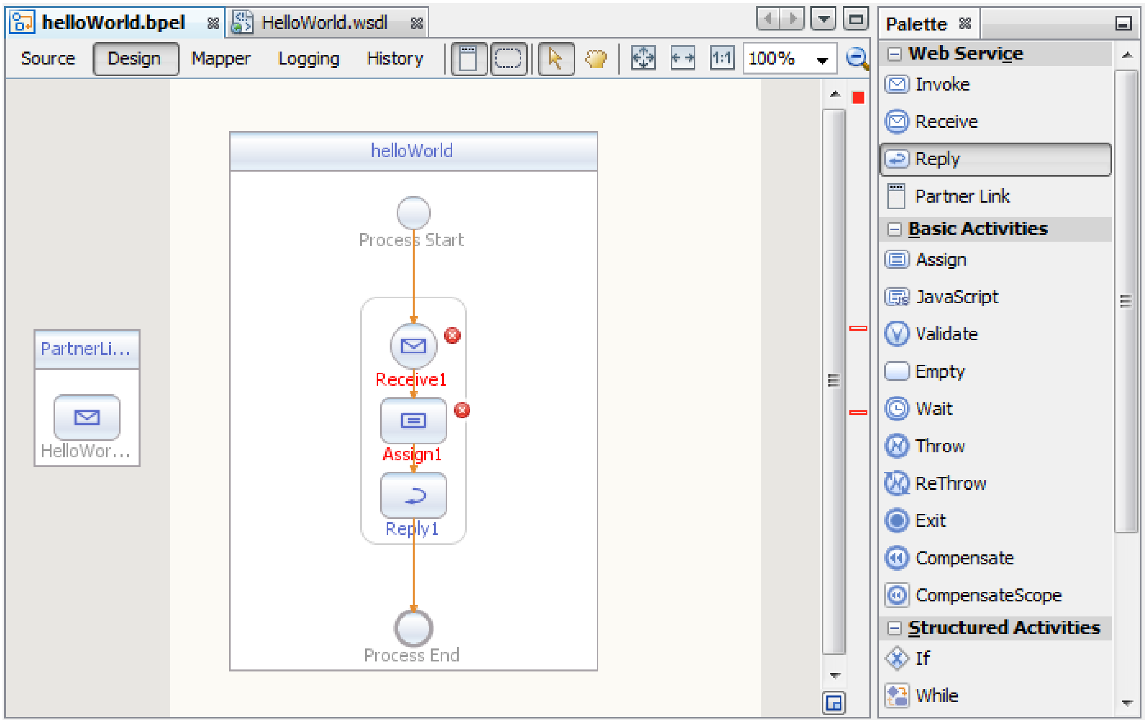


Drag and Drop the Assign icon on the Orange beneath the Receive activity.



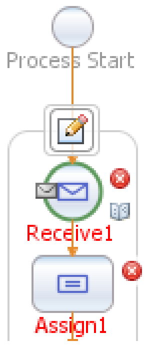


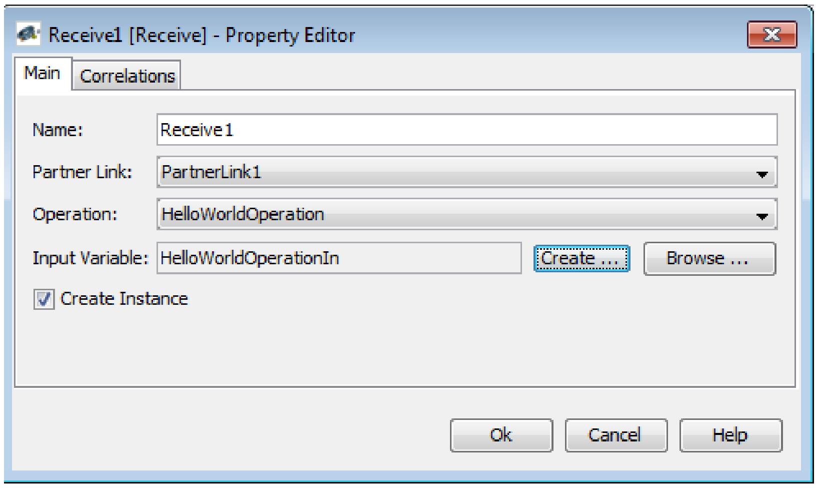
Do the same with the Reply icon.



6.2.2 Link the activities with the PartnerLink(s)

Click on the Receive icon, a small box appears.Click on the small box to edit the Activity properties.




Receive1 [Receive] - Property Editor

Main | Correlations

Name: Receive1

Partner Link: PartnerLink1

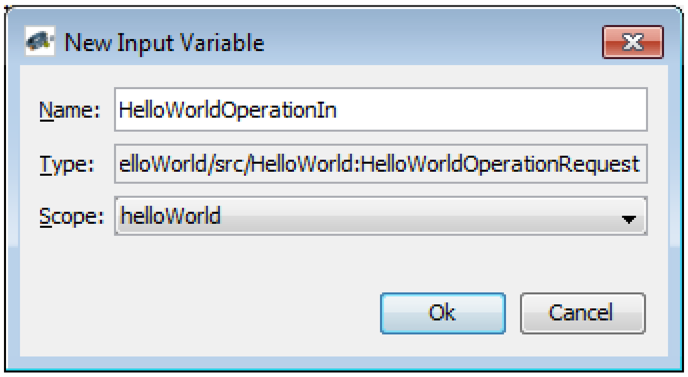
Operation: HelloWorldOperation

Input Variable: HelloWorldOperationIn [Create ...] [Browse ...]

