**OpenESB SSL Installation**

User Guide

**Author:** Akshay Thakur

**Designation:** Technical Consultant

**akshay.thakur1388@gmail.com**

+91 9923246991

Created: 24-Dec-2016

Contents

[1 Introduction 3](#_Toc470457765)

[2 Prerequisites 4](#_Toc470457766)

[3 SSL Installation 5](#_Toc470457767)

[3.1 Using Self Signed SSL 5](#_Toc470457768)

[3.2 Using CA Signed SSL 10](#_Toc470457769)

[4 SSL Testing 11](#_Toc470457770)

[5 OpenESB Project Creation 13](#_Toc470457771)

# Introduction

This document will focus on complete process of SSL installation in OpenESB Integration server v3.0.5. It also covers installing Self signed SSL and CA (Certified Authority) signed SSL certificate process.

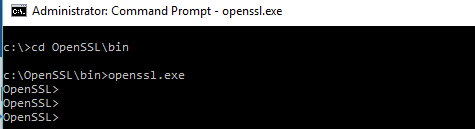
# Prerequisites

1. Before proceeding with the SSL installation process, we require OpenSSL utility to be installed in our local system. Below is the link for download.

Download Link: <https://sourceforge.net/projects/openssl/>

Once the installation is done, open the cmd and check OpenSSL is accessible.

Go to OpenSSL home directory having bin folder and then type openssl.exe, below window should appear.



1. Current HTTP Binding Component running on OpenESB 3.0.5 doesn’t support SSL communication due to some code issue therefore please uninstall current Http BC from OpenESB and install below attached binding component for Http BC in Open ESB application.

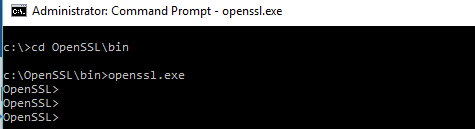
Download Link: <https://drive.google.com/a/inspirage.com/file/d/0B1Uzxdw1z4CKUldIbXA4SGxwc28/view?usp=sharing>

# SSL Installation

## Using Self Signed SSL

Steps to follow,

1. Start OpenSSL utility using command prompt.

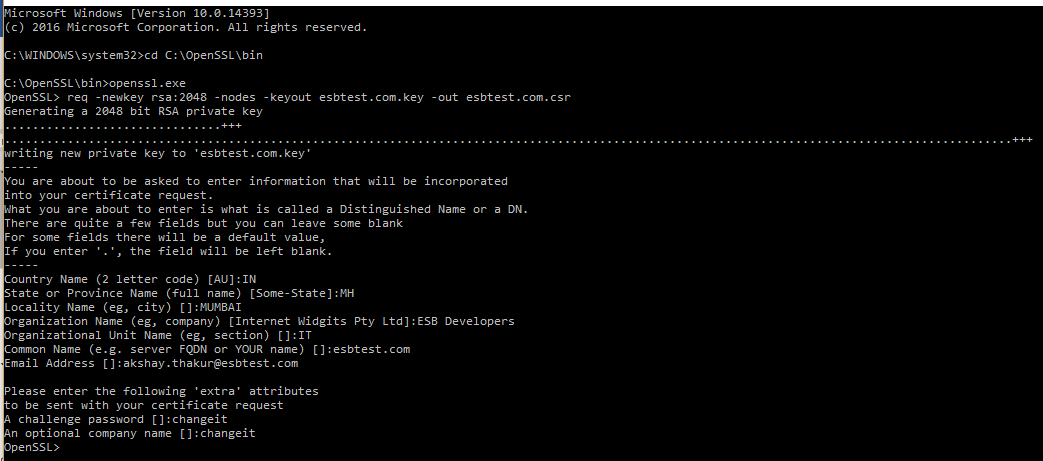


1. Generate a Private Key and a CSR (Certificate Signing Request)

**Command:** req -newkey rsa:2048 -nodes -keyout **name**.key -out **name**.csr

Name: This will be any user defined name or you can use server hostname to specify for key and csr file.

**Example:** req -newkey rsa:2048 -nodes -keyout **esbtest.com.key** -out **esbtest.com.csr**



The above command will create 2 files in the $OpenSSL/bin folder, esbtest.com.key (Private Key File) and esbtest.com.csr (CSR File).

1. Generate a Self-Signed Certificate because we do not want to use CA (Certified Authority) to generate and sign the certificate in this process.

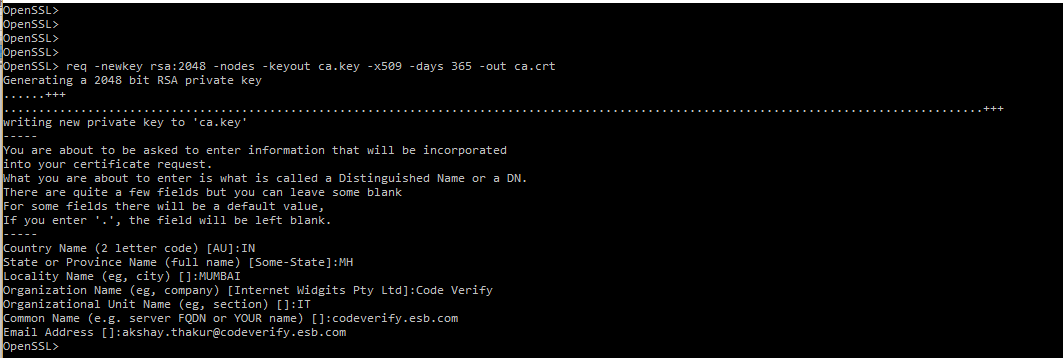
Command: req -newkey rsa:2048 -nodes -keyout **name**.key -x509 -**days** 365 -out **name**.crt

Name: This will be again any user defined name, but it should not be same as Step 2 above. As this is a Self-signed certificate, we need to name this CA as dummy name.

