Oracle BPEL Tutorial

This exercise introduces you to the Business Process Execution (BPEL) language, the Oracle JDeveloper BPEL Designer and to the Oracle BPEL Process Manager engine.

INSTALL JDEVELOPER (if working on your machine)

- Download Oracle JDeveloper version 10.1.3.4 (Studio Edition) (http://www.oracle.com/technology/software/products/jdev/htdocs/soft10134.html)
- **Note**: if you already have a JDK 6 installed in your machine, you only need to download the "Base Install" of JDeveloper, which is considerably smaller than the full install. If you download the base Studio install, you need to unzip it, launch the jdeveloper executable, and follow the prompt to link it to your existing JDK 6 installation.
- Install and start JDeveloper.

CREATE A CONNECTION TO THE ORACLE APPLICATION SERVER

Before starting to create BPEL processes, we need to configure JDeveloper to access the BPEL server where we can deploy and test our processes.

1. In Oracle JDeveloper, select View > Connection Navigator.
2. Right-click the Application Server node and select New Application Server Connection.
3. Click Next on the Welcome page.
4. In Step 1, Type, enter the following values:
   - Connection Name: enter **BPELServer**
   - Connection Type: select **Standalone OC4J 10g 10.1.3**.

   Click **Next**.

5. In Step 2, Authentication, enter the following values:
   - Username: enter **oc4jadmin**. This is the name of the administration user.
   - Password: enter the password that was sent by e-mail.
   - Deploy Password: select the check box.

   Click **Next**.

6. In Step 3, Connection, enter the following values:
   - Connect To: select Single Instance.
   - Host Name: enter **serverName.at.mt.ut.ee** (the server name was sent by e-mail)
   - RMI Port: enter **23791**
   - URL Path: leave blank

   Click **Next**.

7. In Step 4, Test, click **Test Connection**. If the test is not successful, check that the Oracle Application Server instance is available and that the connection values are correct. You can click the Back button to return to the previous page to edit the connection values.
8. Click **Finish**. The connection appears below the Application Server node in the Connection Navigator.

CREATE A CONNECTION TO THE ORACLE INTEGRATION SERVER

In JDeveloper, create a connection to the BPEL server running on the Oracle Application Server instance.

1. In JDeveloper, select View > Connection Navigator.
2. Right-click the Integration Server node and select New Integration Server Connection.
3. Click Next on the Welcome page.
4. In Step 1, Name, enter the following value:
    - Connection Name: enter **BPELIntegConnection**

    Click **Next**.

5. In Step 2, Connection, enter the following values:
    - Application Server: enter **BPELServer**, which is the Application Server connection that you created previously
    - Hostname: The value is derived from the Application Server connection.
    - Port Number: enter **9700**
    - Add Hostname to the List of Proxy Exceptions: select this option.

    Click **Next**.

6. In Step 3, Test Connection, click **Test Connection**. Don't worry if the ESB Server has FAILED – we don't need this component. On the other hand, if the Application Server or the BPEL Process Manager Server test is not successful, check that the Oracle Application Server instance is available and that the connection values are correct. You can click the Back button to return to the previous page to edit the connection values.
7. Click **Finish**. The connection appears below the Integration Server node in the Connection Navigator.

CREATE THE CREDIT FLOW BPEL PROCESS

You will be guided throughout the creation of an asynchronous process-centric service in BPEL which synchronously communicate with a Credit rating service.

TEST THE CREDIT RATING SERVICE

- Connect to the BPEL Console (at http://serverName.at.mt.ut.ee:9700/BPELConsole) and log in (username: "bpeladmin", password: was sent by e-mail).
- In the **Name** column (under the **Dashboard** tab of the console), click the link for the `CreditRatingService` BPEL process.
- In the test form that appears, enter any nine-digit number as the social security number (for example, 123456789) and click **Post XML Message**. You should get back an integer credit rating (or a fault if the social security number you entered began with a 0). In either case, the service is confirmed to be installed successfully.

CREATE A NEW BPEL PROJECT

Now you will use the JDeveloper BPEL Designer's New Project wizard, which automatically generates the skeleton of a BPEL project: the BPEL source, a WSDL interface, a BPEL deployment descriptor, and an Ant script for compiling and deploying the BPEL process. To create your first BPEL project:

- First you need to create a new Application. This application will contain your BPEL projects.
- In the **Applications – Navigator** right click **Applications** and select **New Application.**
- In the wizard that appears select **BPELApplication** as name, click **OK**.
- In the next window that appears press **Cancel**.
- Now right click **BPELApplication** in the navigation menu and select **New Project.**
- In the wizard select **BPEL Process Project**
- Enter *yourLogin_CreditFlow* as the **BPEL Process Name**, where *yourLogin* is your university login. Since the BPEL server is shared among several students, please follow the naming conventions in order to avoid name conflicts.
- Leave the template as **Asyncronous BPEL Process** and the **Use Default Project Settings**, press **Finish**.

