**Migration of BPEL 10g to 11g**

1. Load the BPEL project into JDeveloper 10g and compile it.
2. If it compiles successfully then open it in JDeveloper 11g otherwise remove the external references then compile again until you get the 0 errors.
3. Open the project in JDeveloper 11g.
4. Select Refactor from menu and rename the project if you want ()
5. Move the xsd and xsl files into corresponding folders in 11g.
6. In each .wsdl file change the schema Location prefixed with ‘xsd/’.

Ex: schemaLocation="xyz.xsd" to schemaLocation="xsd/xyz.xsd".

1. Change the project directory name and change the bpel, componentType wsdl file names.
2. Change the project and BPLE component in all the files (Check component type file).
3. Change each file with the wsdl, jca and xsd files (names, paths, links etc.).
4. Remove the callbackinterface link from the composite.xml so that you won’t get the multiple operations in Composite.xml

**Oracle SOA 11g – BPEL 1.1 to BPEL 2.0 Upgrade**

In my current project we were using BPEL 1.1 processes. Then SOA Suite 11gR1PS3 came along and gave use BPEL 2.0. So, we decided to start upgrading our already build BPEL 1.1 processes to version 2.0 for making it a standard implementation considering future enhancements.

As there are no BPEL 2.0 upgrade tools we had basically two options. The first was to complete rebuild all our processes. Not a very good option to redo all the work. The second option was to manually upgrade the BPEL 1.1 definitions to 2.0. Expecting this to be less work we choose the second option.

**Step by Step Process:-**

1. Open the 11g migrated Composite in the JDeveloper and goto BPEL source File,  And then change the BPEL namespace just like below:

                xmlns=”<http://schemas.xmlsoap.org/ws/2003/03/business-process/>”

**into**

xmlns=”<http://docs.oasis-open.org/wsbpel/2.0/process/executable>”

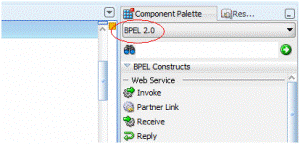
            Then add the below namespace after the xmlns,

             xmlns:bpel="http://docs.oasis-open.org/wsbpel/2.0/process/executable"

1. In the Composite.xml, open the source and add the composite version for the BPEL component as below:

                <component name="bpelprocessname" version="2.0">

1. Finally close JDeveloper, delete the SCA-INF directory and restart JDeveloper. The SCA-INF was created again. When opened the bpel process in the editor it was recognized as a BPEL 2.0 process

[](https://3.bp.blogspot.com/-DJ00ARiRhFI/TlU40SKXZwI/AAAAAAAACh0/Gtdz2hbEXj0/s1600/image002.gif)

Now, when you rebuild, you can see lot of errors.

1. **Assign Activity (Xpath to XSL):-**

The assignment (assign activity) has a major shift in BPEL 2.0. In BPEL 1.1 the assignments were based on XPath but in BPEL 2.0 the assignments are based on XSL notation. Change all the Assign activities like below either changing the Source file (if you are comfortable with XSL & XPATH) or using the BPEL designer.

**BPEL 1.0**

<assign name="Assign1">

<copy>

<from expression="bpws:getVariableData('Var1') + 5000"/>

<to variable="Var2"/>

</copy>

<copy>

<from expression="bpws:getVariableData('Iterator') + 5000"/>

<to variable="Input\_Xyz\_Service" part="Xyz\_ServiceInput\_msg" query="/ns6:Xyz\_ServiceInput/ns1:num"/>

</copy>

<copy>

<from expression="bpws:getVariableData('Xyz\_Output','OutputParameters','/ns1:OutputParameters/ns1:P\_TO')"/>

<to variable="Xyz\_req" part="XyzPayload" query="/XyzPayload/ns1:To"/>

</copy>

</assign>

**BPEL 2.0**

<assign name="Assign1">

<copy>

<from>$Var1 + 5000"</from>

<to>$Var2</to>

</copy>

<copy>

<from>$Iterator + 5000</from>

<to>$Input\_Xyz\_Service.Xyz\_ServiceInput\_msg/ns1:num</to>

</copy>

<copy>

<from>$Xyz\_Output.OutputParameters/ns1:P\_TO</from>

<to>$Xyz\_req.XyzPayload/ns2:to</to>

</copy>

</assign>

**XSL Transformation Mapping**

**BPEL 1.1**

<assign name="Transform\_XXX\_Service">

<bpelx:annotation>

<bpelx:pattern>transformation</bpelx:pattern>

</bpelx:annotation>

<copy>

<from expression="ora:processXSLT('Transformation1.xsl',bpws:getVariableData('Output\_Xyz\_Service','Xyz\_ServiceOutputCollection'))"/>

