ORACLE

BPEL Designer Tutorial

Tutorial 2: Developing a Credit Flow BPEL Process

In this tutorial you will learn how to use the Oracle BPEL Designer to build, deploy, and test your second BPEL process. The process is an asynchronous flow that calls a simple service: a synchronous credit rating service. Creating this process is intended to be the first step toward building a more sophisticated application (like BPEL Loan Flow example).

This tutorial is based on version 0.8 of the BPEL Designer.

Contents

Prerequisites

This tutorial assumes you have completed Tutorial 1, "Developing a Hello World BPEL Process." The same prerequisites apply as for that tutorial (as listed below); you should have the BPEL Designer installed on your system as well.

- Oracle BPEL Process Manager, version 2.0 RC8 or later installed on your system
- Internet Explorer 6.0
- JDK 1.4.1 (or later)
- Windows 2000 or XP, 384 MB RAM
- 100 MB of disk space for the BPEL Designer (including the Eclipse 3.0M9 distribution)
- Some familiarity with XML Schema, WSDL, XPath, BPEL, and related Web service standards

Getting Started

If the BPEL Designer and Oracle BPEL Process Manager are not currently running on your desktop, start them now, as described in Tutorial 1. Like that tutorial, this one assumes you have installed the Oracle BPEL Process Manager on a Windows system into the `C:\orabpel` directory, so you should modify paths as appropriate if you have installed it into a different location or operating system.

Before starting the main content of this tutorial, you should compile and deploy the credit rating service that you will invoke as part of the tutorial, making it available on your local Oracle BPEL Process Manager. You will use the command line rather than the BPEL Designer to compile and deploy this service

> Note that if you wanted to compile this project from the BPEL Designer instead of the command-line, you could use the File / Import / Existing Project into Workspace menu command to open the CreditRatingService in the BPEL Designer and build it using the graphical interface as was done in Tutorial 1.

To compile and deploy the credit rating service from the command-line:

1 Open up a command prompt if you do not already have one open.

2 Change the directory to `C:\orabpel\samples\utils\CreditRatingService`, as follows:

> **cd C:\orabpel\samples\utils\CreditRatingService**

3 Execute the **obant** command.

> **obant**

To test the credit rating service:

1 If you have not already done so, connect to the BPEL Console (at http://localhost:9700/BPELConsole) and log in.

2 In the **Name** column (under the **Dashboard** tab of the console), click the link for the `CreditRatingService` BPEL process.

3 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

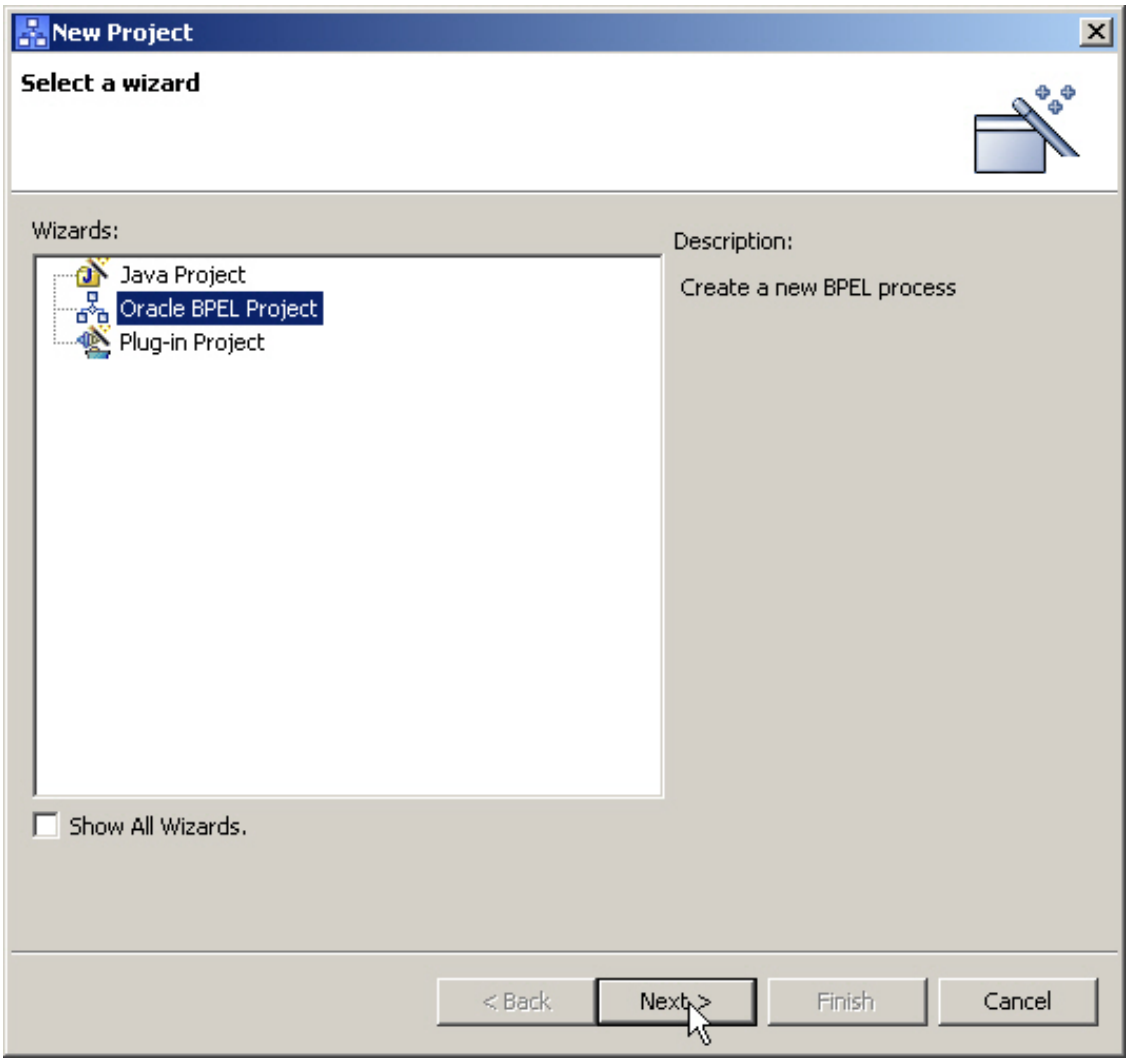
As in Tutorial 1, you will use the BPEL Designer's New Project wizard, which automatically generates the skeleton of a BPEL project: the BPEL source, a WSDL interface, a BPEL deployment descriptor, and an Ant script for compiling and deploying the BPEL process. This time you will specify that you want to start with the template for an asynchronous process.

To create a new BPEL project:

1 In the BPEL Designer, click **File > New > Project**.

By default, **Oracle BPEL Project** is selected.