[✓] Create Instance

Ok | Cancel | Help

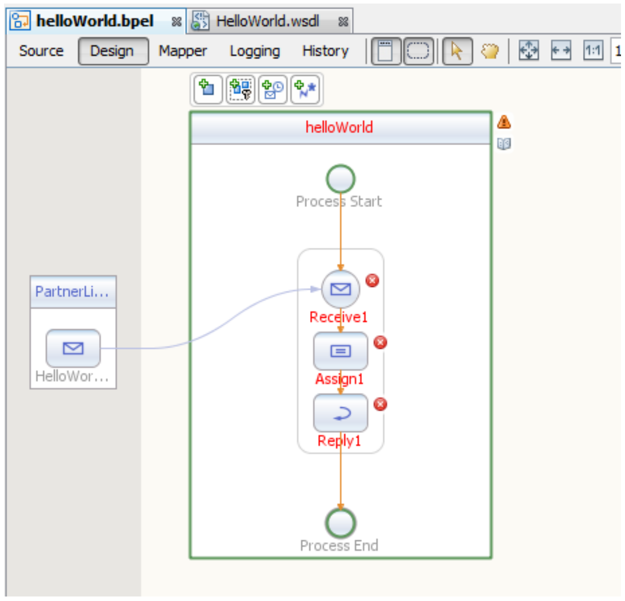
Partner Link: Select PartnerLink1
Operation: Select HelloWorldOperation
Input Variable: Click on Create button.



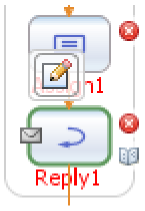
Name: HelloWorldOperationIn
Type: elloWorld/src/HelloWorld:HelloWorldOperationRequest
Scope: helloWorld

Keep the values by default. Then click on OK.

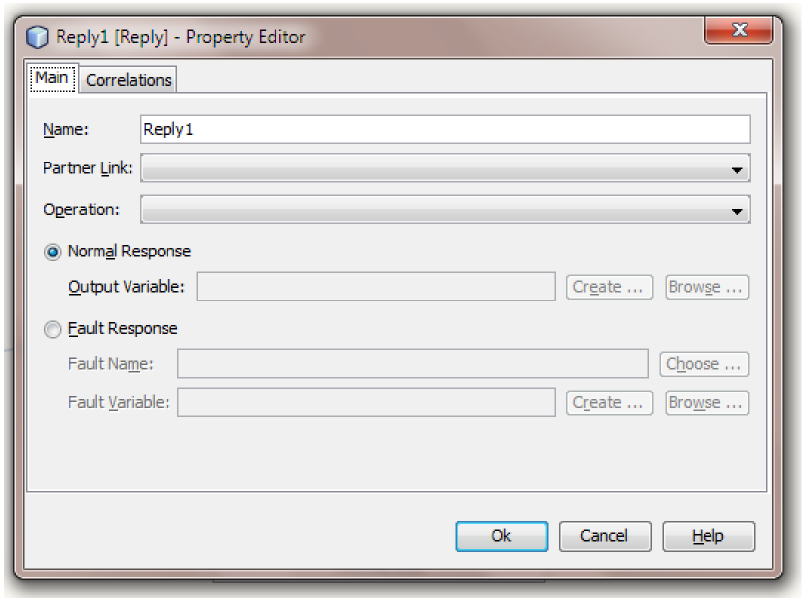
Click OK to validate the properties.



A link has been created between PartnerLink1 and the Receive activity.



Click on the Reply1 activity then click on the properties box.
Do the same process with Reply. Double click on Reply1.


Reply1 [Reply] - Property Editor

Main | Correlations

Name: Reply1

Partner Link:

Operation:

(● Normal Response)

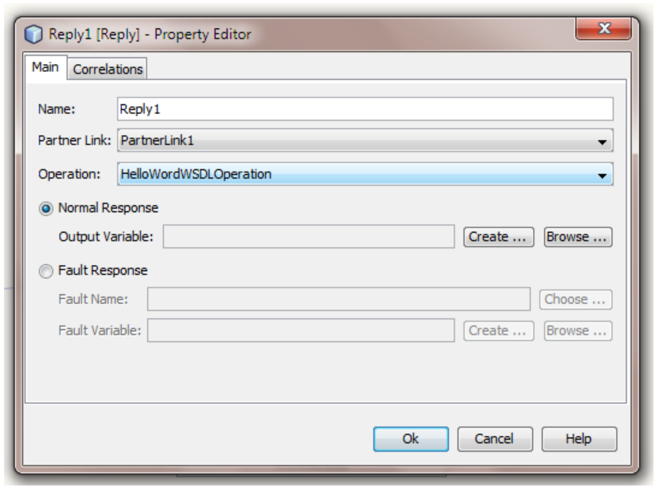
Output Variable:

Fault Response

Fault Name:

Fault Variable:

The pop-up Reply-Property Editor opens. Select the default values for “Partner Link” and “Operation” and set this Reply activity as follows.


Reply1 [Reply] - Property Editor

Main | Correlations

Name: Reply1

Partner Link: PartnerLink1

Operation: HelloWordWSDLOperation

(● Normal Response)

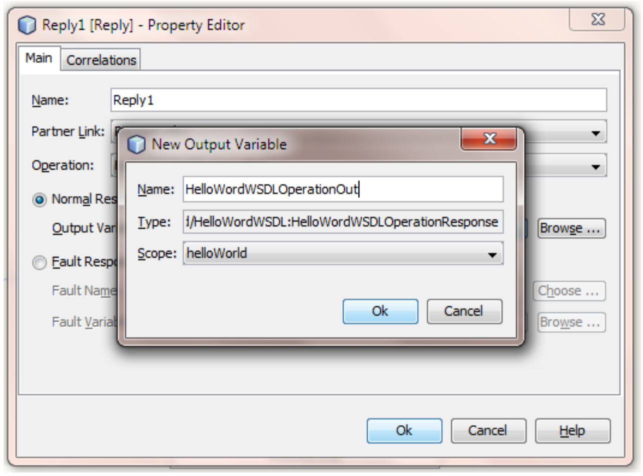
Output Variable:

Fault Response

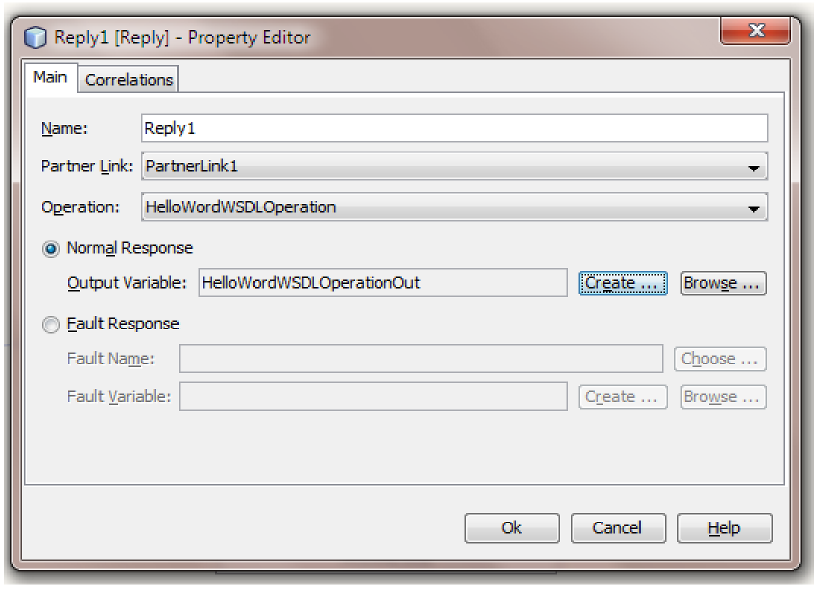
Fault Name:

Fault Variable:

Click on Create.



