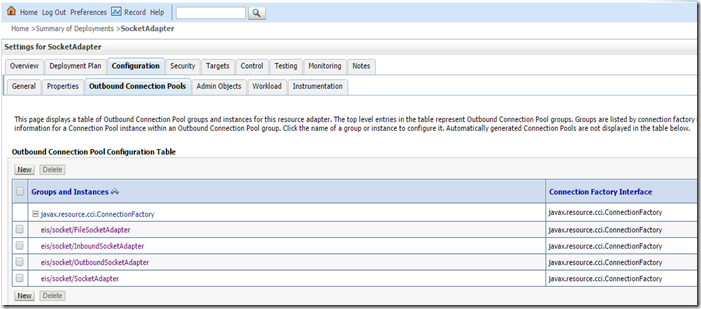
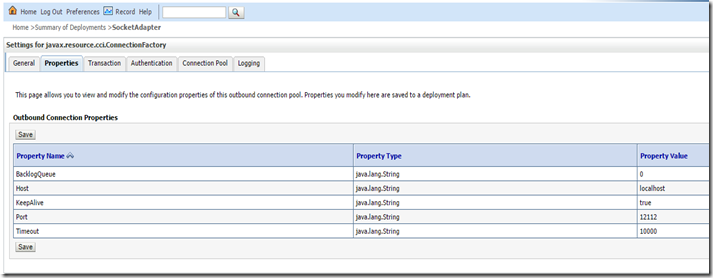
**Demystifying Oracle Socket Adapter**

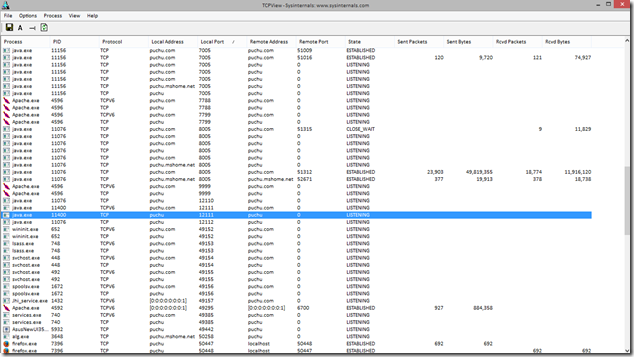
Spent some time today to understand how socket adapter works , how to test composite as it can’t be done from EM for inbound socket adapter. Lets assume the below use case, we have fixed length separated file , say Employee Details coming in source over some TCP port say 12112. SOA composite will bind the incoming file using NXSD and accept the stream data , then it will perform certain operation and return ACK to same socket port with some predefined code. Below are the detailed implementation steps,

1. At first create a new socket adapter outbound connection pool from weblogic console.

[](https://lh6.ggpht.com/-sHrNpN0sQmY/VHH2DIro6qI/AAAAAAAAD1o/PczjipWnqj0/s1600-h/image%25255B4%25255D.png)

[](https://lh3.ggpht.com/-BIX7FA5OLB4/VHH2E2uiOaI/AAAAAAAAD14/cirxWehphxE/s1600-h/image%25255B10%25255D.png)

Give the port no which should be free in your system, to check at your windows PC you can issue netstat -an | find "12112"  from command-prompt, else you can download TCPView from <http://technet.microsoft.com/en-in/sysinternals/bb897437.aspx>, it has good gui.

[](https://lh5.ggpht.com/-gBQyHvJv7fE/VHH2GO47acI/AAAAAAAAD2I/REuTwymAwH0/s1600-h/image%25255B32%25255D.png)

By default KeepAlive property is set as false, you can change the same to true for better performance.

2. Next create a emp.dat file like below ,

fn111111ln111111232007-01-01100  
fn211111ln211111232007-11-01200  
fn311111ln311111232007-12-01300

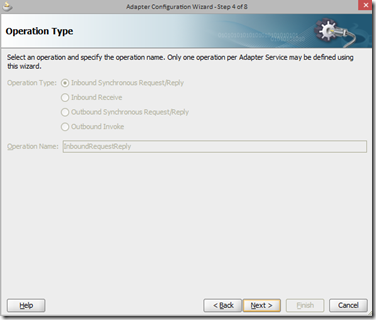
And a fixed length NXSD file to read the same,