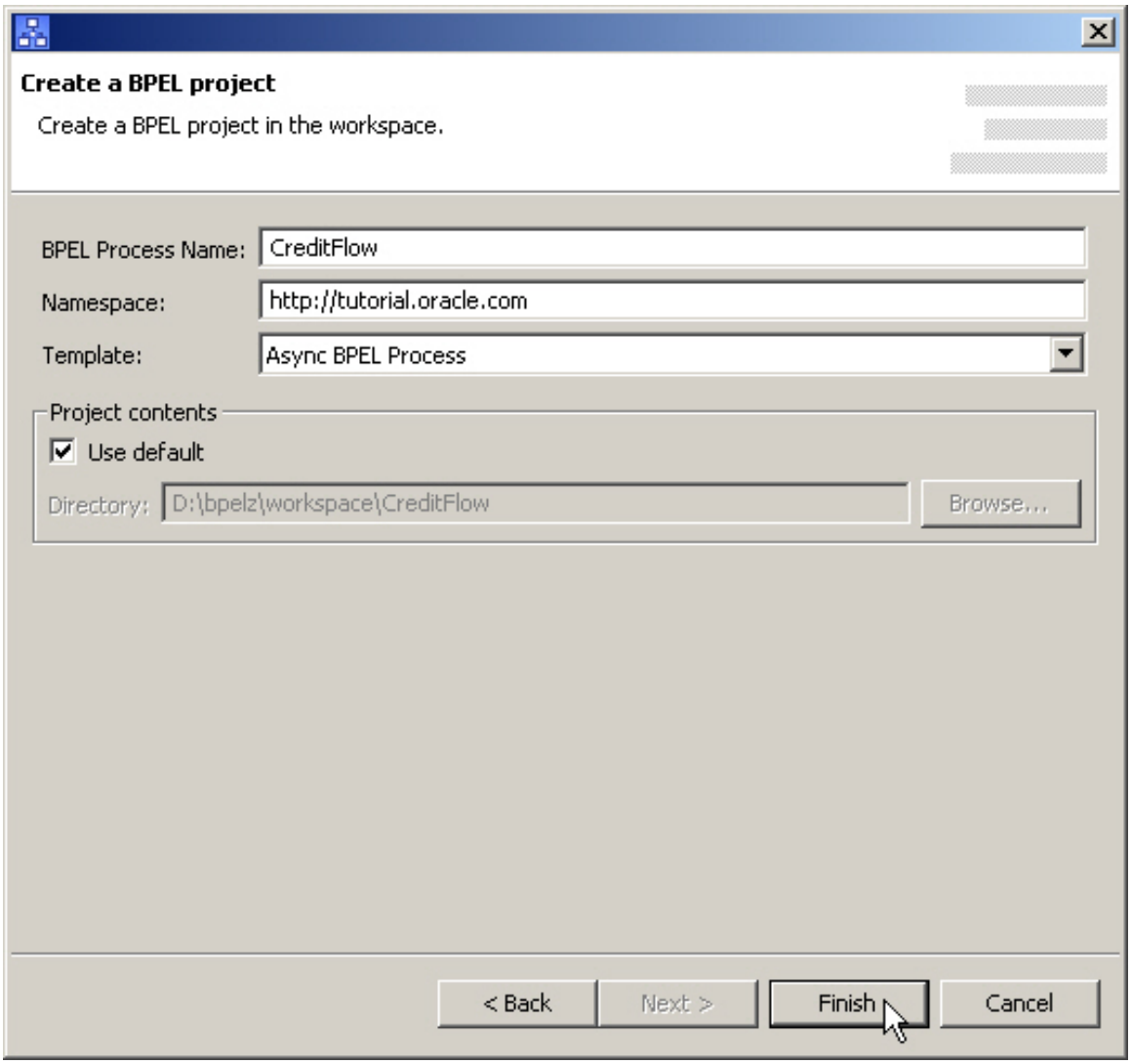
2 Click **Next**.



3 Enter a BPEL process name of `CreditFlow`.

4 [Optional] Enter `http://tutorial.oracle.com` as the namespace.

5 Leaving the template as **Async BPEL Process** and the **Use default** location at its default setting, click **Finish**.



The new project will be created in the `C:\bpelz\workspace` directory. You can see (in the Navigator within the 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 similar to those created for a synchronous process (as in Tutorial 1), but the contents of the `.bpel` and `.wsdl` files differ as appropriate for an asynchronous process; you will see these differences as you proceed through this tutorial.

Note: We recommend saving your project frequently as you build it.

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:

1 In the Navigator, double-click the `CreditFlow.wsdl` file.

2 Scroll down as necessary to browse the code, focusing on how it differs from the WSDL interface for the synchronous process you created in Tutorial 1. The main differences (excluding name differences) are pointed out below.

Whereas only one portType was defined for the earlier synchronous process, two portTypes are defined for this asynchronous process, 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="CreditFlow">
    <operation name="initiate">
        <input message ="tns:CreditFlowRequestMessage"/>
    </operation>
</portType>

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

The other key difference is that, unlike the partnerLinkType defined for the synchronous process, which has only one role, the one defined for this asynchronous process has two roles, one for the service provider and one for the requester.

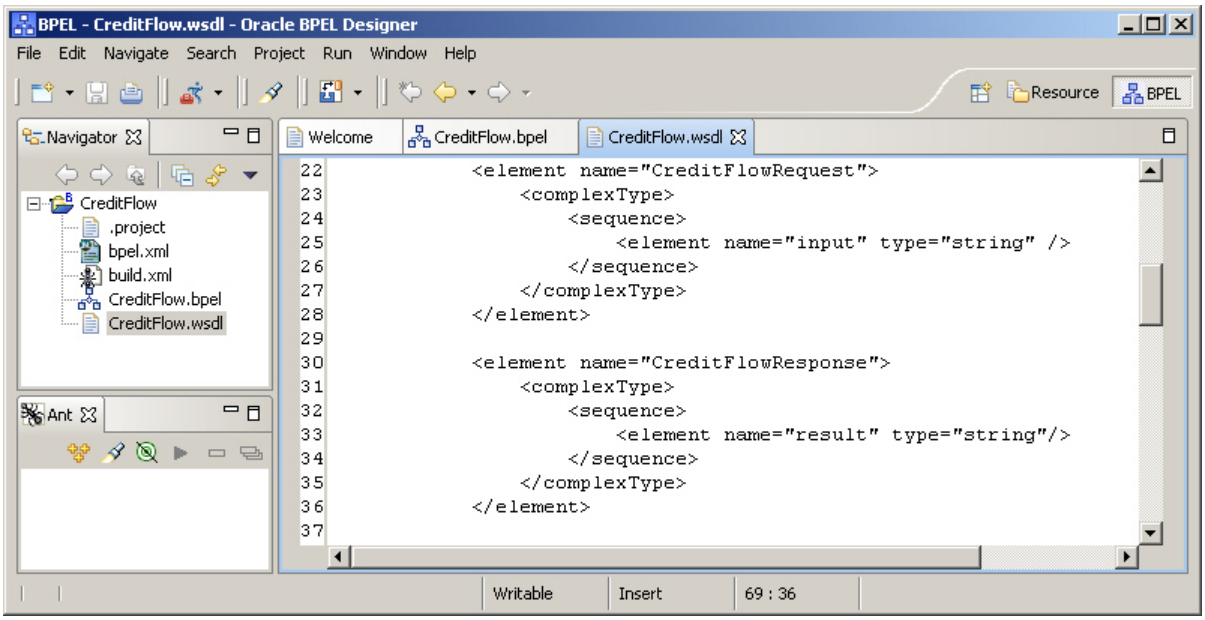
```
<plnk:partnerLinkType name="CreditFlow">
    <plnk:role name="CreditFlowProvider">
        <plnk:portType name="tns:CreditFlow"/>
    </plnk:role>
    <plnk:role name="CreditFlowRequester">
        <plnk:portType name="tns:CreditFlowCallback"/>
    </plnk:role>
</plnk:partnerLinkType>
```

Edit the WSDL Interface of Your BPEL Process

Now you will edit the WSDL file to modify the input and output messages.