You can see (in the centre pane within BPEL Designer) that the New Project wizard has created a skeleton for a new asynchronous BPEL process, with all the necessary source files. The file names are the same whether you create a synchronous or an asynchronous process, but the contents of the `.bpel` and `.wsdl` files will differ as appropriate.

Note that JDeveloper is now showing 5 key panes, the centre pane displays the BPEL process itself. The Navigator on the left displays workspaces, projects and project source files. The Structure pane on the lower left shows further information about the currently selected file in the centre pane and there is a Message – Log on the bottom of the screen that displays the status of the project. The final

pane is the Component Palette on the right of screen that displays the process activities available to drag and drop into your BPEL process.

REVIEW THE WSDL INTERFACE OF YOUR ASYNCHRONOUS BPEL PROCESS

You will now review (and, in the next section, edit) the WSDL file for the process. To view the WSDL interface:

- In the Navigator, double-click the `yourLogin_CreditFlow.wsdl` file to edit it (contained in the folder Integration Content).
- Choose Source in bottom tab of the canvas to switch to the source view. Here scroll down as necessary to browse the code. The most "interesting" parts are pointed out below.

Notice that the New Project wizard has defined both a `yourLogin_CreditFlowRequest` complextype element that your flow accepts as input (in a document-literal style WSDL message) and a `yourLogin_CreditFlowResponse` element that it returns. This information is contained in the imported XML Schema file `yourLogin_CreditFlow.xsd`, located under folder Integration Content/Schemas of your Project.

Because this process is asynchronous, two portTypes are defined, each with a one-way operation: one to initiate the asynchronous process and one for the process to call back the client with the asynchronous response.

```
<!-- portType implemented by the CreditFlow BPEL process -->
<portType name=" yourLogin_CreditFlow">
  <operation name="initiate">
    <input message ="client:yourLogin_CreditFlowRequestMessage"/>
  </operation>
</portType>

<!-- portType implemented by the requester of CreditFlow BPEL process
for asynchronous callback purposes -->
<portType name="yourLogin_CreditFlowCallback">
  <operation name="onResult">
    <input message ="client:yourLogin_CreditFlowResponseMessage"/>
  </operation>
</portType>
```

Also note that the partnerLinkType defined for this asynchronous process has two roles, one for the service provider and one for the requester.

```
<plnk:partnerLinkType name="yourLogin_CreditFlow">
  <plnk:role name=" yourLogin_CreditFlowProvider">
    <plnk:portType name="client:yourLogin_CreditFlow"/>
  </plnk:role>
  <plnk:role name="yourLogin_CreditFlowRequester">
    <plnk:portType name="client:yourLogin_CreditFlowCallback"/>
  </plnk:role>
</plnk:partnerLinkType>
```

EDIT THE SCHEMA OF YOUR BPEL PROCESS

Now you will edit the XML Schema file `yourLogin_CreditFlow.xsd` to modify the input and output message of your BPEL process:

- Change the two type definitions so that your flow will take an `ssn` field as input (keeping the `string` type) and return a `creditRating` element as output of type `int`. The parts you need to change are shown in bold (and red) below.

```
<element name="yourLogin_CreditFlowProcessRequest">
  <complexType>
    <sequence>
      <element name="ssn" type="string">
    </sequence>
  </complexType>
</element>
<element name="yourLogin_CreditFlowProcessResponse">
  <complexType>
    <sequence>
      <element name="creditRating" type="int">
    </sequence>
  </complexType>
</element>
```

- Save your WSDL file and close the window in which you have been editing it.

SWITCH BETWEEN THE DIAGRAM VIEW AND SOURCE CODE

You will now turn to the BPEL file and view it in a variety of ways. To view the BPEL process:

- Double-click `yourLogin_CreditFlow.bpel` in the Navigator if it is not already open in the Designer (however, it should be already open if you have followed this tutorial exactly).
- The initial view that you will see for editing a BPEL process file consists of 3 swimlanes;
- the colored outer swim-lanes contain references to external services called Partnerlinks. The center swim-lane contains the CreditFlow process itself. Note that the BPEL Designer automatically puts a partnerLink named `client` on the left side of the window. This represents any process, service or interaction that initiates your CreditFlow service.