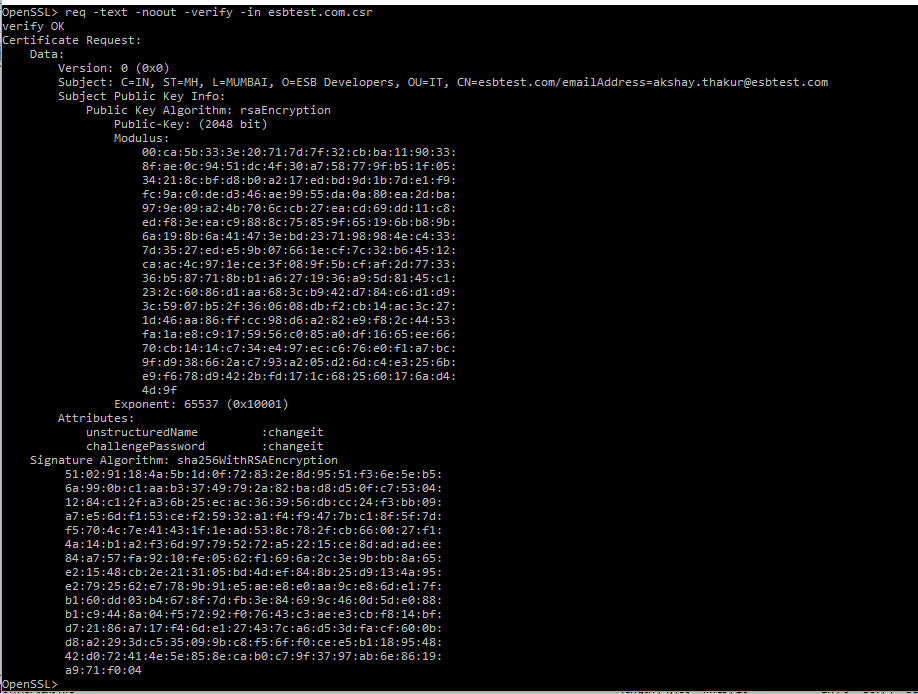
Days: You can specify any number of days for which you need this key to be valid.

Example: req -newkey rsa:2048 -nodes -keyout **ca**.key -x509 -days 365 -out **ca**.crt

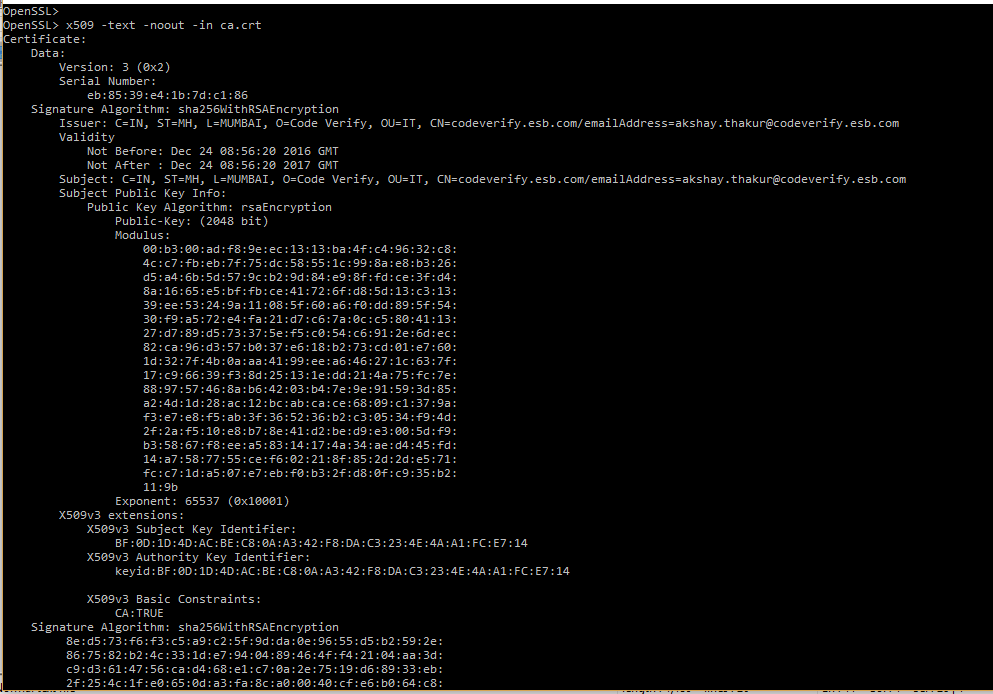
We will consider the name of CA as Code Verify with server name (i.e. FQDN – Full Qualified Domain Name) as codeverify.esb.com



1. Verify the above generated certificates using below commands.
2. View Certificates: req -text -noout -verify -in esbtest.com.csr

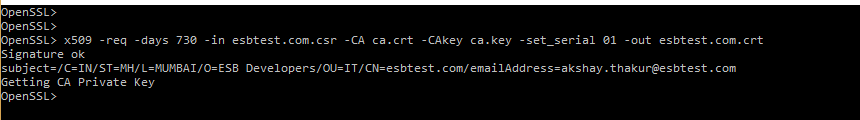


1. View Certificate Entries: x509 -text -noout -in ca.crt



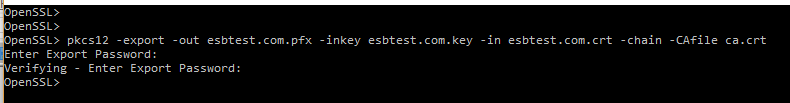
1. In this step, sign our CSR (Certificate Signing Request) created in Step 2 using dummy CA certificate and key created in Step 3, and we will generate CRT file which are used to verify a secure website's authenticity.

Command: x509 -req -days 730 -in esbtest.com.csr -CA ca.crt -CAkey ca.key -set\_serial 01 -out esbtest.com.crt

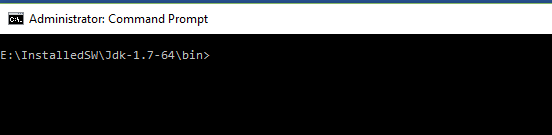


1. Package the keys and certs in a PKCS12 file (PKCS12 with CA and server certificate chain).

Command: pkcs12 -export -out esbtest.com.pfx -inkey esbtest.com.key -in esbtest.com.crt -chain -CAfile ca.crt

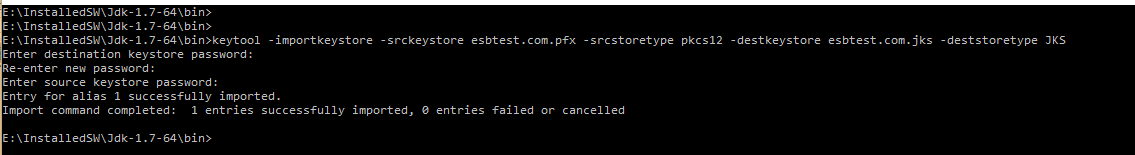


1. Once you have \*.pfx file created, you need to go to $JAVA\_HOME/bin location used for OpenESB application. Open this location in cmd terminal.



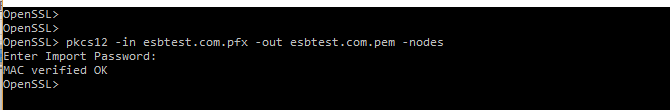
1. Now we will convert PKCS12 file created in Step 6 to JKS file using keytool utility provided by Java.