To modify the input and output messages of your BPEL process:

1 While still editing your `CreditFlow.wsdl` file, scroll up near the top. Notice that the New Project wizard has defined both a `CreditFlowRequest complexType` element that your flow accepts as input (in a document-literal style WSDL message) and a `CreditFlowResponse` element that your flow returns.



2 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.

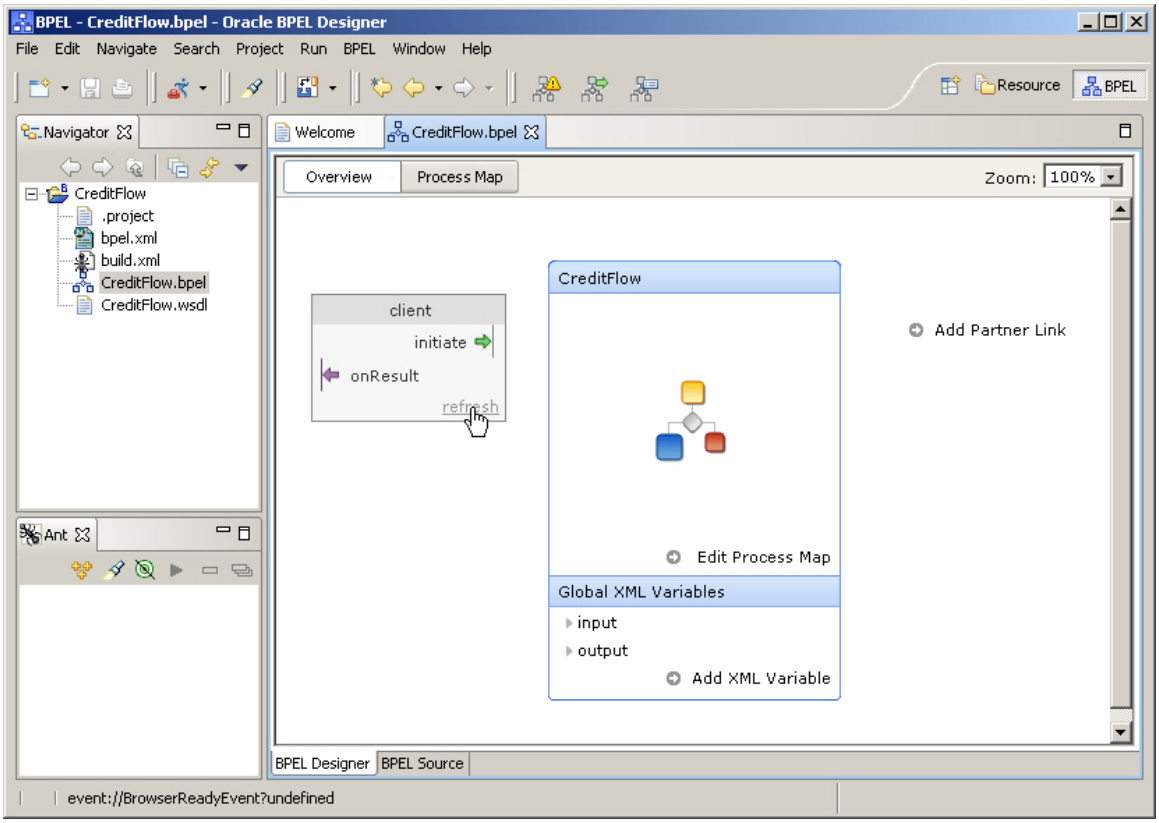
```
<element name="CreditFlowRequest">
    <complexType>
        <sequence>
            <element name="ssn" type="string">
        </sequence>
    </complexType>
</element>

<element name="CreditFlowResponse">
    <complexType>
        <sequence>
            <element name="creditRating" type="int">
        </sequence>
    </complexType>
</element>
```

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

Review the BPEL Source Code

As in Tutorial1, the New Project wizard has created a basic skeleton for you of a BPEL process, but in this case it is an asynchronous process.

1 Note that because you have changed the WSDL file for your CreditFlow.bpel process, you need to refresh the BPEL client interface for the process. You can do this either by clicking the "refresh" link for the client partnerLink (as shown below) or by simply closing and re-opening the CreditFlow.bpel file. You won't see any difference in the UI, but the wizards will now be updated with the new datatypes for the input and output messages for your flow.



To view the BPEL source code:

1 Click the **BPEL Source** tab at the bottom of the `CreditFlow.bpel` window.

2 Scroll down as necessary to browse the code, focusing on how it differs from the code for the synchronous process you created in Tutorial 1. The main differences (excluding name differences) are pointed out below.

The partnerLink created for the client interface includes a `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="tns:CreditFlow"
                 myRole="CreditFlowProvider"
                 partnerRole="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 uses a `<reply>` activity to respond synchronously to the caller.)

```
<sequence name="main">

    <!-- Receive input from requester.
         ...
         -->
    <receive name="receiveInput" partnerLink="client"
             portType="tns:CreditFlow"
             operation="initiate" variable="input"
             createInstance="yes"/>

    <!-- Asynchronous callback to the requester.
         ...
         -->
    <invoke name="callbackClient"
            partnerLink="client"
            portType="tns:CreditFlowCallback"
            operation="onResult"
            inputVariable="output"
            />
</sequence>
```