REVIEW THE BPEL SOURCE CODE

The New Project wizard has created a basic skeleton for you of an asynchronous BPEL process. To view the BPEL source code:

- With `yourLogin_CreditFlow.bpel` open and active, click the **Source** tab at the bottom of the `yourLogin_CreditFlow.bpel` window.
- Scroll down as necessary to browse the code. The interesting parts are pointed out below.

The partnerLink created for the client interface includes two roles, `myRole` and `partnerRole` assignment. As you saw in the WSDL file for this process, an asynchronous BPEL process typically has two roles for the client interface: one for the flow itself, which exposes an input operation, and one for the client, which will get called back asynchronously.

```
<partnerLinks>
<!-- comments... -->
  <partnerLink name="client"
    partnerLinkType="client:yourLogin_CreditFlow"
    myRole="yourLogin_CreditFlowProvider"
    partnerRole="yourLogin_CreditFlowRequester"
  />
</partnerLinks>
```

Also, the `<receive>` activity in the main body of the process is followed by an `<invoke>` activity to perform an asynchronous callback to the requester. (Note the difference between this and a synchronous process, which would use a `<reply>` activity to respond synchronously to the caller.)

```
<sequence name="main">
  <!-- Receive input from requester.
  ...
  -->
  <receive name="receiveInput" partnerLink="client"
    portType="client:yourLogin_CreditFlow"
    operation="initiate" variable="ssn"
    createInstance="yes"/>
    <!-- Asynchronous callback to the requester.
    ...
   -->
  <invoke name="callbackClient"
    partnerLink="client"
    portType="client:yourLogin_CreditFlowCallback"
    operation="onResult"
    inputVariable="creditRating"
  />
</sequence>
```

The process is editable from both the source view and the diagram view interchangeably.
Switch back to visual design view by clicking to the DiagramView tab at the bottom of the `yourLogin_CreditFlow.bpel` window to return to the Diagram view.

ADD ACTIVITIES TO THE PROCESS MAP

You are now ready to edit the process. A BPEL scope is a collection of activities that can have its own local variables, exception handling, compensation, and so on — very much analogous to a programming language `{ }` block. To insert a <scope> activity:

- In the BPEL Component Palette on the right, select “Process Activities” is selected in the drop down menu (by default you will see “BPA Blue Prints”).
- Drag a `<scope>` activity (from **scope** on the BPEL Palette) to the available position between the `receiveInput <receive>` activity and the `callbackClient <invoke>` callback element.
- Double click the scope element and enter *getCreditRating* into the name field.
- Click the “+” icon to the left of the `<scope>` activity in the process flow, to expand the scope so that you can drop activities into it.
- Drag an `<invoke>` activity from the BPEL Palette into the **Drop activity here** area within the scope.

The next step is to configure the `<invoke>` activity to call your intended service (in this case the credit rating service). In BPEL, this means you first need to create a partnerLink for the service.

CREATE A PARTNERLINK FOR THE CREDIT RATING SERVICE

Partnerlinks represent services that BPEL processes can call or interact with.

- In right swimlane of the Diagram view of the BPEL process, right-click and select **Create PartnerLink…** This will bring up the Create PartnerLink Wizard above.
- Select the flashlight (Service Explorer) icon above the WSDL File text field. This opens the service browser where you can select the service endpoint you are looking for.
- The wizard presents an explorer listing all the services deployed on any Oracle BPEL server or UDDI registers JDeveloper has been configured to.

- Navigate to **BPEL Services** > **BPELIntegServer > processes > default > CreditRatingService** and Click **OK**. You have selected the CreditRatingService that is deployed to the BPEL server.

The browser will close and the URL of the service will appear in the WSDL File field. The wizard will fetch the contents of the WSDL file so that it can populate the drop-down lists appropriately. Alternatively, you can enter the URL for a WSDL directly in this field if you know the specific location of the WSDL for the service you want to invoke.

- Click **Apply**, then **OK**. A new partnerLink has been added to your flow.

- Now double-clink onto the CreditRatingService PartnerLink and in Partner Role list, select `CreditRatingServiceProvider`. You will leave the `myRole` field blank. (Because this is a synchronous service without any callbacks, the client does not need a role.) Note: due to a bug in JDeveloper, we had to create the PartnerLink first and then specify the Partner Role.