<to variable="Input\_Xyz\_Service" part="XyzCollection"/>

</copy>

</assign>

**BPEL 2.0**

<assign name="Transform\_XXX\_Service">

<bpelx:annotation>

<bpelx:pattern patternName="bpelx:transformation"/>

</bpelx:annotation>

<copy>

<from>ora:doXSLTransformForDoc("xsl/Transformation1.xsl", $Output\_Xyz\_Service.Xyz\_ServiceOutputCollection)</from>

<to variable="Input\_Xyz\_Service" part="XyzCollection"/>

</copy>

</assign>

**While Loop**

**BPEL 1.1**

<while name="While\_1"

condition="bpws:getVariableData('Variable') &lt;= bpws:getVariableData('Ouput\_Xyz\_Service','Xyz\_Output','/ns1:Xyz\_Output/ns1:Count')">

<sequence name="Sequence\_1">

</sequence>

</while>

**BPEL 2.0**

<while name="While\_1">

<condition>$Variable &lt;= $Ouput\_Xyz\_Service.Xyz\_Output/ns1:Count</condition>

<sequence name="Sequence">

</sequence>

</while>

**Case Statement**

**BPEL 1.0**

<switch name="switch-1">

<case condition="bpws:getVariableData('XyzIn','payload', '/xyz:Mno/xyz:cnt') < 10">

<assign name="abc">

</assign>

</case>

<case condition="bpws:getVariableData('XyzOut','payload', '/xyz:Mno/xyz:cnt') > 10">

<assign name="def">

</assign>

</case>

<otherwise>

<assign name="mno">

</assign>

</otherwise>

</switch>

**BPEL 2.0**

<if name="switch-1">

<condition>$XyzIn/xyz:Mno/xyz:cnt < 10</condition>

<assign name="abc">

</assign>

<elseif>

<condition>$XyzOut/xyz:Mno/xyz:cnt < 10</condition>

<assign name="def">

</assign>

</elseif>

<else>

<assign name="mno">

</assign>

</else>

</if>

**Changes in the XSL File**

**BPEL 1.0**

<?xml version="1.0" encoding="UTF-8" ?>

<?oracle-xsl-mapper

<mapSources>

<source type="XSD">

<schema location="../../../bpel/ref\_DB\_Xyz.xsd"/>

<rootElement name="Xyz\_ServiceOutputCollection" namespace="http://xmlns.oracle.com/pcbpel/adapter/db/ref\_DB\_Xyz"/>

</source>

</mapSources>

<mapTargets>

<target type="XSD">

<schema location="../../../bpel/ref\_DB\_Mno.xsd"/>

<rootElement name="XyzCollection" namespace="http://xmlns.oracle.com/pcbpel/adapter/db/top/ref\_DB\_Mno"/>

</target>

</mapTargets>

?>

<xsl:stylesheet version="1.0"

yService/xpath"

xmlns:hwf="http://xmlns.oracle.com/bpel/workflow/xpath"

exclude-result-prefixes="xsl ns0 xs ns1 xref xp20 bpws ora ehdr orcl ids hwf">

<xsl:template match="/">

<ns1:XyzCollection>

<xsl:for-each select="/ns0:XyzServiceOutputCollection/ns0:Xyz\_ServiceOutput">

<ns1:Xyz>

<ns1:xyz>

<xsl:value-of select="ns0:xyz"/>

</ns1:xyz>

<ns1:userKey>

<xsl:value-of select="ns0:KEY"/>

</ns1:userKey>

</ns1:Xyz>

</xsl:for-each>

</ns1:XyzCollection>

</xsl:template>

</xsl:stylesheet>

**BPEL 2.0**

<?xml version="1.0" encoding="UTF-8" ?>

<?oracle-xsl-mapper

<mapSources>

<source type="WSDL">

<schema location="../ref\_DB\_Xyz.wsdl"/>

<rootElement name="Xyz\_ServiceOutputCollection" namespace="http://xmlns.oracle.com/pcbpel/adapter/db/ref\_DB\_Xyz"/>