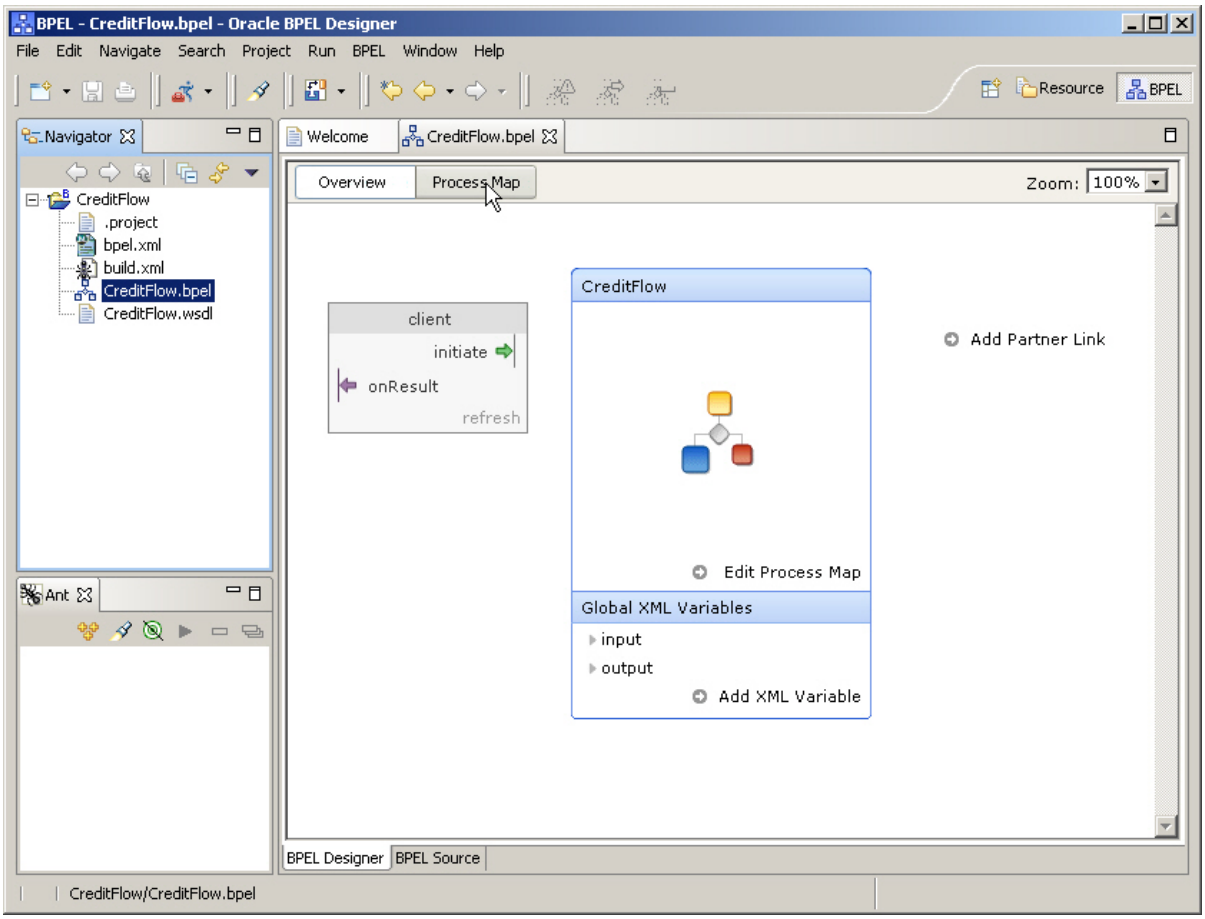
View the Process Map

You will now go to the Process Map view, where you will edit your BPEL process.

1 Click the **BPEL Designer** tab at the bottom of the window.

This returns you to the Overview view of the process, which looks similar to what you observed for the synchronous process created in Tutorial 1. The differences are that in the client interface, the input operation generated by the New Process wizard is named `initiate` and there is an asynchronous callback operation named `onResult`.

2 Click the **Process Map** button at the top of the window (or the **Edit Process Map** link in the middle of the window).



Add Activities to the Process Map

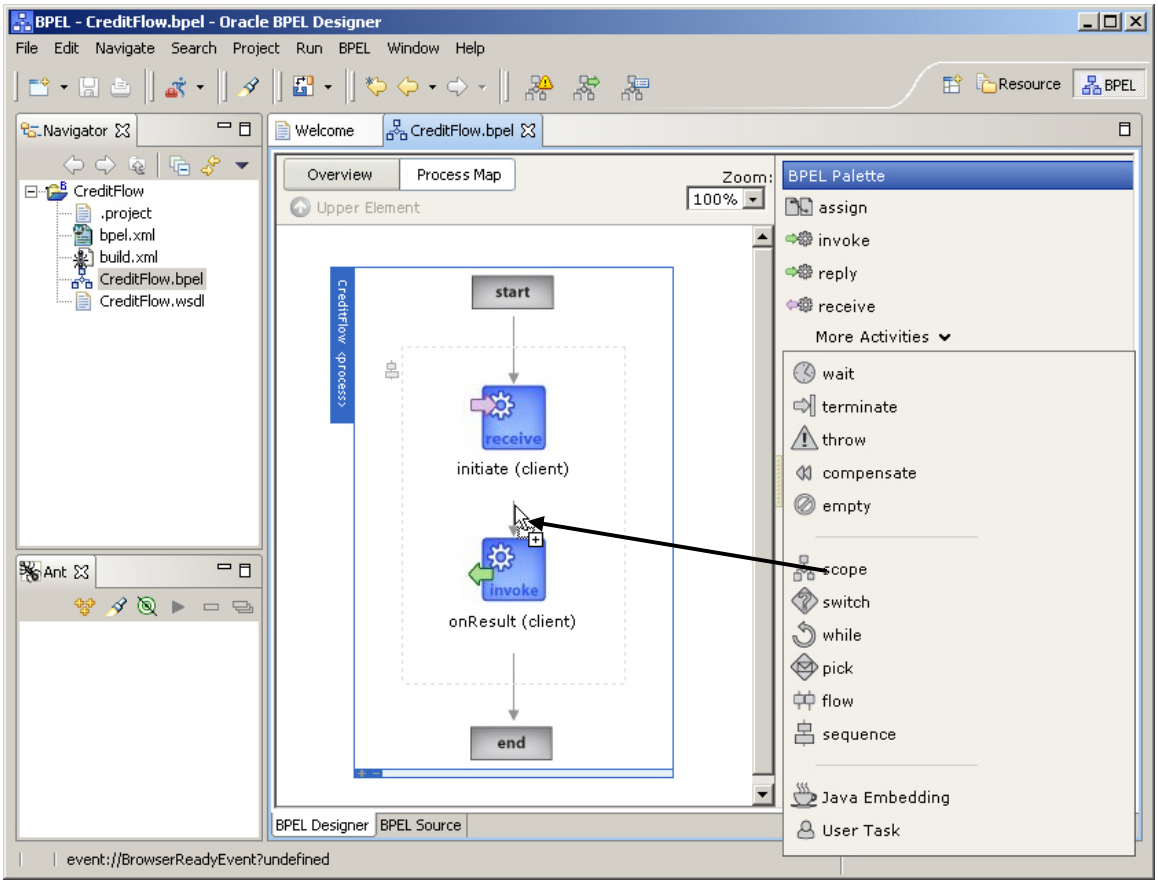
You are now ready to edit the process. You will start by adding two new activities to it: a `<scope>` activity and an `<invoke>` activity.

To insert a <scope> activity:

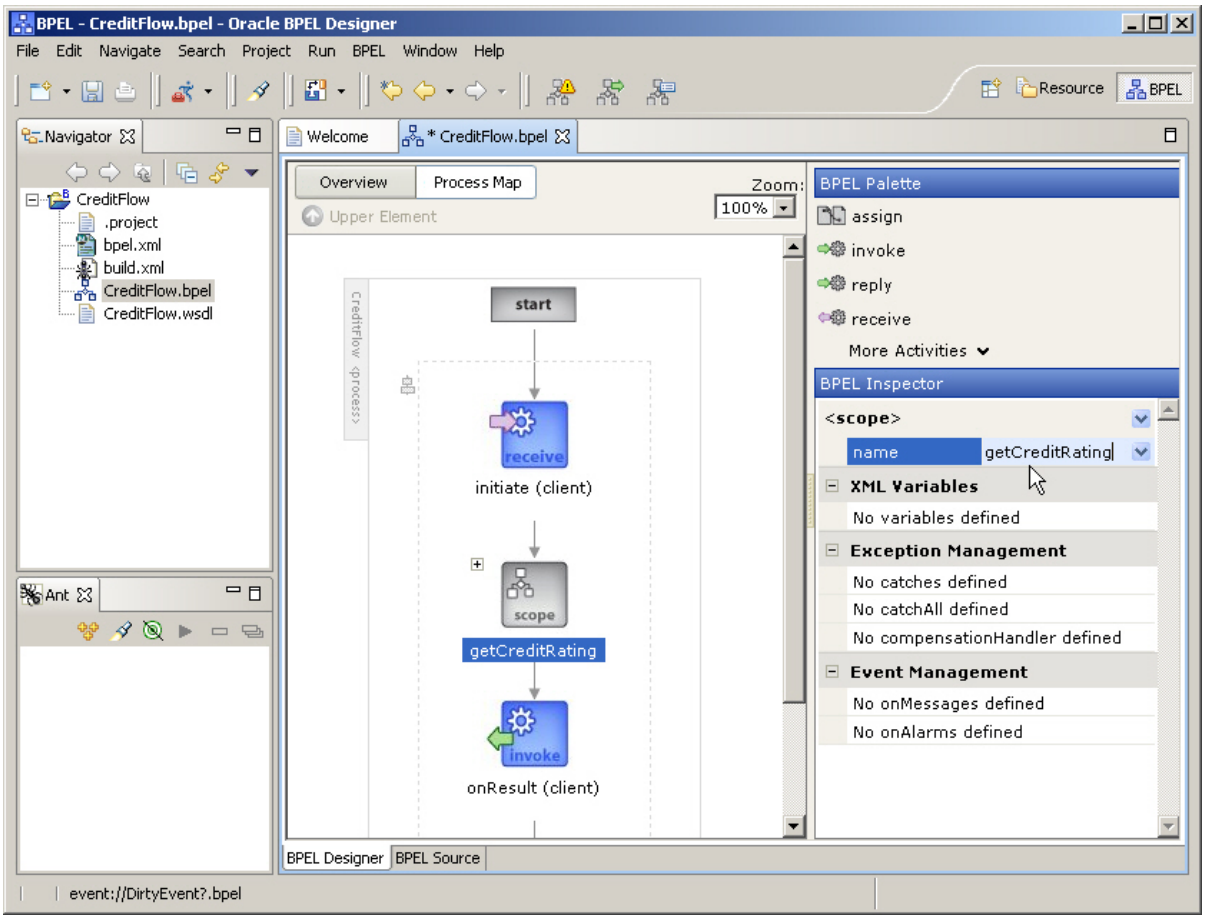
A BPEL scope is a collection of activities that can have its own local variables, exception handling, compensation, and so on — very much analogous to a programming language `{ }` block.

1 In the BPEL Palette on the right, click the **More Activities** link to see additional BPEL activities.

2 Drag a `<scope>` activity (from **scope** on the BPEL Palette) to the transition arrow between the `initiate (client) <receive>` activity and the `onResult (client) <invoke>` callback element.



In the BPEL Inspector, enter the value `getCreditRating` for the `name` attribute of the newly created scope.



To insert an <invoke> activity into the scope:

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.

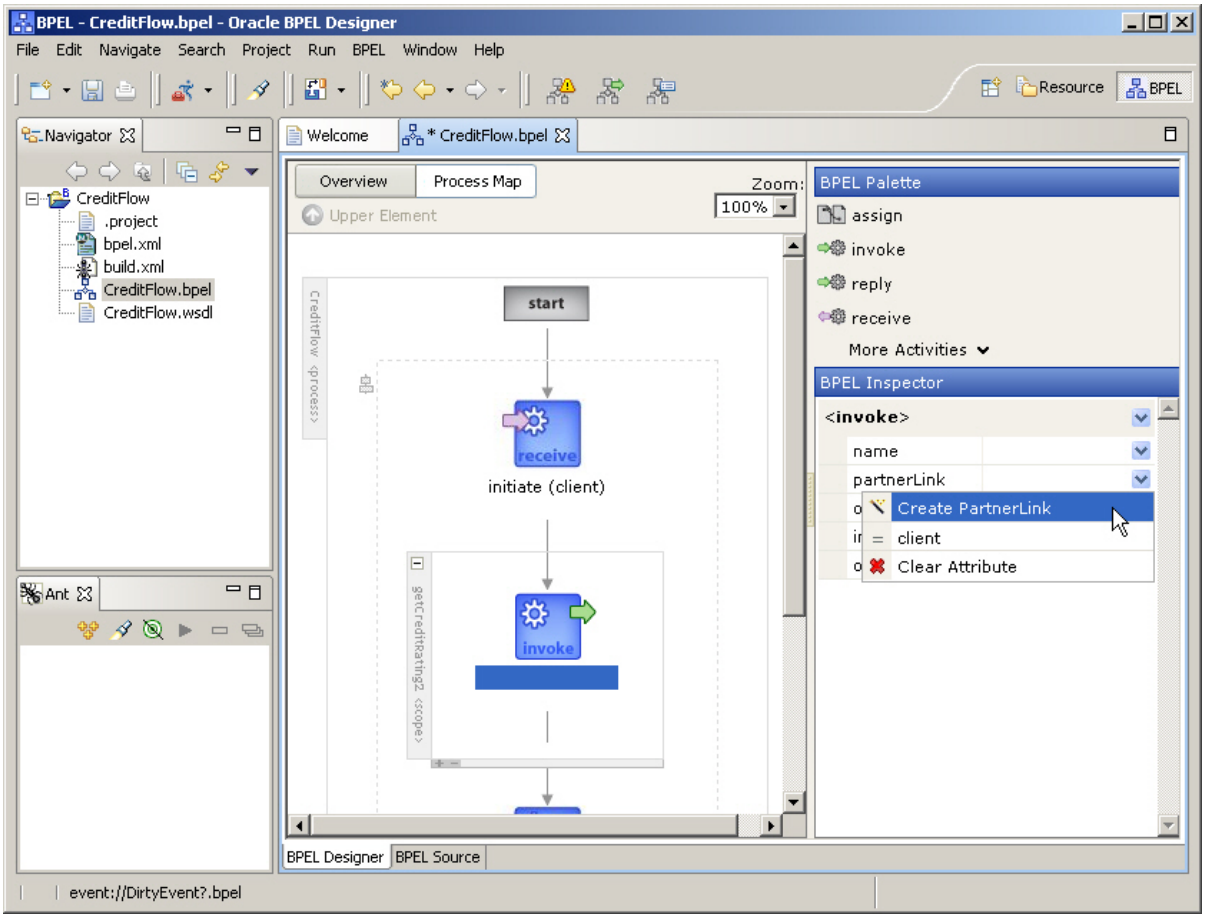
Drag an `<invoke>` activity from the BPEL Palette into the **Drop activity here** area within the scope.

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

In the BPEL Designer, you can add partnerLinks to your BPEL process either in the Overview view or at the time you configure the `<invoke>` activity (using a wizard). Here you will use the latter approach to add a partnerLink for the credit rating service that you built and deployed locally at the beginning of this tutorial.

1 In the BPEL Inspector, select **Create PartnerLink** in the drop-down list for the `partnerLink` attribute.



This will bring up the New partnerLink Wizard.