Command: keytool -importkeystore -srckeystore esbtest.com.pfx -srcstoretype pkcs12 -destkeystore esbtest.com.jks -deststoretype JKS



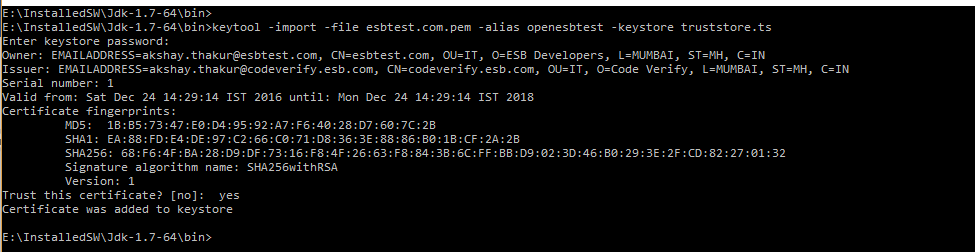
1. Go to OpenSSL/bin location and convert PFX file to PEM file which is required for importing into Truststore of Java Keystore.

Command: pkcs12 -in esbtest.com.pfx -out esbtest.com.pem -nodes

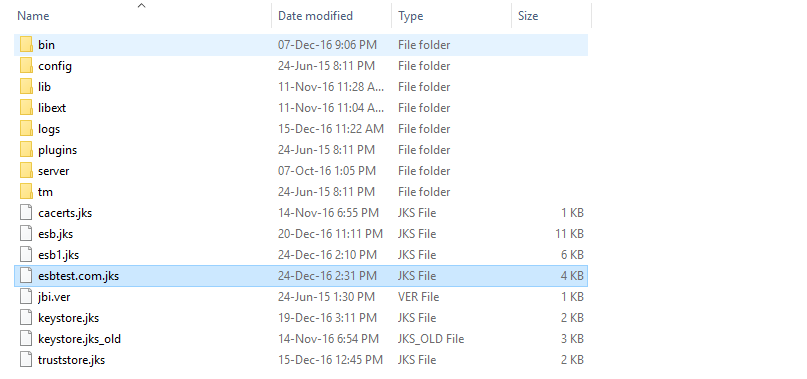


1. Add the certificate in Java Truststore using below command.

Command: keytool -import -file esbtest.com.pem -alias openesbtest -keystore truststore.ts



1. In the Step 10 output, please verify your Owner and Issuer Parties. Owner will be the one who is requesting for certificate and Issuer will be one who is providing the Signed certificate.
2. Copy \*.jks file in $OPEN\_ESB\_HOME/OE-Instance/ location. Screen shot below,



1. Go to OpenESB server startup location, $OPEN\_ESB\_HOME/OE-Instance/bin and open openesb.bat file for windows or openesb.sh for Linux machine and add below VM parameters,

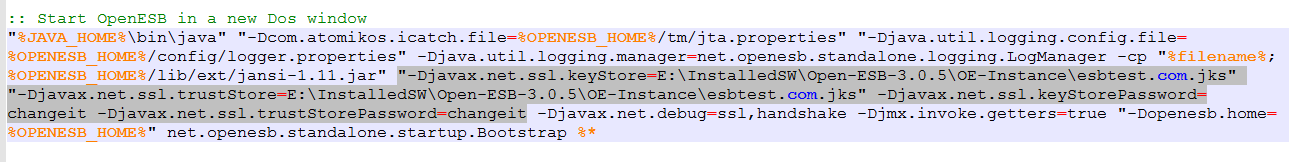
-Djavax.net.ssl.keyStore=path.jks

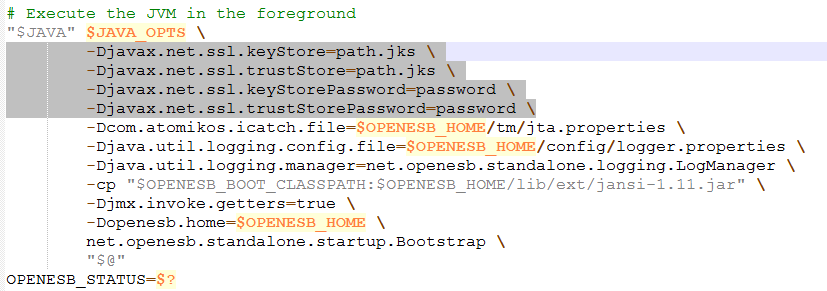
-Djavax.net.ssl.trustStore=path.jks

-Djavax.net.ssl.keyStorePassword=password

-Djavax.net.ssl.trustStorePassword=password

**For Windows:**



**For Linux:**

## Using CA Signed SSL

We will describe the process of obtaining the CA signed SSL certificate therefore screenshots are not covered here.

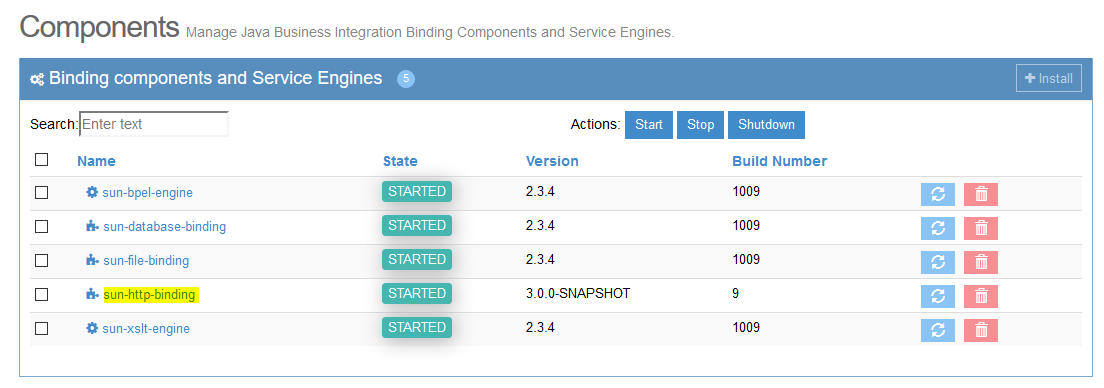
Steps to Follow,

1. Generate a Private Key and a CSR (Certificate Signing Request). The same process which we covered in Self-signed process. The \*.csr file is what you will send to the CA to request your SSL certificate.
2. Based on the CA you choose, each one has its own way of providing the SSL certificates. Some of the CA are, Namecheap, GoDaddy, DigiCert etc. Based on our CSR, CA will provide us 2 files, one is your SSL certificate (which should have a random name) and CA intermediate certificate bundle. Rename the certificate to the domain name with a .crt extension, e.g. example.com.crt, and rename the intermediate certificate bundle as intermediate.crt.
3. So now, in this case we have **example.com.crt** as our SSL certificate (as per Step 5 in Self-signed process) and **intermediate.crt** as our CA certificate (as per Step 3 in Self-signed process).
4. Now, follow the Steps 6 to 13 same as Self-signed process and install the certificate on OpenESB server.