Keep the default values and click on OK.



Name: Reply1

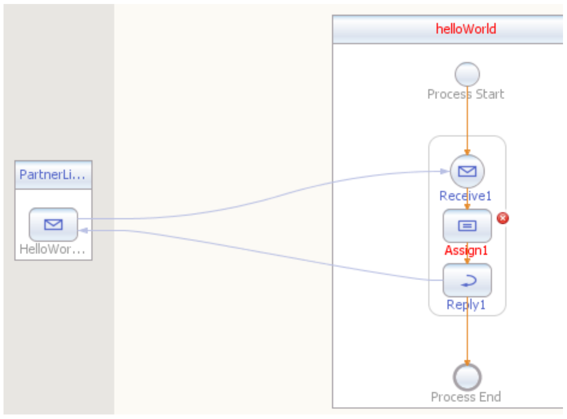
Partner Link: PartnerLink1

Operation: HelloWordWSDLOperation

(● Normal Response)

Output Variable: HelloWordWSDLOperationOut

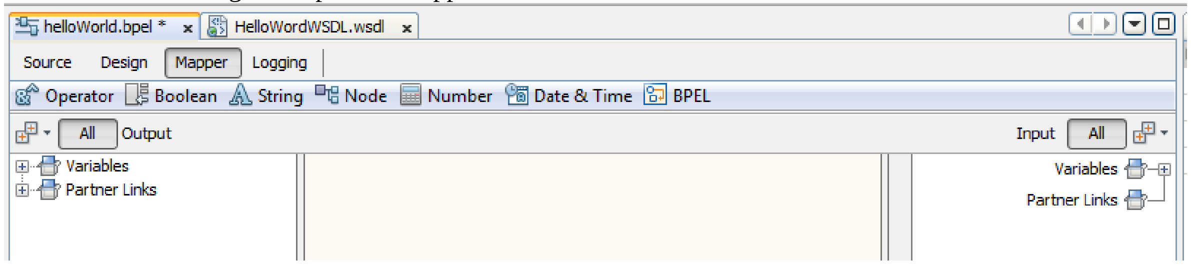
The Receive activity is now set up. Click on OK.



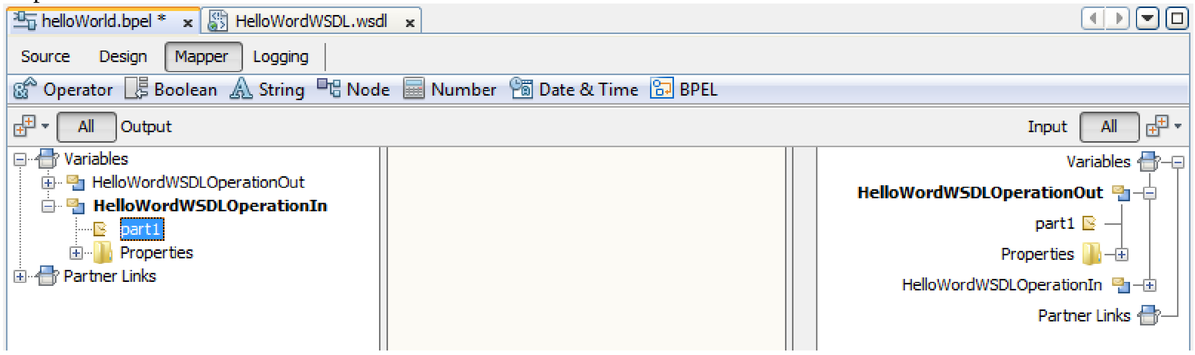
The set-up creates a link between the partner link and the activity Reply. This means that when a message is sent to Reply1 activity, it is also forwarded to Partner link.

6.2.3 Setup Assign activity

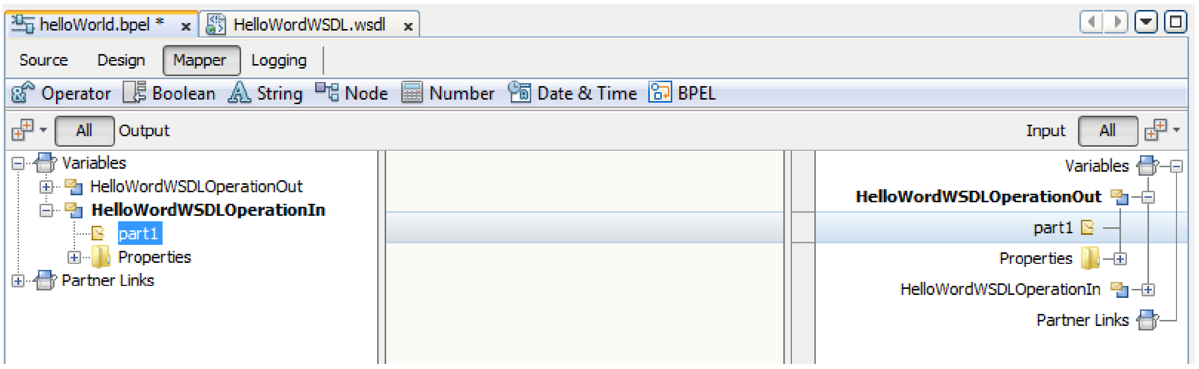
Assign is the activity that maps the input messages to output messages. This is the place where we concatenate "Hello"+ the input name.
Double click on Assign1 to open the mapper editor.