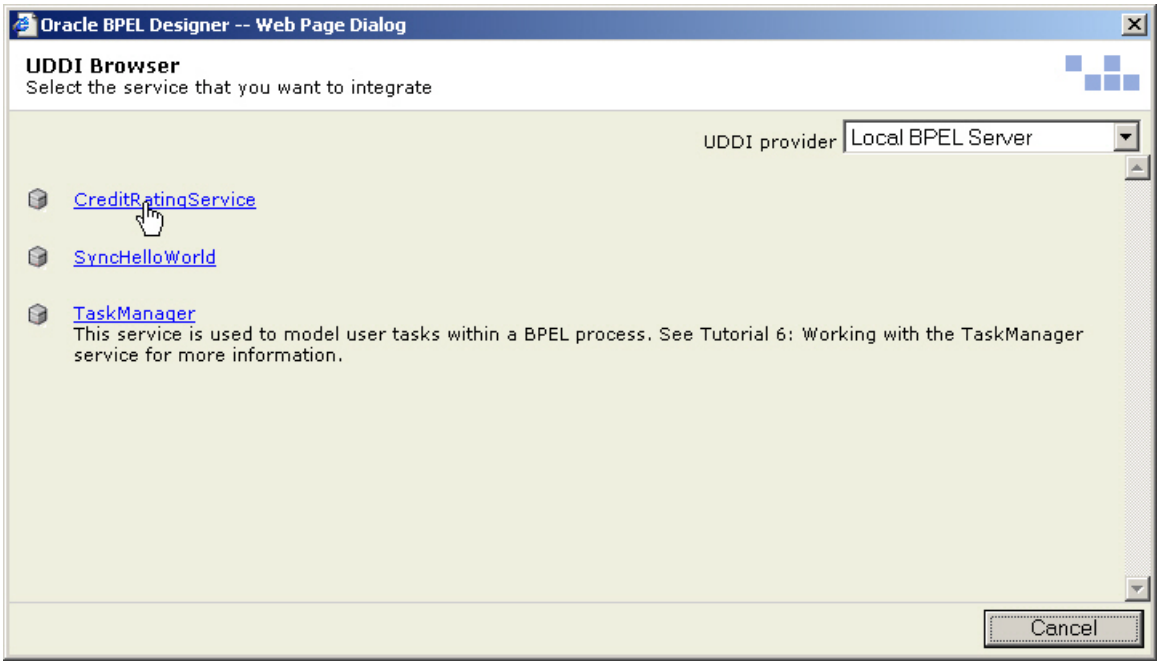
2 Enter the name `creditRatingService` for the partnerLink.

3 Enter the location of the WSDL file as follows:

a Click the "..." button to the right of the `wsdlLocation` field.

The wizard presents a UDDI browser listing all the services deployed on the local Oracle BPEL Process Manager. (It may take a while for the browser to come up the first time.)

b In the UDDI browser, click `CreditRatingService`.



The browser will close, and the URL of the service will appear in the `wsdlLocation` field. The wizard will fetch the contents of the WSDL file so that it can populate 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. If you are using a commercial UDDI server, please refer to http://otn.oracle.com/bpel for information on how to register your UDDI server with the BPEL Designer.

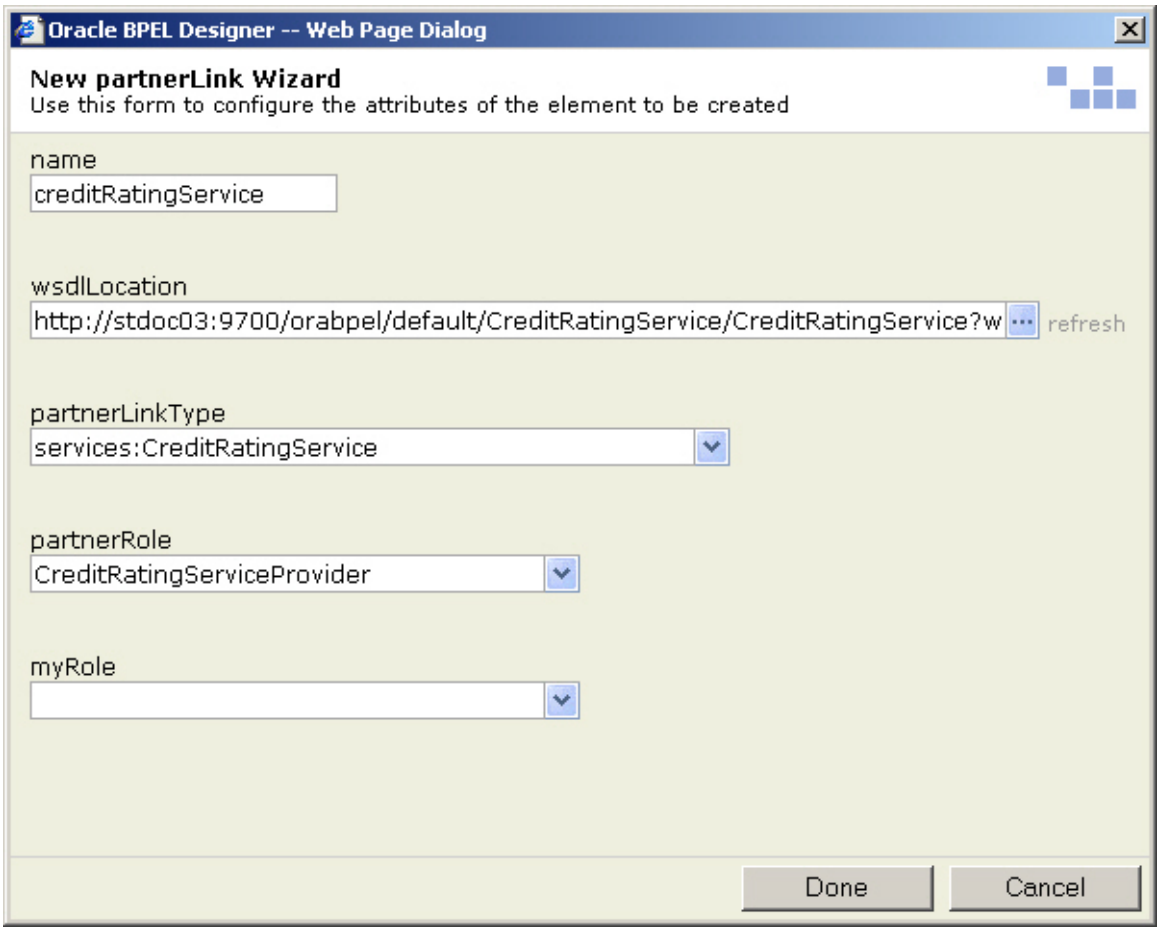
4 In the `partnerLinkType` list, select `services:CreditRatingService` (the only one defined in the WSDL file).

Note: If your service's WSDL file does not include a `partnerLinkType` definition, the BPEL Designer will issue a message to this effect, and you will need to add a `partnerLinkType` definition to the WSDL file. A dialog will prompt you as to whether you want the BPEL Designer to automatically generate one for you in this case.