<?xml version="1.0" encoding="UTF-8" ?>  
<xsd:schema xmlns:xsd="[http://www.w3.org/2001/XMLSchema"](http://www.w3.org/2001/XMLSchema%22)  
            xmlns:nxsd="[http://xmlns.oracle.com/pcbpel/nxsd"](http://xmlns.oracle.com/pcbpel/nxsd%22)  
            xmlns:tns="[http://shrikworld.blogspot.in/ReadEmpDetails"](http://shrikworld.blogspot.in/ReadEmpDetails%22)  
            targetNamespace="[http://shrikworld.blogspot.in/ReadEmpDetails"](http://shrikworld.blogspot.in/ReadEmpDetails%22)  
            elementFormDefault="qualified" attributeFormDefault="unqualified"  
            nxsd:version="NXSD" nxsd:stream="chars" nxsd:encoding="US-ASCII">  
  <xsd:element name="EmpDetails">  
    <xsd:complexType>  
      <xsd:sequence>  
        <xsd:element name="Emp" minOccurs="1" maxOccurs="unbounded" nxsd:style="array" nxsd:cellSeparatedBy="${eol}">  
          <xsd:complexType>  
            <xsd:sequence>  
              <xsd:element name="FirstName" type="xsd:string" nxsd:style="fixedLength" nxsd:length="8"/>  
              <xsd:element name="LastName" type="xsd:string" nxsd:style="fixedLength" nxsd:length="8"/>  
              <xsd:element name="Age" type="xsd:int" nxsd:style="fixedLength" nxsd:length="2"/>  
              <xsd:element name="DOB" type="xsd:string" nxsd:style="fixedLength" nxsd:length="10"/>  
              <xsd:element name="Salary" type="xsd:int" nxsd:style="fixedLength" nxsd:length="3"/>  
            </xsd:sequence>  
          </xsd:complexType>  
        </xsd:element>  
      </xsd:sequence>  
    </xsd:complexType>  
  </xsd:element>  
</xsd:schema>

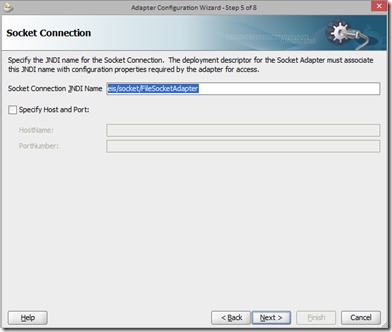
Create another XSD file for sync response to socket request,

<?xml version="1.0" encoding="UTF-8" ?>  
<xsd:schema xmlns:xsd="[http://www.w3.org/2001/XMLSchema"](http://www.w3.org/2001/XMLSchema%22)  
            xmlns="[http://shrikworld.blogspot.in/response"](http://shrikworld.blogspot.in/response%22)  
            targetNamespace="[http://shrikworld.blogspot.in/response"](http://shrikworld.blogspot.in/response%22)  
            elementFormDefault="qualified">  
  <xsd:element name="Response">  
    <xsd:annotation>  
      <xsd:documentation>  
        A sample element  
      </xsd:documentation>  
    </xsd:annotation>  
    <xsd:complexType>  
      <xsd:sequence>  
        <xsd:element name="Code" type="xsd:string"/>  
      </xsd:sequence>  
    </xsd:complexType>  
  </xsd:element>  
</xsd:schema>

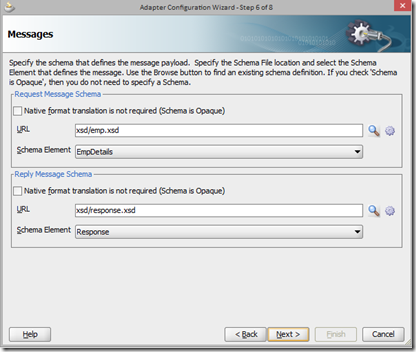
3. Now create a empty composite and drag-drop an inbound socket adapter, select Inbound Sync Req/Response from operation ,

[](https://lh6.ggpht.com/-syeWF7wocEI/VHH2Hk7eP-I/AAAAAAAAD2Y/YuFmUfGH-9c/s1600-h/image%25255B19%25255D.png)

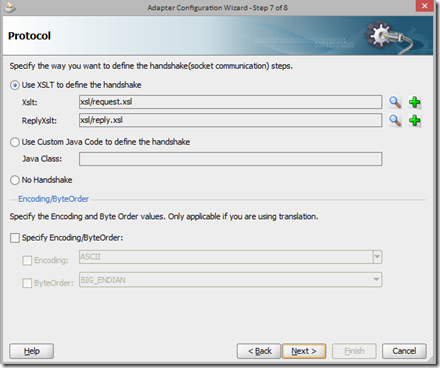
Select the JNDI you created earlier at step#1.