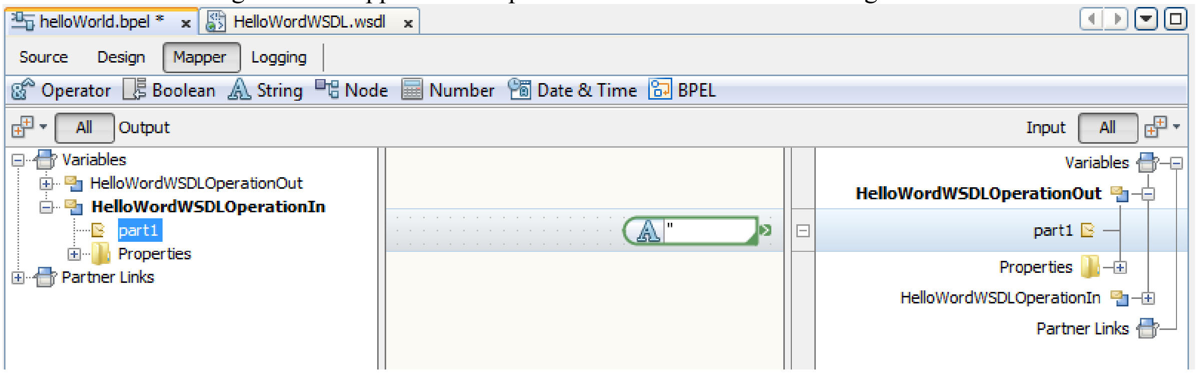
Expand the variable nodes on both sides.



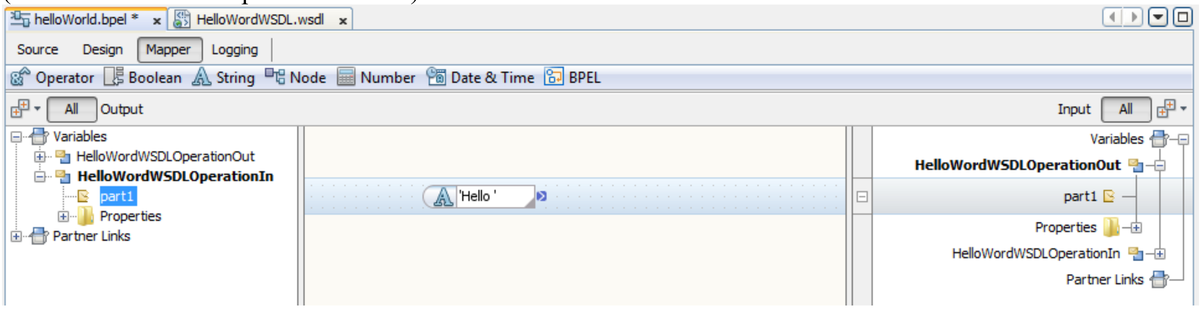
Click on HelloWorldWSDLOperationOut→part1. A blue lane appears at the part1 level.



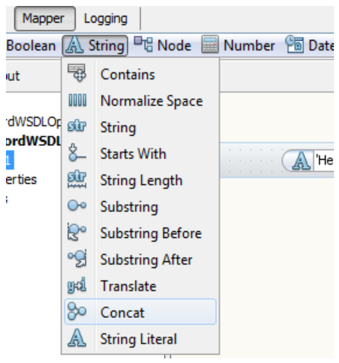
Select the Icon “String’ on the mapper editor top menu and select the item “String Literal”.



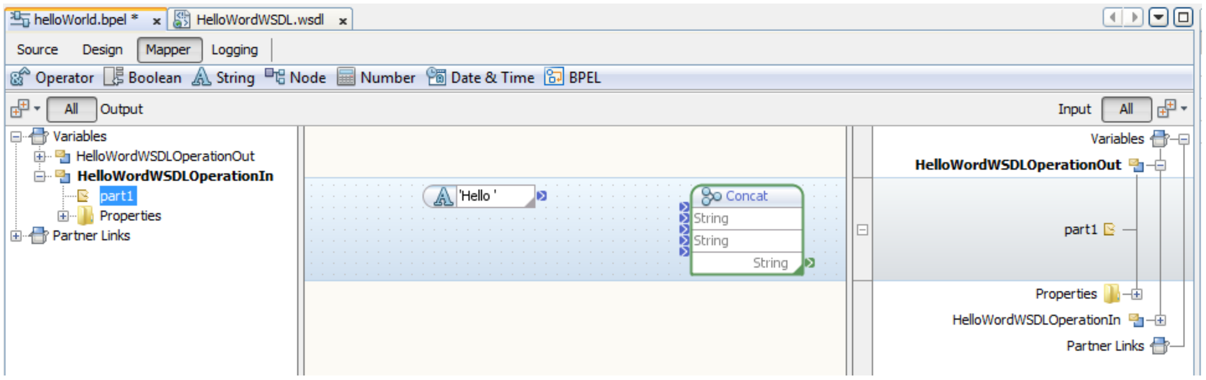
A new icon appears on the blue lane. It is the place to store “Hello ”. Double click on the icon and type “Hello” (without “ but with a space after Hello).



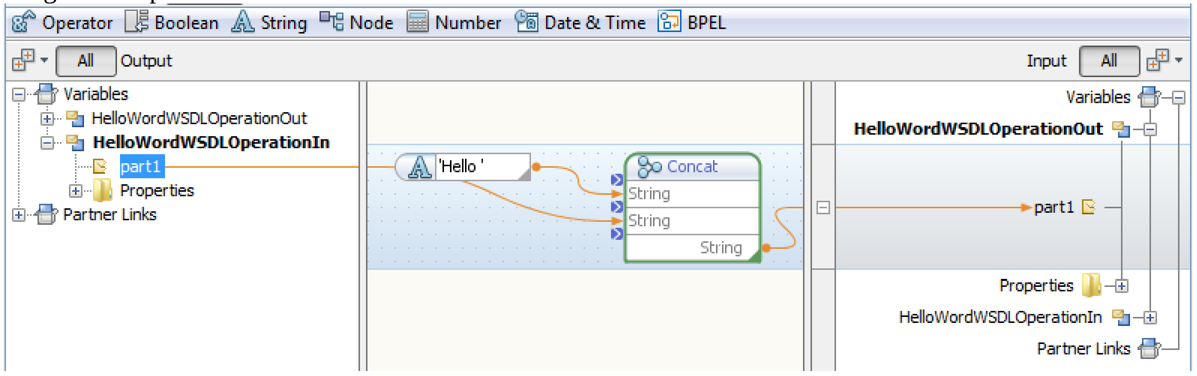
Select the Icon “String’ on the mapper editor top menu and select the item “Concat”.