5 In the `partnerRole` 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.)

6 Click **Done**.

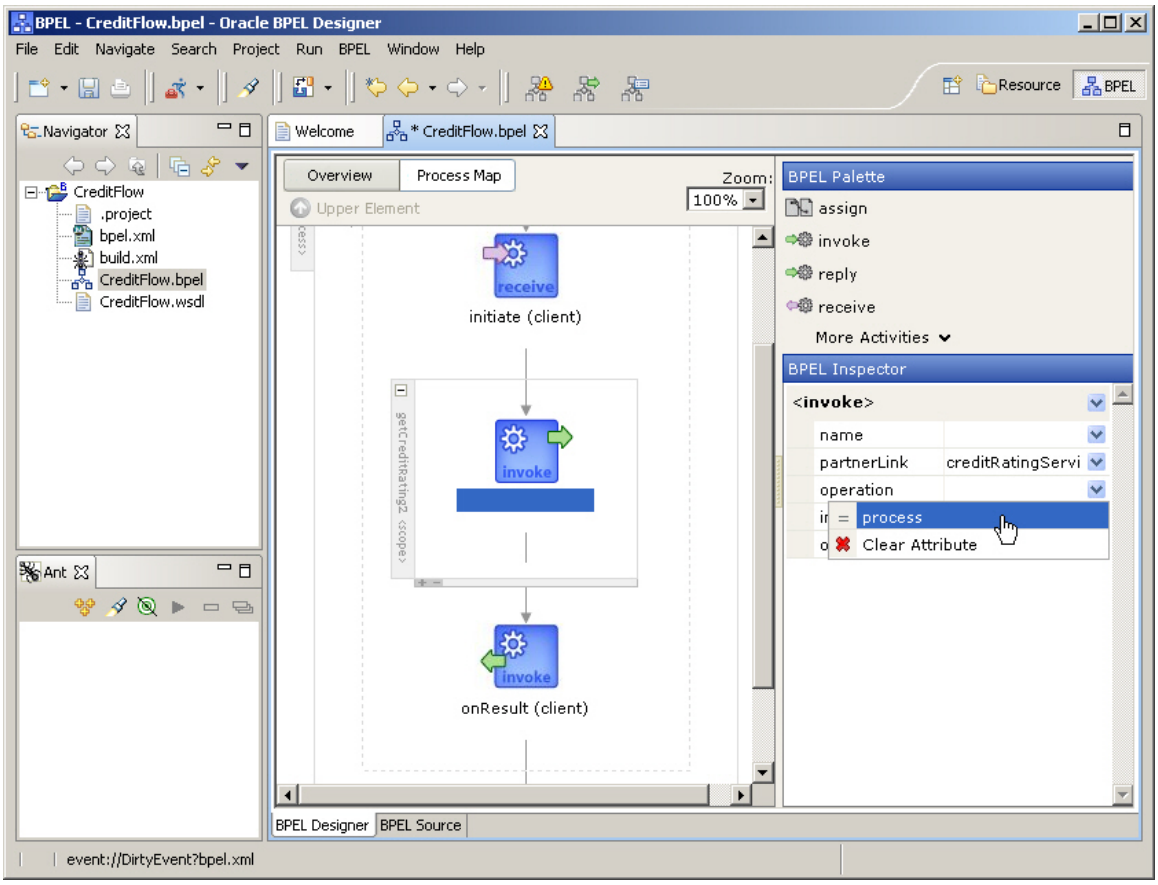


A new partnerLink will be added to your flow (automatically creating namespace shortcuts in your BPEL file as appropriate), and the `<invoke>` activity you just created will have the `partnerLink` attribute automatically filled in.

Configure the <invoke> Activity

Now you will specify the operation you want to invoke for the service. Here the only operation defined by the service is called `process`.

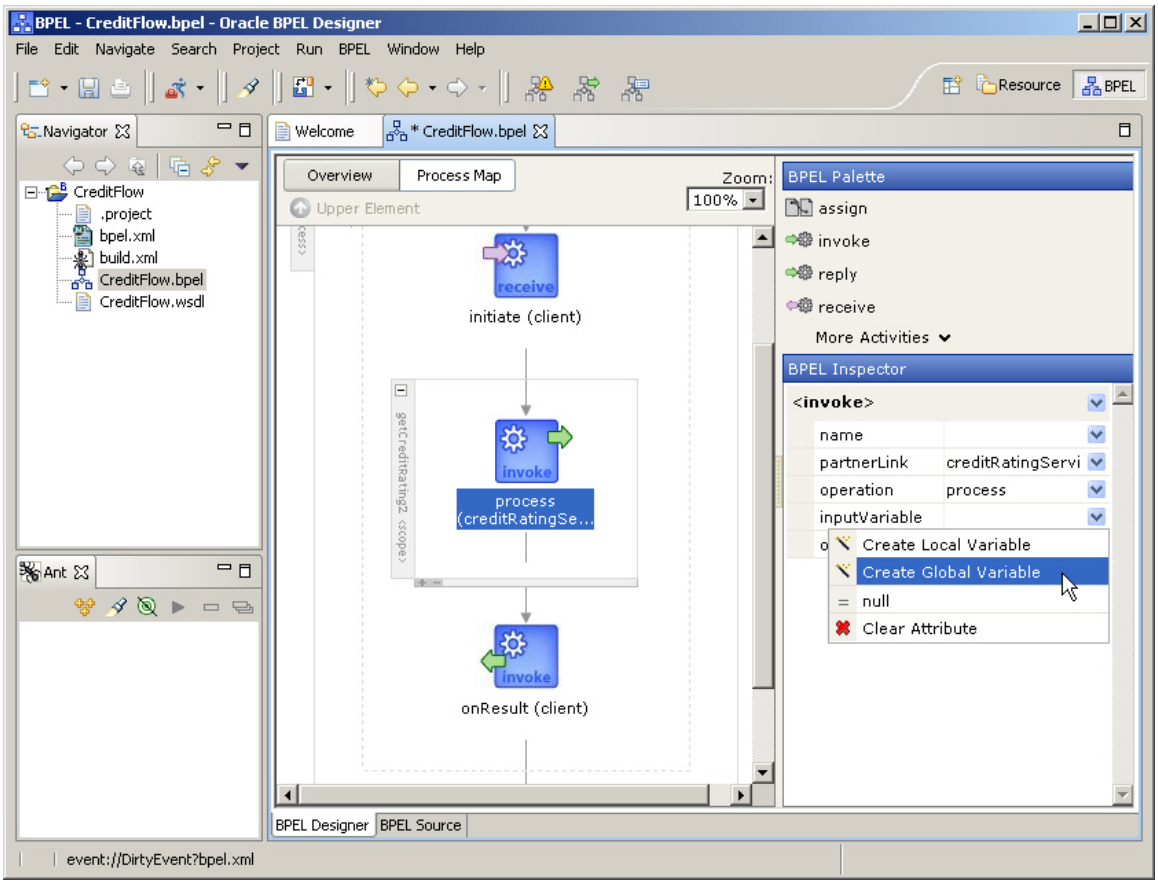
1 In the BPEL Inspector, select `process` in the `operation` list.



After taking this step, you will use the drop-down arrows to the right of the `inputVariable` and `outputVariable` fields. The commands in these drop-down lists will bring up wizards for creating new variables, because the BPEL Designer knows from the WSDL file that the `process` operation has both input and output variables (and what their types are).