[](https://lh5.ggpht.com/-zYL-Ccgykec/VHH2JApL6BI/AAAAAAAAD2o/l7ugwitBq6w/s1600-h/image%25255B20%25255D.png)

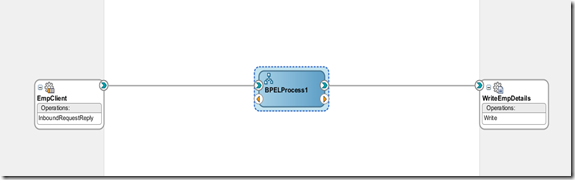
Select request and response XSD that you created step#2,

[](https://lh4.ggpht.com/-HID5f4h6apY/VHH2KnfEdII/AAAAAAAAD24/_2ct86NNIOY/s1600-h/image%25255B24%25255D.png)

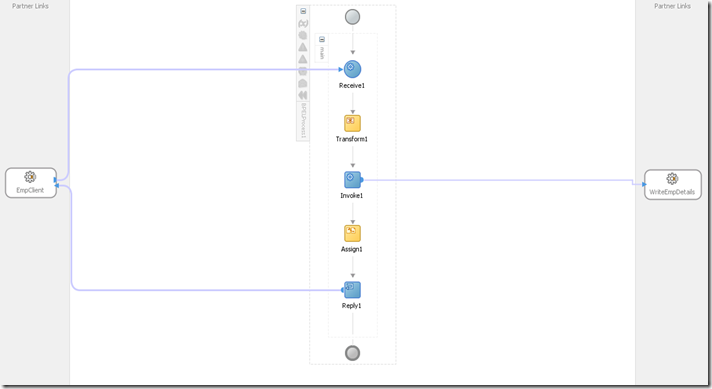
Create request/response XSL for handshake like below and finish.

[](https://lh5.ggpht.com/-vcAVw2L3I4c/VHH2MN7fIaI/AAAAAAAAD3I/nd1n3BVyfyk/s1600-h/image%25255B28%25255D.png)

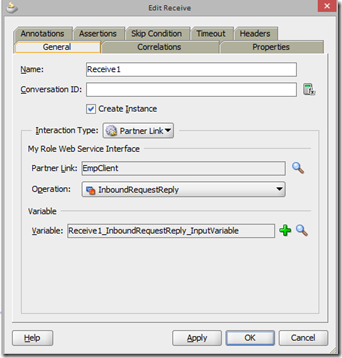
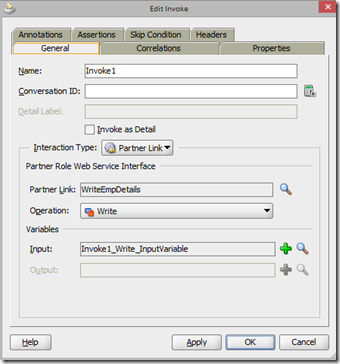
Now create a outbound file adapter to write the input data in the file.Drag and drop a BPEL component defining interface later.

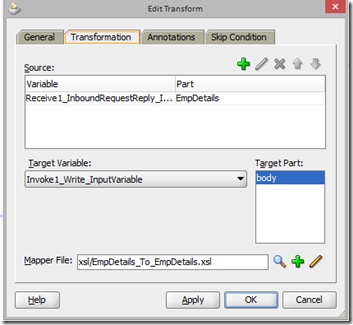
[](https://lh4.ggpht.com/-jmy6U3z9mKU/VHH2NaI9e9I/AAAAAAAAD3Y/Z6Bilf7bbKw/s1600-h/image%25255B37%25255D.png)

Now go to the BPEL process and design like below,

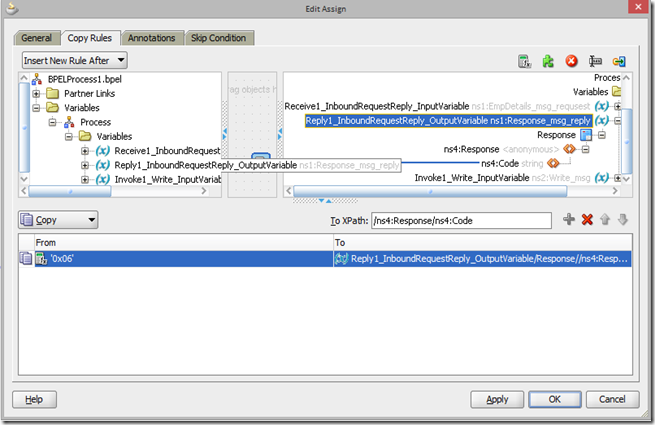
[](https://lh5.ggpht.com/-JMYfpvnu-3U/VHH2P7VsmvI/AAAAAAAAD3o/hcUjYcD8-XQ/s1600-h/image%25255B41%25255D.png)