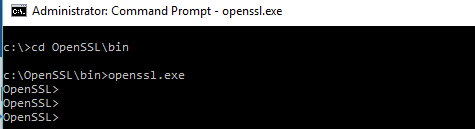
# SSL Testing

Steps for Testing SSL Connection,

1. Start the OpenESB application server and verify HTTP BC is up and running. Below is the sample screen shot,

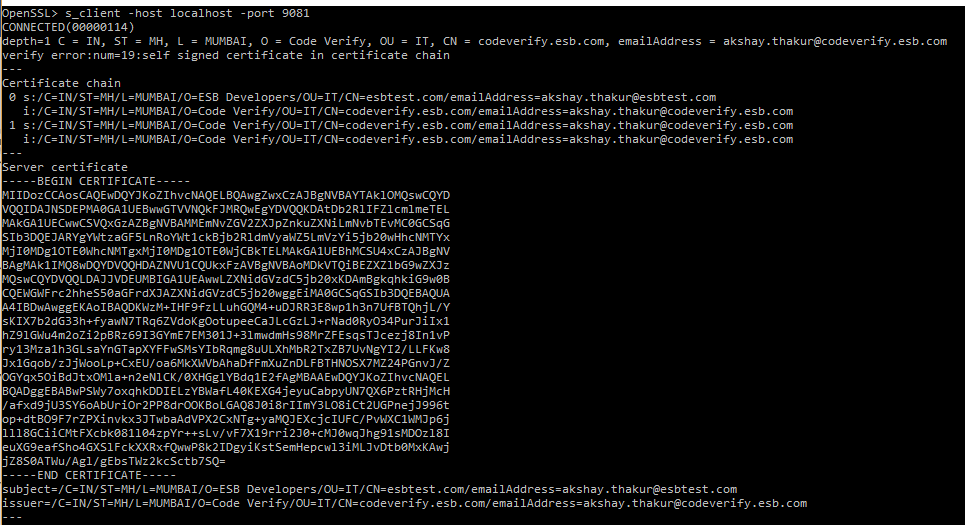


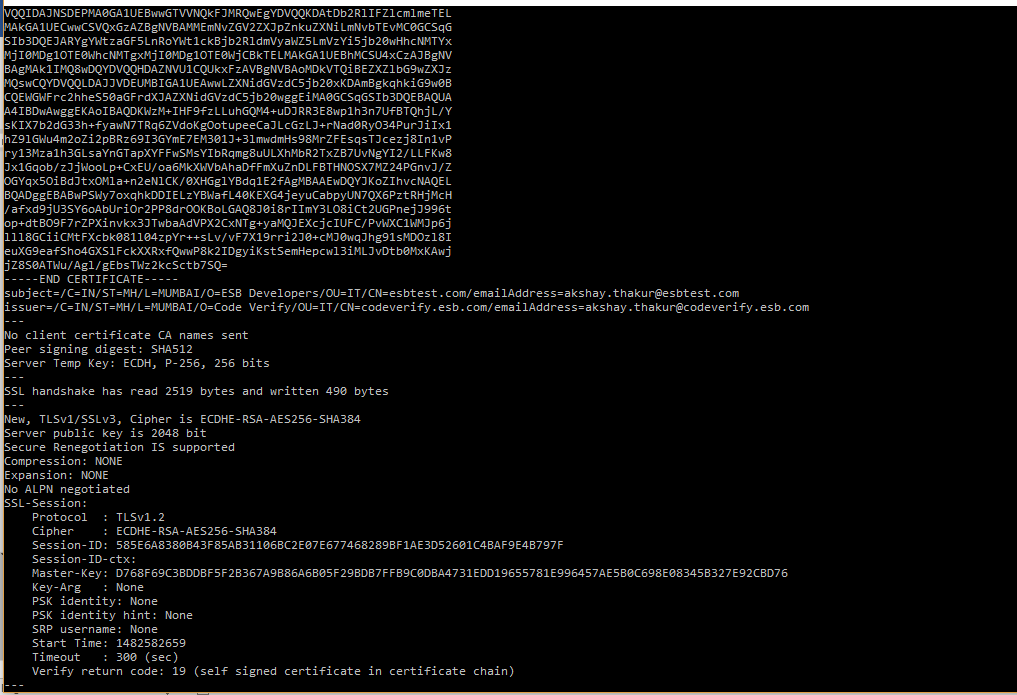
1. Go to command prompt and open the OpenSSL utility.



1. Type following command to initiate the SSL connection and press enter, you should be getting your server certificate on output screen.

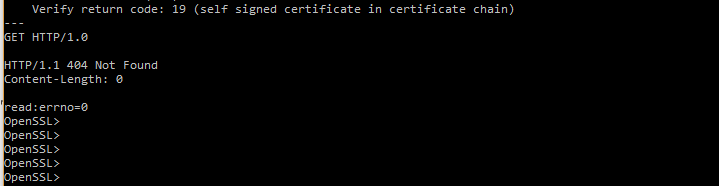
Command: s\_client -host localhost -port 9081





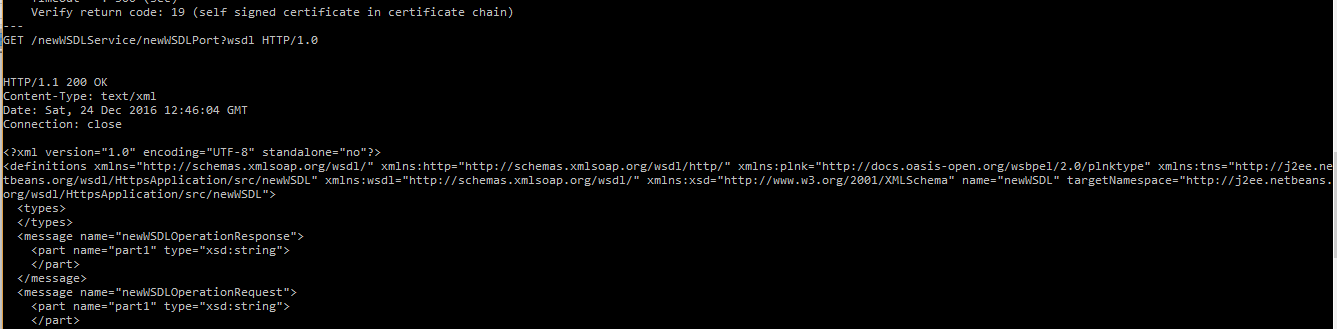
1. Now type below command to read the Http response from server. As we do not have any application working on 9081 port, we will receive Http 404.

Command: GET HTTP/1.0 and press enter 2 times to get the response.



1. If you have any OpenESB application deployed on 9081 port, then you will receive Http 200 response. Check below screen shot.