CONFIGURE THE <INVOKE> ACTIVITY

Now you will need to associate the invoke activity with the PartnerLink you just created:

- Select the arrow on the right of the invoke activity and drag it onto the creditRatingService partnerlink. After taking this step the Edit Invoke wizard appears.
- The Partnerlink Field is already filled out. The Operation drop-down box has the operation process selected. This is because the creditRatingService only provides one available operation.
- Type *invoke_CRS* in the name field. You need to specify an input variable and an output variable. These two variables are passed to, and received from the creditRatingService. The wizard allows you to create a new variable or select existing variables.
- You will create new variables by selecting the wand icon on the right of the variable fields. A new Create Variable wizard appears, accept the defaults (or specify your preferred variable name) and select **OK**. Complete this step for both the Input and Output variables.
- Select **Apply** and **OK**.

You have now configured the Invoke activity to call the creditRatingService partnerlink.

INITIALIZE THE INPUT VARIABLE

Now you will use XPath and the BPEL `<assign>` activity to perform simple data manipulation to initialize the input variable you are passing to the credit rating service.

- Drag an `<assign>` activity from the BPEL Palette into your flow just before the invocation of the credit rating service (but within the `getCreditRating` scope).
- Double click the newly created Assign activity to update its properties.
- Select the **General** tab on the Assign activity and enter *assign_SSN* into the Name field.
- Select the **Copy Operation** tab and click **Create → Copy Operation…** from the drop-down menu. The XPath wizard opens. This wizard allows you to assign the value of a variable selected in the From field to the value of a variable selected in the To field.
- In the From section of the Copy Operation form ensure **Variable** is selected in the Type drop-down box and navigate to and select **Variables** > **Process** > **Variables** > **inputVariable** > **payload** > **client:CreditFlowProcessRequest** > **client:ssn**
- Note: because of a bug in JDeveloper, you might see client:input instead of client:ssn when entering the “from” section of the Copy Operation. If so, check first that you have saved the XML schema. If the problem persists, then select client:input, then go to the “XPath” text field at the bottom of the pop-up window, and manually replace “input” with “ssn”.

- Fill in the **To** section of the Copy Operation form as follows: In the Type drop-down select **Variable**. Navigate to and select **Variables > Invoke_CRS_process_InputVariable > payload > > ns1:ssn**
- In the Copy Operation form, click **OK**.
- In the Assign form select **Apply**, then **OK**.
- Make sure you Save the BPEL process.

Note that you can always enter XPath queries directly into the text field if you know what the correct query is. For more information regarding data manipulation in BPEL please refer to the BPEL Data Manipulation chapter in the BPEL PM Developer's Guide at http://otn.oracle.com/bpel.

To complete the implementation of this flow, you would add another `<assign>` activity to the flow (after the credit rating service has been invoked), which would copy the result returned from the credit rating service (in your `invoke_CRS_process_OutputVariable` variable, if you have kept the default name) into the `creditRating` field for the result of your process itself (the `outputVariable` variable).

VALIDATE, COMPILE, DEPLOY AND TEST THE BPEL PROCESS

To validate your BPEL Process, press the green tick icon on the upper-left corner of the BPEL canvas. If you followed all the steps of this tutorial, any warning will disappear.

Now the process is ready to be deployed. To deploy the CreditFlow process to the BPEL server, right click on the project yourLogin_CreditFlow and select **Deploy** > **BPELIntegServer** > **Deploy to defaul**t domain.

Then open the console at http://serverName.at.mt.ut.ee:9700/BPELConsole and test your process with mock data. To do so:

- Click yourLogin_CreditFlow on the Dashboard and in the automatically generated HTML Form interface that appears, enter a nine-digit number that does not begin with a 0, and click Post XML Message to initiate the process.
- Click the Visual Flow link to see a visual audit trail representing the current state of the process instance. Note that because this process is asynchronous and long-running you do not get the result returned right away (like you did when testing the CreditRatingServce). Instead, if you follow the Visual Flow link, you will see an audit trail displaying the current state of the process, very similar to the process map displayed by the BPEL Designer. It will show that you have successfully invoked the credit rating service and replied back to the client.
- Click the `Invoke_CRS <invoke>` activity in the audit trail (a few boxes down) to see the messages sent to and received from the credit rating service.
- Likewise, click the `callbackClient <reply>` activity to see the response sent to the client, which incorporates the rating provided by the credit rating service.

A number of other examples are available under the directory where you chose to install the Oracle SOA suite (e.g. C:\OraBPELPM_1\integration\orabpel\samples). Other tutorials are available here: http://www.oracle.com/technology/products/ias/bpel/htdocs/dev_support.html#tutorials

Acknowledgement: This tutorial was developed by Marcello La Rosa, Queensland University of Technology, Australia