A Concat icon is added in the blue lane next to “Hello”.

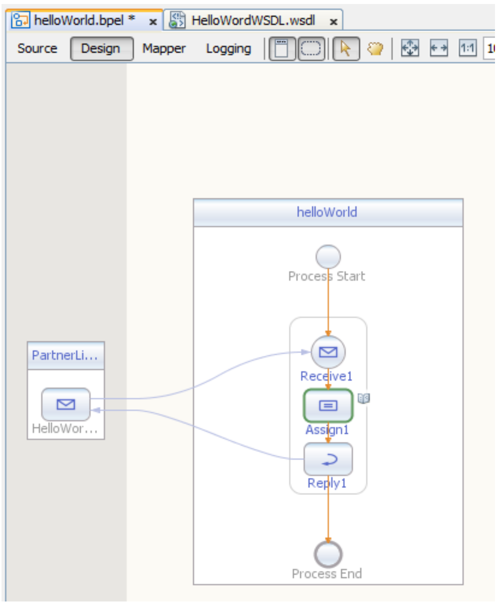


Drag and Drop then link the element as shown below.



1-Link Hello to Concat input.
2-Link HelloWorldWSDLOperationIn→part1 to another Concat input.
3-Link Concat output to HelloWorldWSDLOperationOut→part1.

Concat = concatenates Hello + the name found in the input message and put the result into the output message.



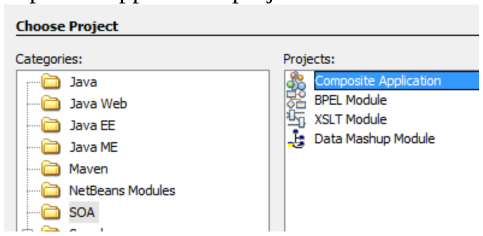
Click on design to come back to the BPEl editor. Note that the red signs are not present anymore; it indicates your BPEL is valid without error.

Save your BPEL. Click on icon or type ctrl-shift-S, then close the BPEL editor.

7 Create a Composite Application

To deploy an application with OpenESB you have to create a composite application. A composite application is the deployable entity for OpenESB.
Right click on the project tab and select “New project “.

Select the SOA Category and the Composite Application project.

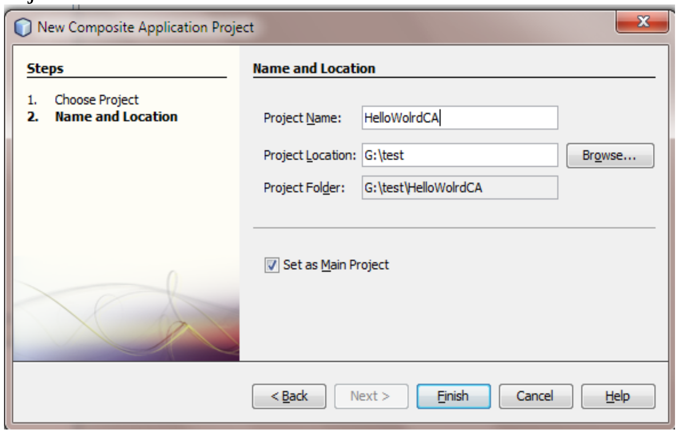


Choose Project

Categories: Java, Java Web, Java EE, Java ME, Maven, NetBeans Modules, SOA

Projects: Composite Application, BPEL Module, XSLT Module, Data Mashup Module

Click on Next and set the project name.



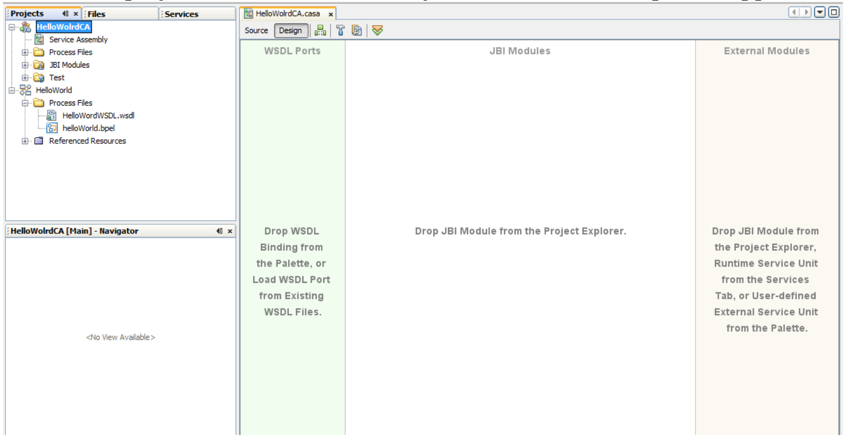
New Composite Application Project

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

Name and Location

Project Name: HelloWolrdCA
Project Location: G:\test
Project Folder: G:\test\HelloWolrdCA
Set as Main Project [✓]

Click on Finish
After a while, HelloWorldCA project is added into the Project tab, and the Composite Application Editor opens.



Projects | Files | Services
- HelloWolrdCA
  - Service Assembly
  - Process Files
  - JBI Modules
  - Test
- HelloWorld
  - Process Files
    - HelloWordWSDL.wsdl
    - helloWorld.bpel
  - Referenced Resources

HelloWolrdCA [Main] - Navigator
<No View Available>

HelloWolrdCA.casa

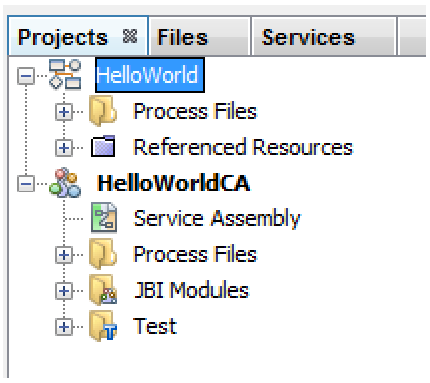
WSDL Ports: Drop WSDL Binding from the Palette, or Load WSDL Port from Existing WSDL Files.