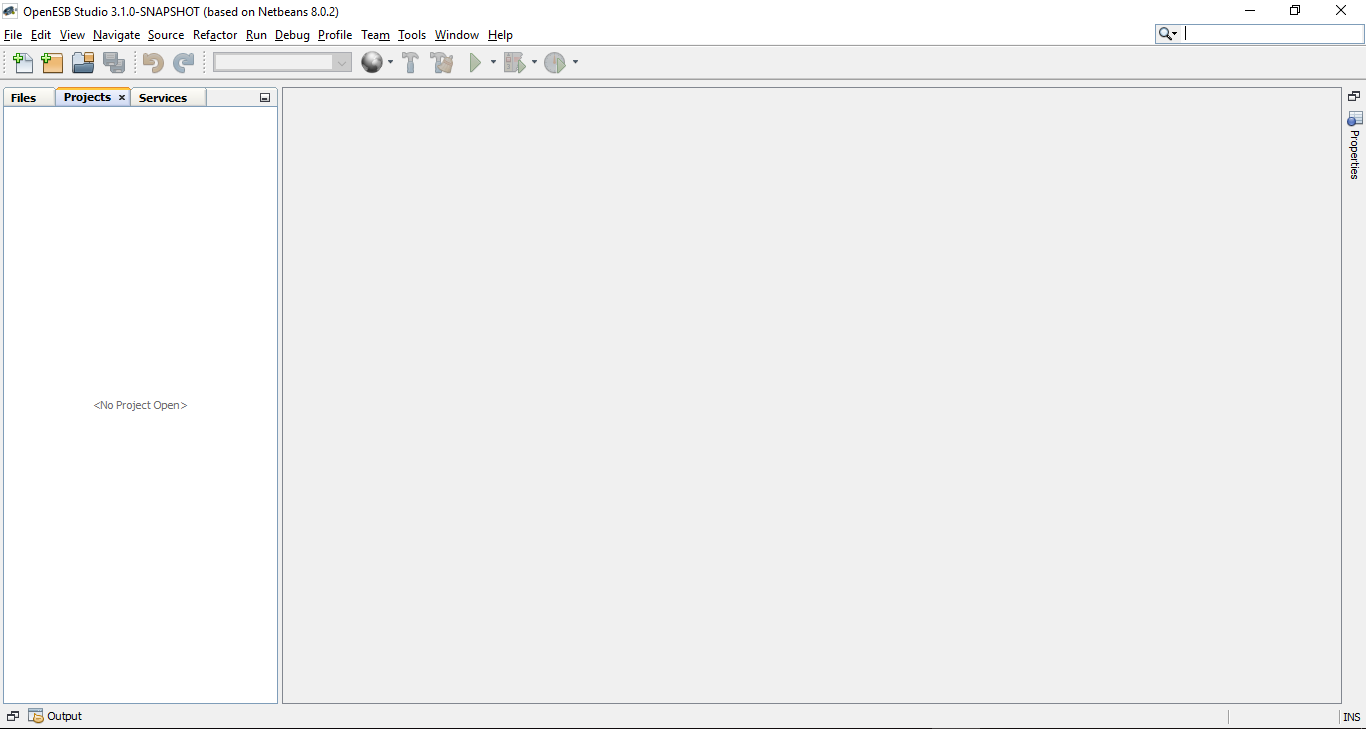
Command: GET /newWSDLService/newWSDLPort?wsdl HTTP/1.0



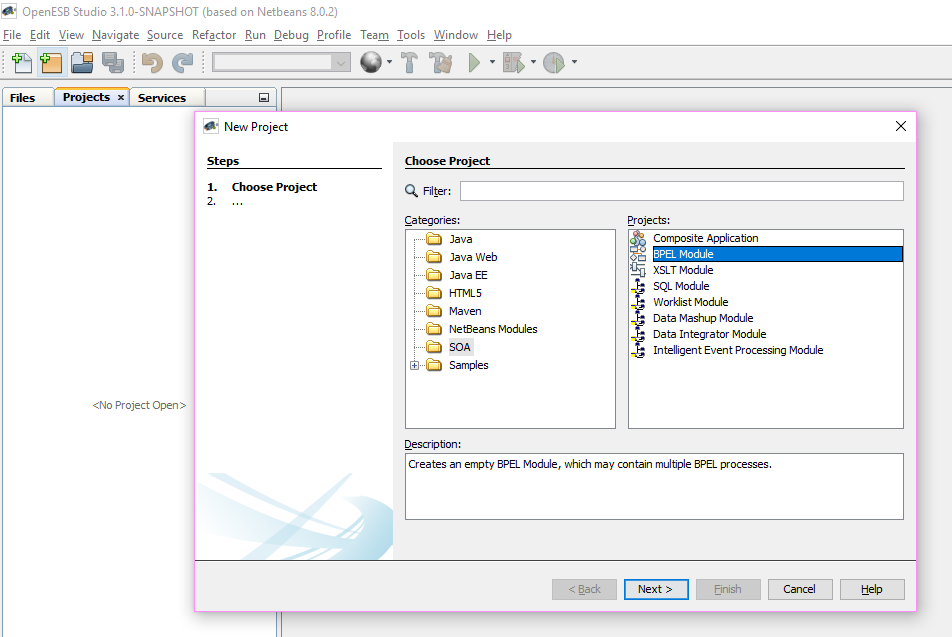
# OpenESB Project Creation

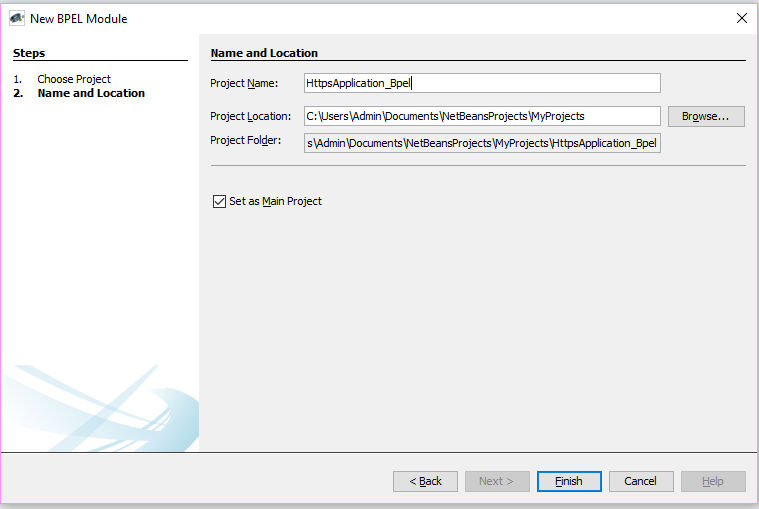
Create sample HTTPs based project using OpenESB 3.0.5 version. Follow below steps to develop the same.