It’s just any other BPEL process, screenshot of different activity,

[](https://lh6.ggpht.com/-8Af06Jh-v3c/VHH2RqCS9QI/AAAAAAAAD34/I5-Vxn0hj10/s1600-h/image%25255B52%25255D.png) [](https://lh4.ggpht.com/-75rXL83111A/VHH2TGluSaI/AAAAAAAAD4I/QKVxGFAO38M/s1600-h/image%25255B54%25255D.png)

[](https://lh5.ggpht.com/-rcmIw_BQKpo/VHH2USi-hvI/AAAAAAAAD4Y/4kTjKbz7qvA/s1600-h/image%25255B58%25255D.png)

Notice the last assign activity, here I’m setting ‘0x06’ to the response code, you can set any value.

[](https://lh5.ggpht.com/-_VejG_WGorQ/VHH2V9QAs8I/AAAAAAAAD4o/8iqC_OQTmTQ/s1600-h/image%25255B63%25255D.png)

4. Next step is important, here we’ll create request.xsl and reply.xsl.

There is a function socketReadWithXlation with that input data will be read from port and will be translated using the input message NXSD, so our request.xsl file look likes,

<xsl:template match="/">  
    <xsl:copy-of select="socket:socketReadWithXlation()" />  
  </xsl:template>

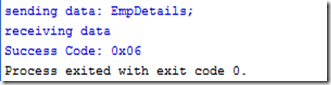
In sync response.xsl we need the response code that we set from BPEL process on step#3.We are going to use socketWrite function like below in the xslt,

<xsl:template match="//ns0:Response">  
  <xsl:variable name="temp">  
            <xsl:value-of select="/ns0:Response/ns0:Code"/>  
        </xsl:variable>  
    <xsl:variable name="var1" select="socket:socketWrite($temp, '','')"/>  
    <xsl:variable name="var2" select="socket:socketEndOutput()"/>  
  </xsl:template>

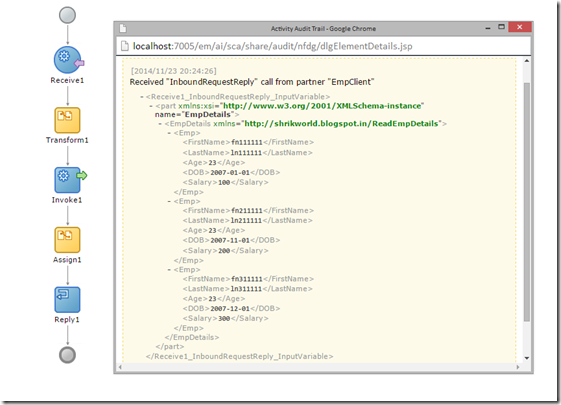
5. Now its time to test. First deploy the composite and you’ll notice that it can’t be tested from EM. You need a client program to push the data to the socket.

public class EmpClient {  
    public static void main(String[] args) {  
        try {  
            Socket socket;  
            final String HOST = "localhost";  
            final int PORT = 12112;  
            try {  
                socket = new Socket(HOST, PORT);  
            } catch (IOException ioe) {  
                System.out.println(">>>");  
                System.out.println(ioe.getMessage());  
                System.out.println(">>>");  
                throw ioe;  
            }  
            System.out.println("sending data: EmpDetails;");  
            OutputStream os = socket.getOutputStream();  
            byte[] b = "**fn111111ln111111232007-01-01100\nfn211111ln211111232007-11-01200\nfn311111ln311111232007-12-01300**".getBytes();  
            for (int i = 0; i < b.length; i++) {  
                os.write(b[i]);  
            }  
            os.flush();  
            socket.shutdownOutput();  
            System.out.println("receiving data");  
            BufferedReader soc\_in = new BufferedReader(new InputStreamReader(socket.getInputStream()));  
            String successCode = soc\_in.readLine();  
            System.out.println("Success Code: " + successCode);  
            socket.close();  
        } catch (IOException ioe) {  
            ioe.printStackTrace();  
        }  
    }  
}

You might noticed I’m sending the same data from Java code with \n separator.When you run this program you should get below output,

[](https://lh3.ggpht.com/-iCjgNaIrT2U/VHH2XOit3sI/AAAAAAAAD44/DXLzTQcR_xQ/s1600-h/image%25255B67%25255D.png)

Now check the flow from EM console,

[](https://lh3.ggpht.com/-oetZuuLfbKg/VHH2YhZfHnI/AAAAAAAAD5I/mpWErnDIbho/s1600-h/image%25255B71%25255D.png)