JBI Modules: Drop JBI Module from the Project Explorer.

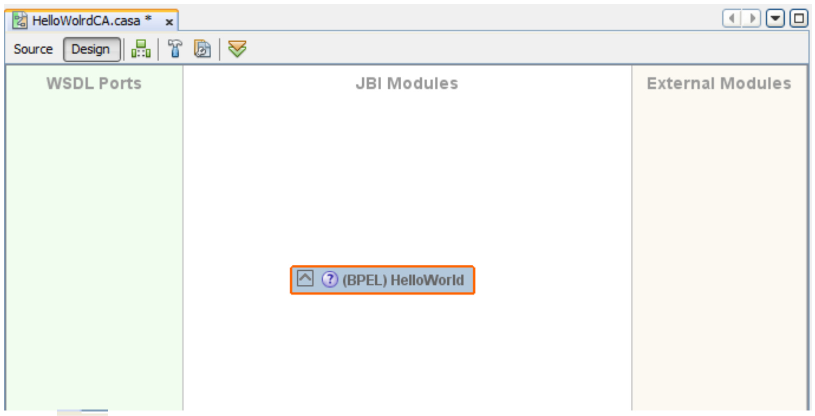
External Modules: Drop JBI Module from the Project Explorer, Runtime Service Unit from the Services Tab, or User-defined External Service Unit from the Palette.

7.1 Add the BPEL Module in a composite application

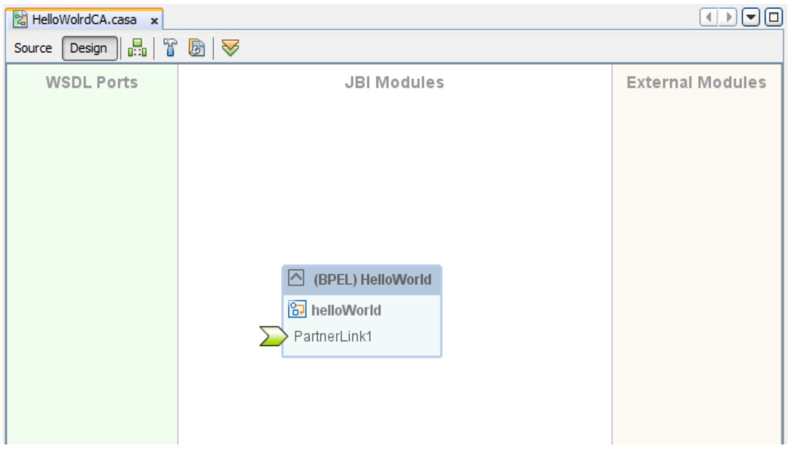
Select HelloWorldCA project.



Drag it and drop it in the JBI Module Lane, in the Composite Application editor.



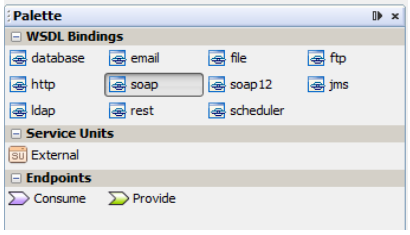
Click on the build button represented by a hammer.



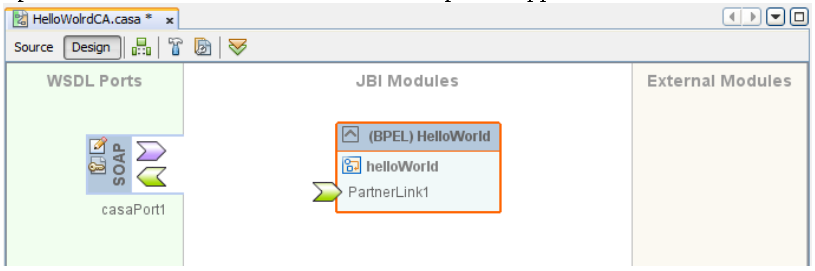
Composite Editor introspects Hello Word project and find out the partner links associated with the project.

7.2 Add a Binding Component

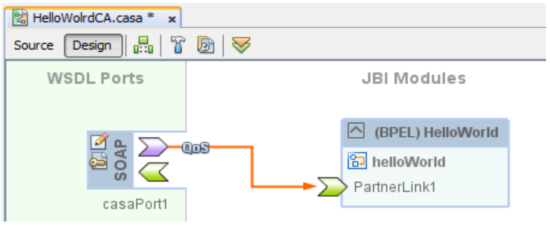
Our project is ready to be deployed but we have to precise how the project will receive the messages. To send messages to the BPEL module, we use a SOAP protocol. You have to define a SOAP port where the message will be sent.
On the right part of the screen, in the palette, select the icon SOAP (not SOAP12).



Drag the icon and drop it on the WSDL Ports lane in the Composite application editor.



A SOAP icon appears on the WSDL Ports Lane.
With a simple drag and drop Link SOAP and BPEL.

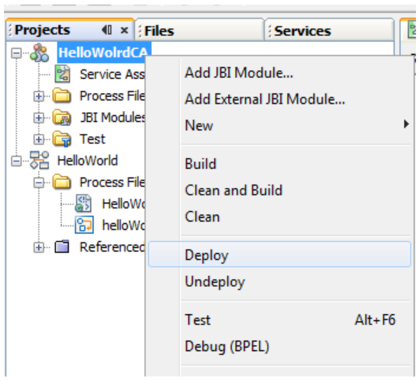


Save the project: Type Crtl-shift-S.

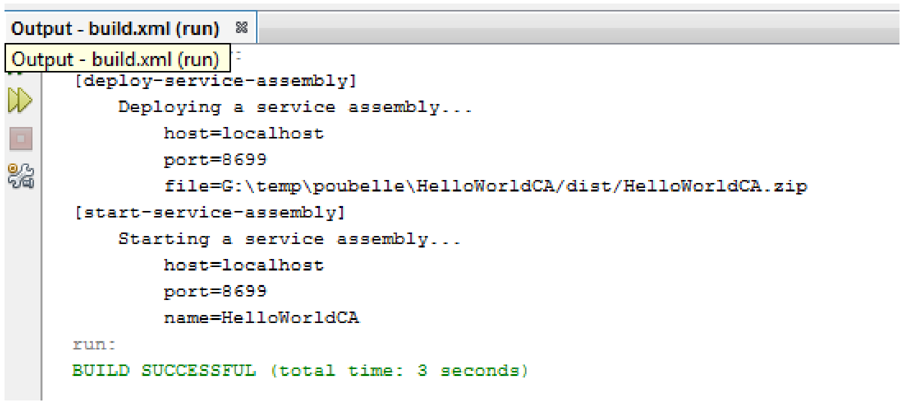
8 Deployment

8.1 Deploy the composite application

Go back to the Projects tab. Select HelloWorldCA. Click right and select “Deploy”.