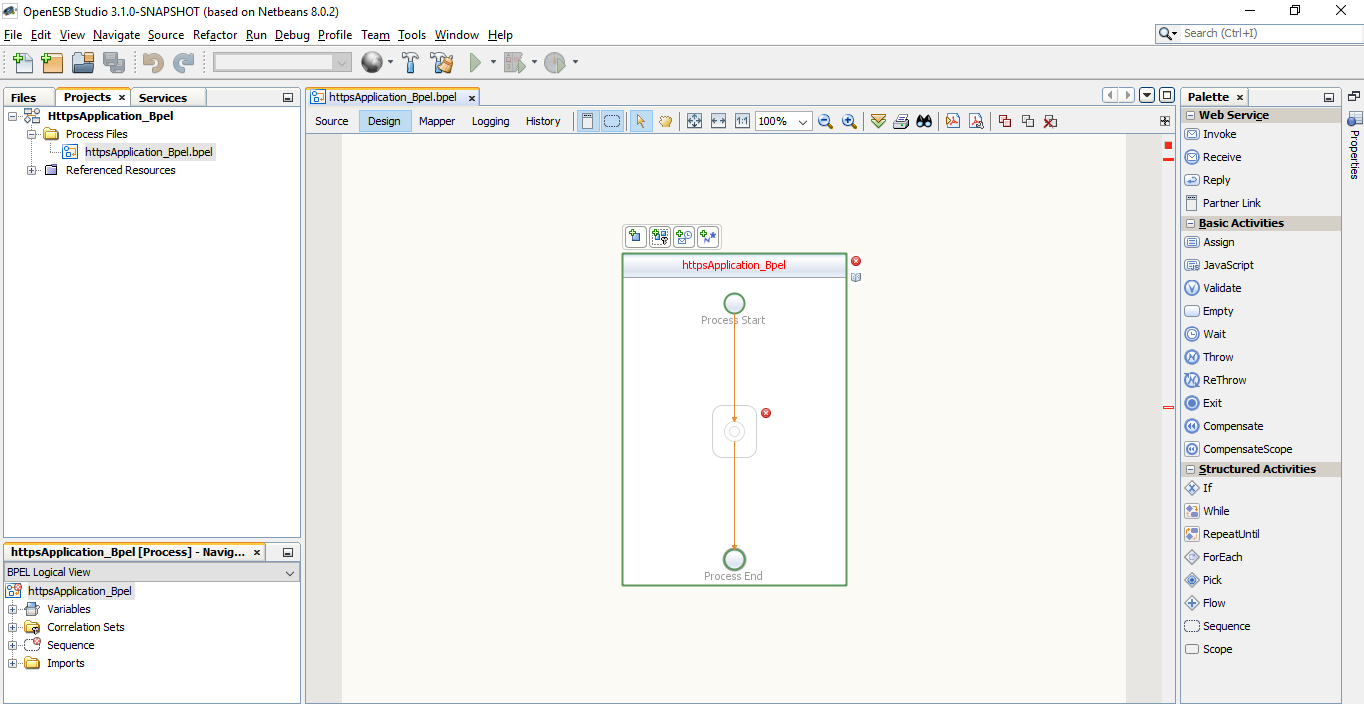
1. Start OpenESB NetBeans IDE.



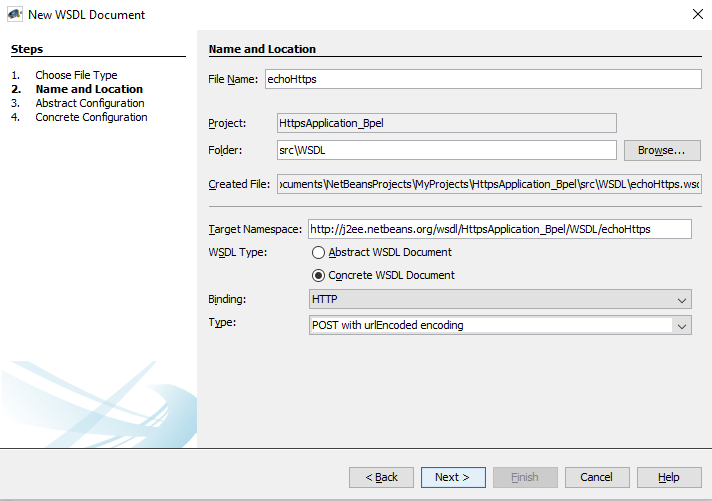
1. Create sample BPEL project and name as HttpsApplication\_Bpel.