</source>

</mapSources>

<mapTargets>

<target type="WSDL">

<schema location="../ref\_DB\_Mno.wsdl"/>

<rootElement name="XyzCollection" namespace="http://xmlns.oracle.com/pcbpel/adapter/db/top/ref\_DB\_Mno"/>

</target>

</mapTargets>

?>

<xsl:stylesheet version="1.0"

exclude-result-prefixes="xsi xsl plt ns0 xsd tns wsdl ns2 ns1 bpws xp20 bpel bpm ora socket mhdr oraext dvm hwf med ids xdk xref ldap">

<xsl:template match="/">

<ns1:XyzCollection>

<xsl:for-each select="/ns0:XyzServiceOutputCollection/ns0:Xyz\_ServiceOutput">

<ns1:Xyz>

<ns1:xyz>

<xsl:value-of select="ns0:xyz"/>

</ns1:xyz>

<ns1:userKey>

<xsl:value-of select="ns0:KEY"/>

</ns1:userKey>

</ns1:Xyz>

</xsl:for-each>

</ns1:XyzCollection>

</xsl:template>

</xsl:stylesheet>

1. **JAVA embedding (bpelx:exec & import):-**

Sometimes, in our BPEL we import java classes to utilize those functions on the java embedding object.  And also for regular java embedding activity, enclose the Embedded Java activity element (bpelx:exec) by the extensionActicity element and removed the version attribute.

**BPEL 1.1 java exec import**:-

                <bpelx:exec import="org.w3c.dom.Element"/>

                <bpelx:exec name="evaluateCriteria" language="java" version="1.3">

                                        <![CDATA[// initialize ]]>

                </bpelx:exec>

**BPEL 2.0 java exec** **import** is changed to a process level import using the java extension:- This import should be specified at the top of the BPEL, just after the “<process>” element

            <import importType="http://schemas.oracle.com/bpel/extension/java" location="org.w3c.dom.Element"/>

                                <extensionActivity>

                                    <bpelx:exec name="evaluateCriteria" language="java">

                                        <![CDATA[// initialize ]]>

                                    </bpelx:exec>

                                    </extensionActivity>

1. The **checkpoint** element can be replaced by the dehydrate element also enclosed by an extensionActicity element as it is a oracle specific bpel 2.0 extension like the embedded java activity.
2. The **terminate** activity just needs to be renamed to exit.
3. Need to add an additional attribute to the **catch** element when you are catching faults. This can be a faultMessageType or faultElement. Open the catch fault editor and select the right one.
4. The **until** attribute of the BPEL 1.1 **Pick/onAlarm** element has become a child element in 2.0 and needed to be moved.
5. Not all annotations **<bpelx:annotation>** are supported anymore. Comment some of them out for now as they were mainly for documentation purposes only.
6. Unfortunately the **skipCondition** attribute is no longer supported.  Its a little disappointment about this although it’s understood why it was removed. This attribute is just not part of the BPEL 2.0 standard. So, we have to add if/then/else constructions to mimic the same.
7. **BPEL 2.0 refactoring:-**

Since the BPEL process is upgraded to 2.0 from 1.1, we have some refactoring options also which would be worth doing it.

         In BPEL 2.0 there is the notation of **variable initialization**. Where in BPEL 1.1 we needed an additional assign activity to initialize a variable, in 2.0 it can be done as part of the variable definition. So we can make the BPEL process cleaner by moving the initialization to the definition part.

         The assign activity has some additional functionality: **keepSrcElementName** (in BPEL 2.0 projects only): Select this option to toggle the keepSrcElementName attribute on the copy rule on and off. This option enables to replace the element name of the destination (as selected by the to-spec) with the element name of the source.

         **Change Rule Type** (in BPEL 2.0 projects only): Select this option to change the type of the selected rule to one of the BPEL extension rules: bpelx:copyList, bpelx:insertAfter, bpelx:insertBefore, or bpelx:append.