After a few seconds, at the bottom of your screen, in the build.xml tab in the output window, check if the build is successful.

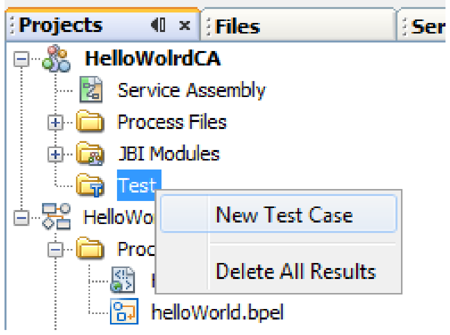


```
[deploy-service-assembly]
    Deploying a service assembly...
        host=localhost
        port=8699
        file=G:\temp\poubelle\HelloWorldCA/dist/HelloWorldCA.zip
[start-service-assembly]
    Starting a service assembly...
        host=localhost
        port=8699
        name=HelloWorldCA
run:
BUILD SUCCESSFUL (total time: 3 seconds)
```

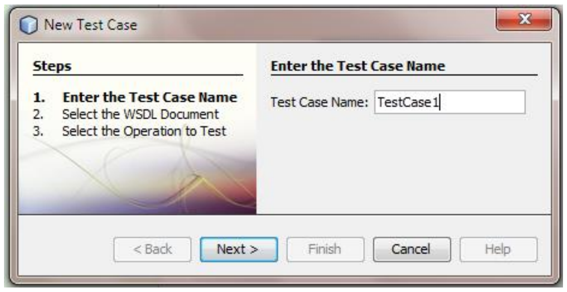
Your hello World application is now deployed.

9 Test the application.

Now let’s test the application. The OpenESB has a simple test generator. It allows you to test any SOAP port in your applications. Select the node “Test” in the HelloWorldCA project. Right click and select “New Test case”.

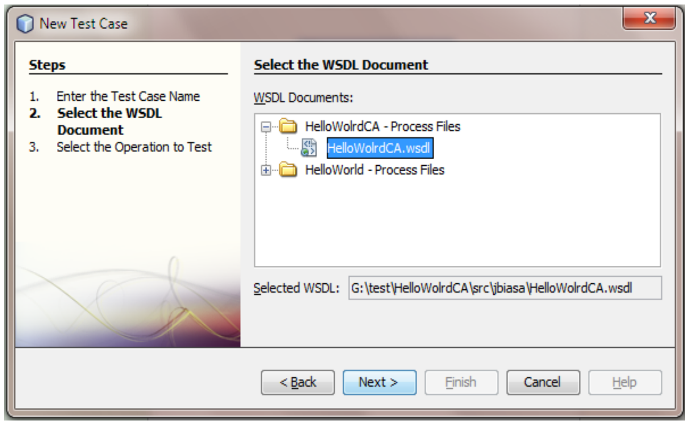


Give a name to the test case and click Next.


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
< Back
Next >
Finish
Cancel
Help

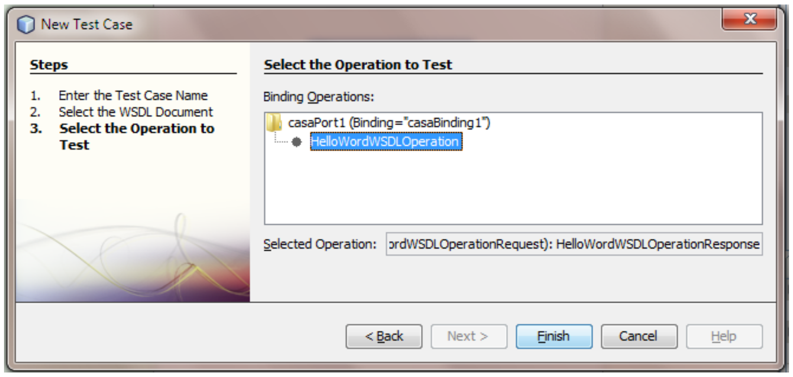
Test Case Name: TestCase1

Select the HelloWorldCA and click Next.

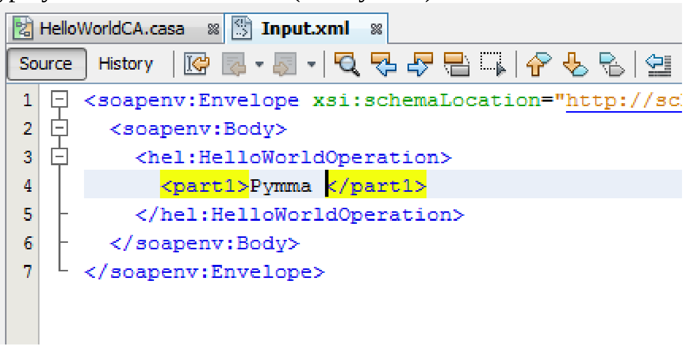

New Test Case
Steps
1. Enter the Test Case Name
2. Select the WSDL Document
3. Select the Operation to Test
Select the WSDL Document
WSDL Documents:
HelloWolrdCA - Process Files
HelloWolrdCA.wsdl
HelloWorld - Process Files
< Back
Next >
Finish
Cancel
Help

Selected WSDL: G:\test\HelloWolrdCA\src\jbiasa\HelloWolrdCA.wsdl

Select the HelloWorldWSDLOperation and click Finish.