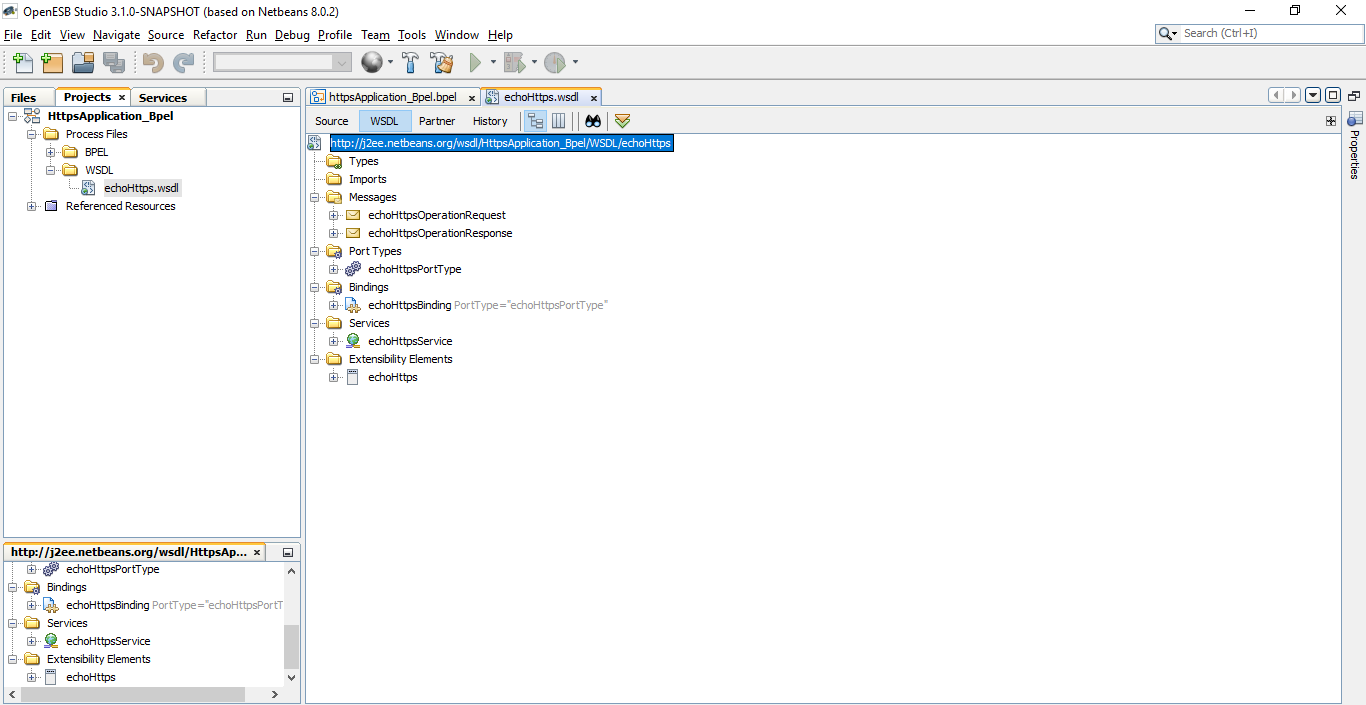




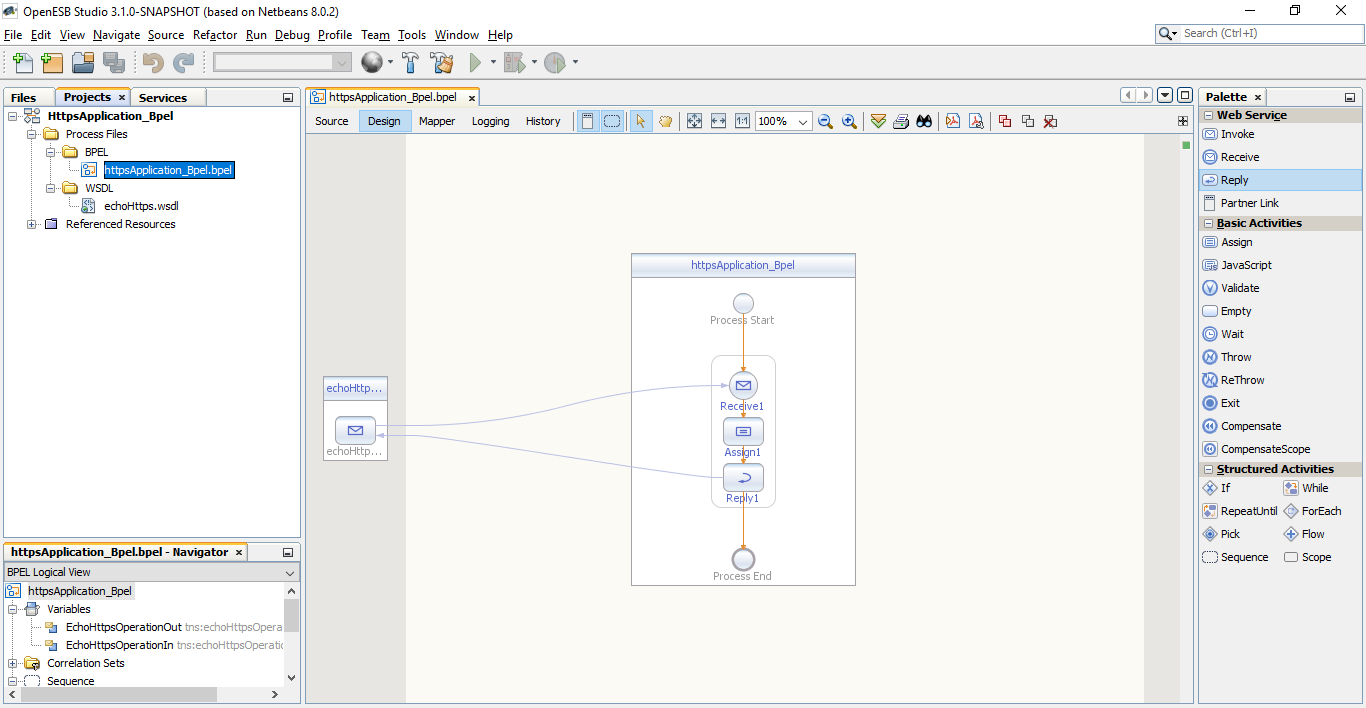


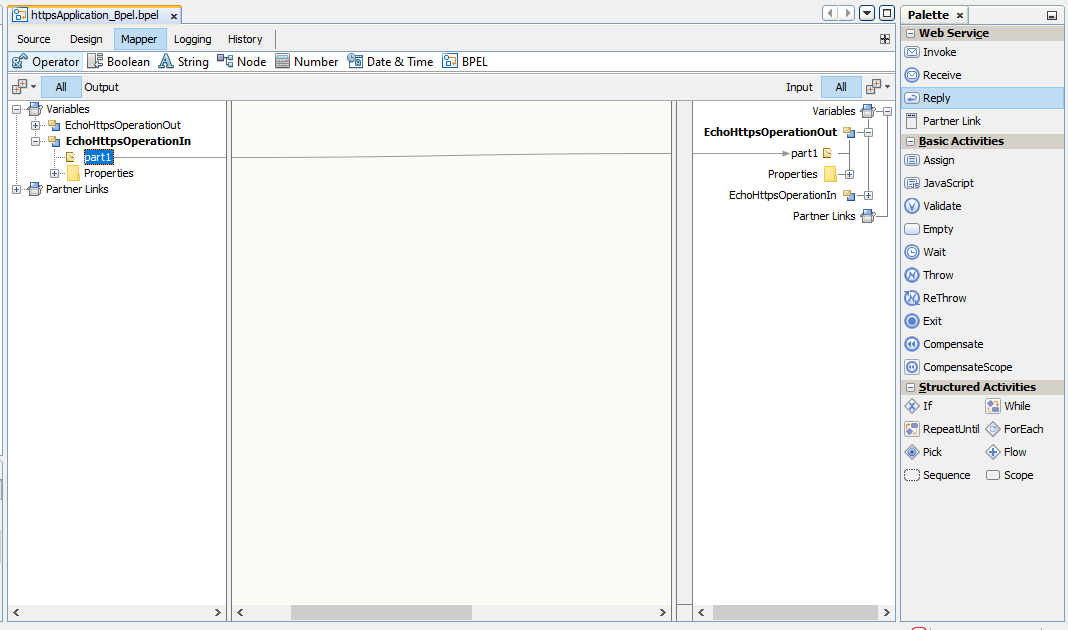
1. Create new Http based WSDL document with name echoHttps.



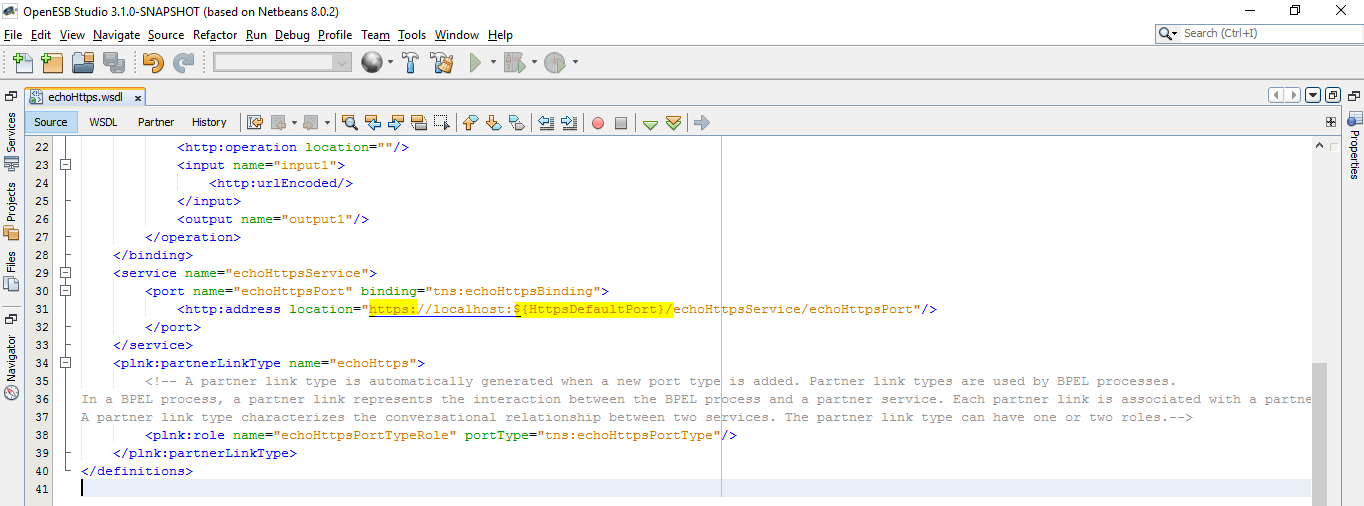


1. Create sample mapping in BPEL process.

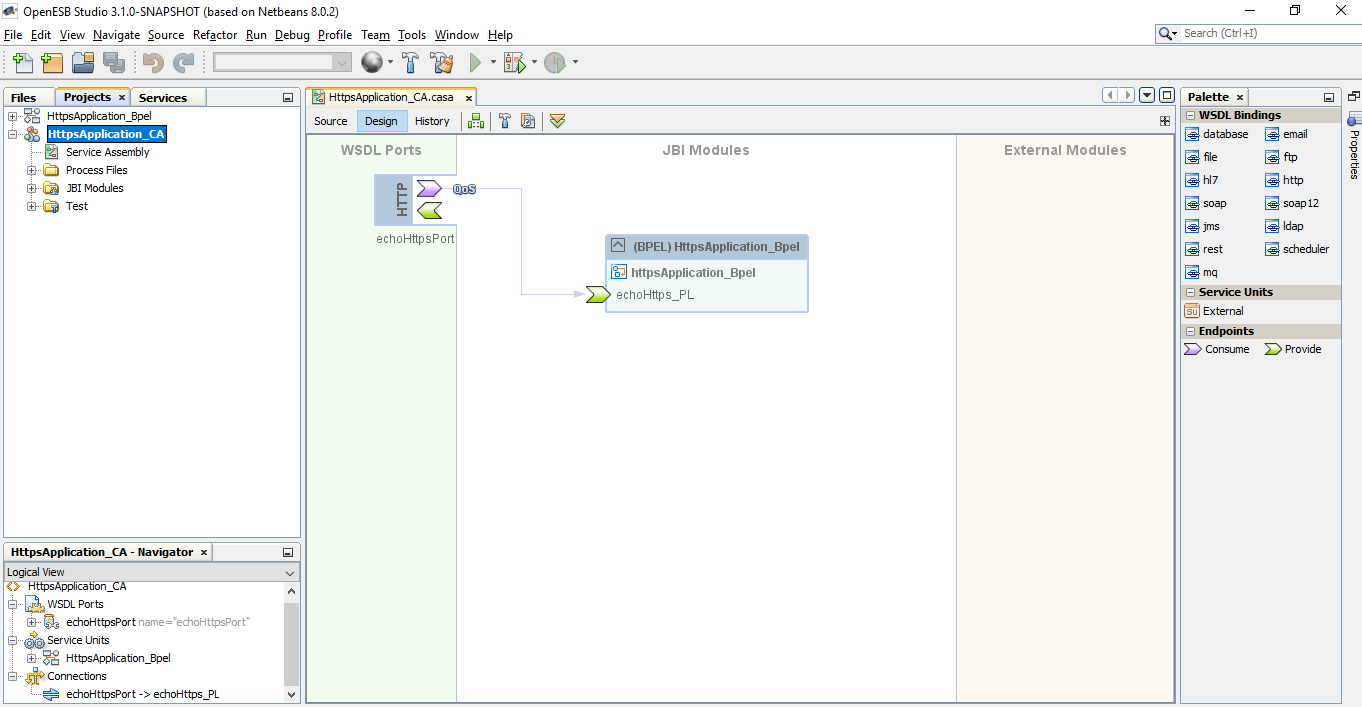




1. Update the WSDL document echoHttps.wsdl with Https mode.



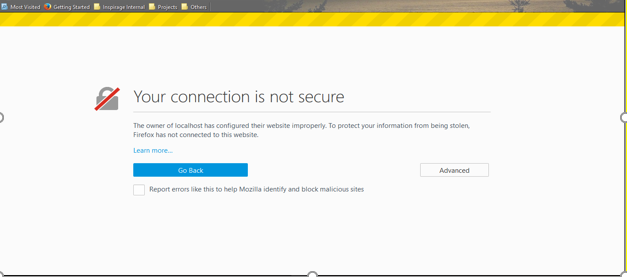
1. Save the project and build it.
2. Create Composite application using above Bpel project.

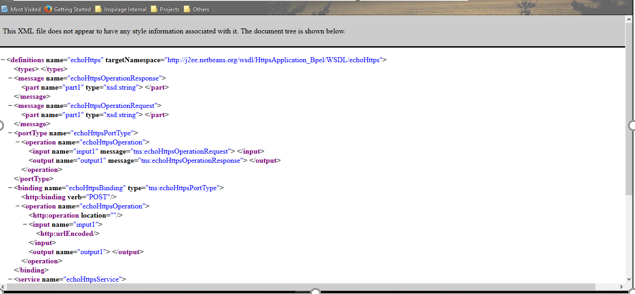


1. Finally, deploy the application.
2. Use below URL to test on browser and check whether you are able to access your application with Https mode.

URL: [https://localhost:${HttpsDefaultPort}/echoHttpsService/echoHttpsPort?wsdl](https://localhost:$%7bHttpsDefaultPort%7d/echoHttpsService/echoHttpsPort?wsdl)

While accessing the URL for the first time, you will receive warning message regarding your certificate because we are using self-signed SSL. Just add the exception in browser and simply go ahead.





This ensures, complete setup of SSL is successful.