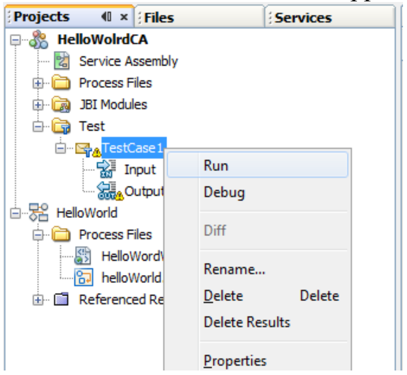

New Test Case
Steps
1. Enter the Test Case Name
2. Select the WSDL Document
3. Select the Operation to Test
Select the Operation to Test
Binding Operations:
casaPort1 (Binding="casaBinding1")
HelloWordWSDLOperation
Selected Operation: ›rdWSDLOperationRequest): HelloWordWSDLOperationResponse
< Back
Next >
Finish
Cancel
Help

The XML Editor opens. Type your name in the Part1 (Ex: Pymma).

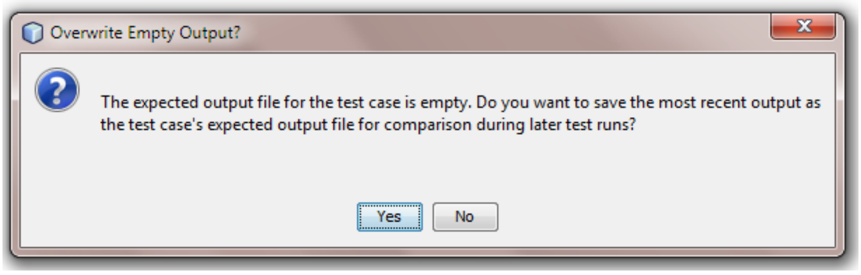


```
<soapenv:Envelope xsi:schemaLocation="http://sc
  <soapenv:Body>
    <hel:HelloWorldOperation>
      <part1>Pymma </part1>
    </hel:HelloWorldOperation>
  </soapenv:Body>
</soapenv:Envelope>
```

Save the XML document: Ctrl-Shift-S. The new test case “TestCase1” appears in the hierarchy. Run TestCase1.

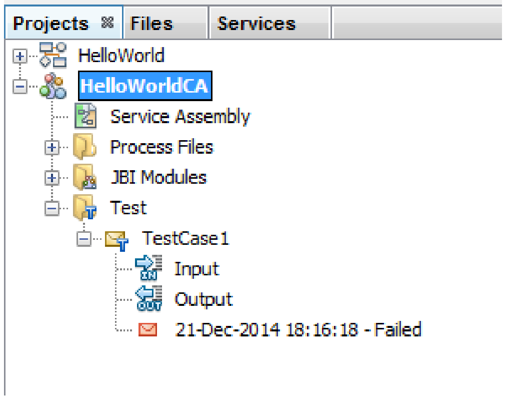


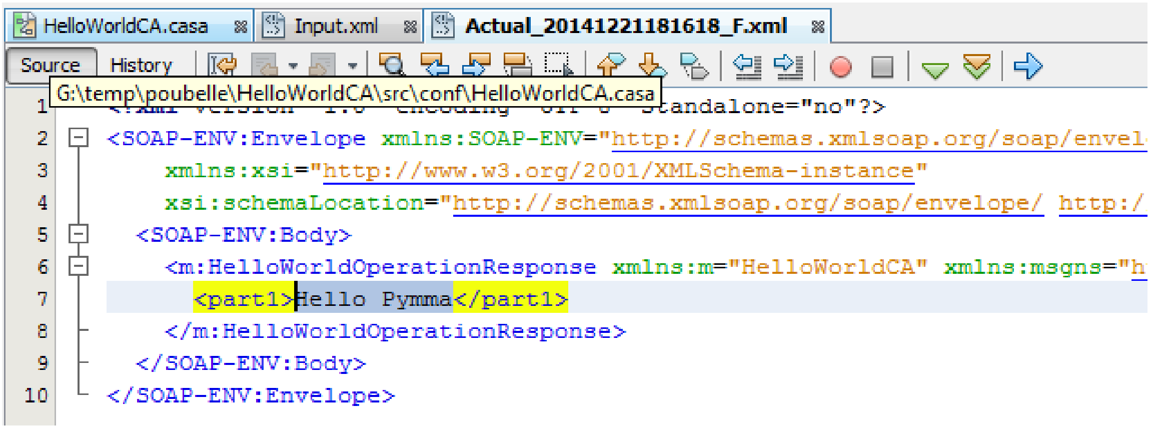
After a while the following pop-up appears.



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?

Double click on the output and ignore the message saying Failed that opens afterwards.




HelloWorldCA.casa
Input.xml
Actual_20141221181618_F.xml
Source
History
G:\temp\poubelle\HelloWorldCA\src\conf\HelloWorldCA.casa

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envel
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://schemas.xmlsoap.org/soap/envelope/ http:/
  <SOAP-ENV:Body>
    <m:HelloWorldOperationResponse xmlns:m="HelloWorldCA" xmlns:msgns="h
      <part1>Hello Pymma</part1>
    </m:HelloWorldOperationResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The unit test failed because we did not set up an output for the test. Do the test again and you won't get an error.

Congratulation!!! It works. You're the best ;-).

10 Next steps

The next step will be to install the OpenESB components and Libraries on your instance then create and deploy your project with the OpenESB Studio.
We advise you to read our documents: **770-003 OE Web admin console** and the administrative guide: **770-004 OE Admin Guide.**

11 Help and support

11.1 From the community

You can find all our OpenESB documentations on the OpenESB official web site:
www.open-esb.net.
If you have any questions or would like to share your feedback, use the OpenESB forum at:
http://**openesb**-community-forum.794670.n2.**nabble**.com
Feel free to notify us with a bug or suggest how to improve our services on :
https://openesb.atlassian.net/secure/Dashboard.jspa

11.2 From Pymma

Pymma is deeply involved in the community and offers services and consulting on OpenESB. Pymma has professional services that can assist you from the development of your SOA design, implementation and ongoing management. All of our skills and background are based on our extensive first-hand experience and industry-leading methods.

Pymma releases an OpenESB Enterprise Edition with many additional enterprise features and a professional support.

In addition to OpenESB development, Pymma designed a new Service-Oriented development process named Rebecca to help business, architect and development team during the design and the implementation of their service oriented projects with OpenESB or any other service oriented development tool.

Feel free to contact us by email at contact@pymma.com for any further information on our OpenESB Services.

11.3