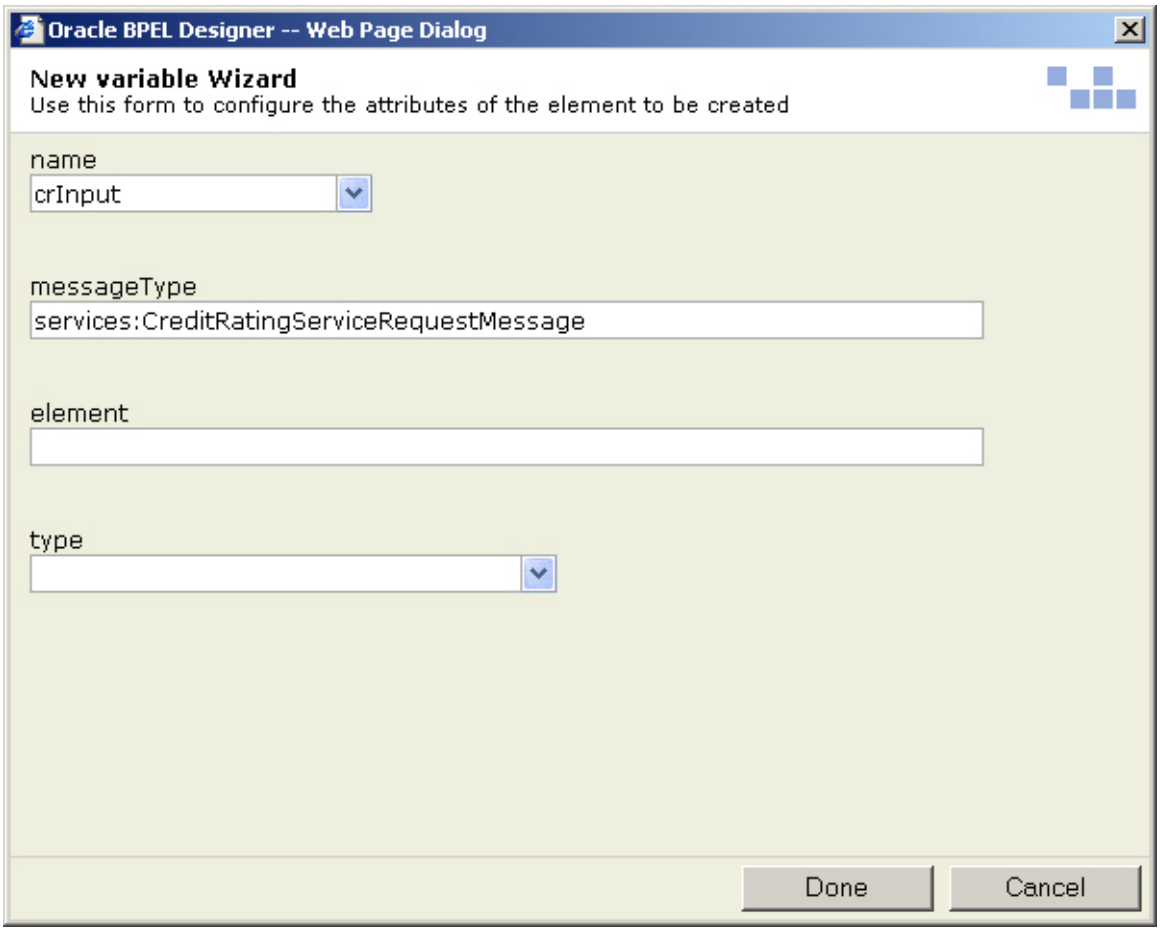
Note: The Inspector doesn't show `portType` as one of the `<invoke>` activity's attributes because that attribute is fully specified by the partnerLink. If you look in the BPEL source code, however, you will see that `portType` is set automatically.

2 In the `inputVariable` list, select **Create Global Variable**



This will bring up the New variable Wizard, with the `messageType` field already filled in with the appropriate type. In the next step, you will enter the variable name. The remaining two fields enable you to specify XML schema elements or built-in XML schema simple types.

3 Enter a variable name of `crInput` and click **Done**.



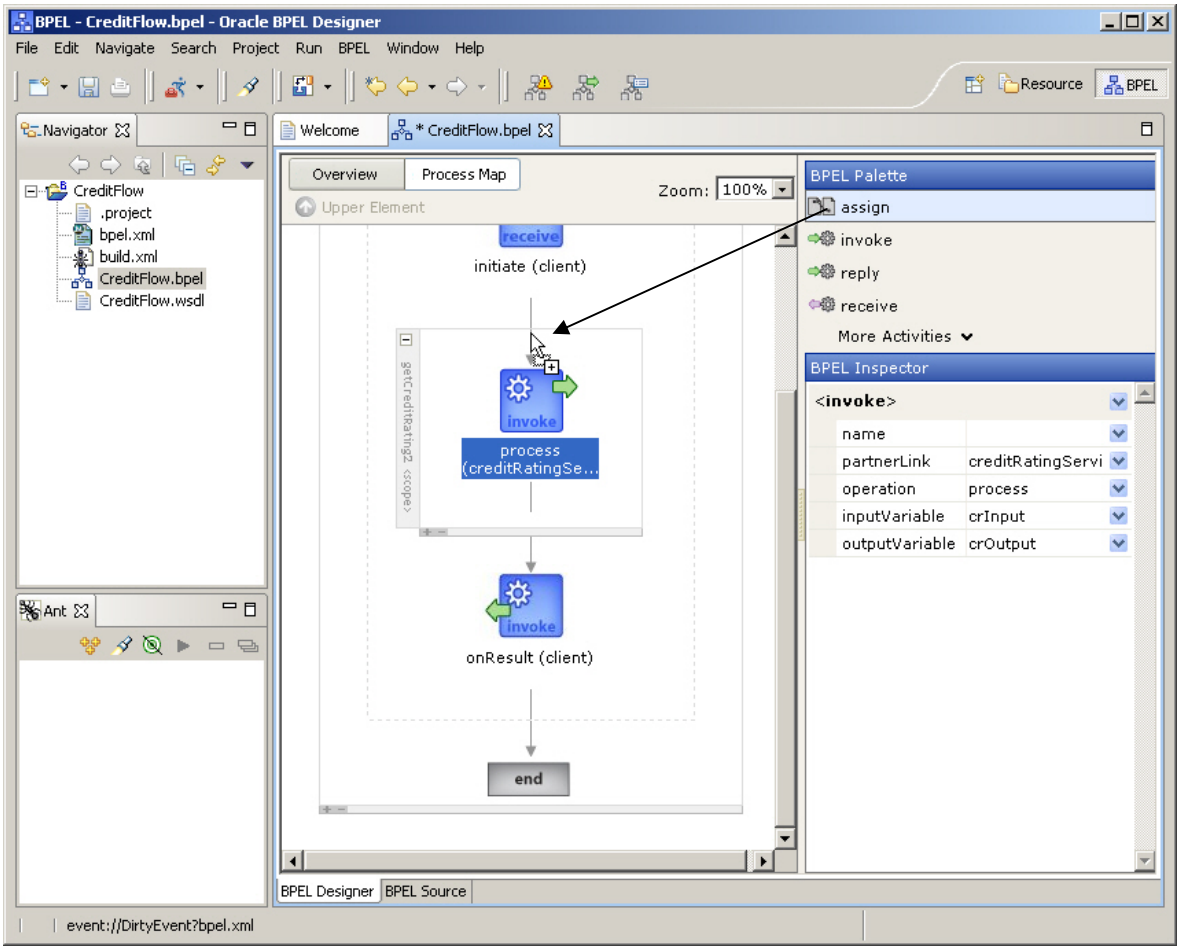
Note that you should only fill in one of the three type fields: `messageType`, `element`, or `type`. A future release of the BPEL Designer will enforce this by requiring you to select the type field you will use with a radio button.

4 Do the same for the `outputVariable` attribute to create a global variable named `crOutput`, using the automatically specified messageType.

Initialize the crInput Variable

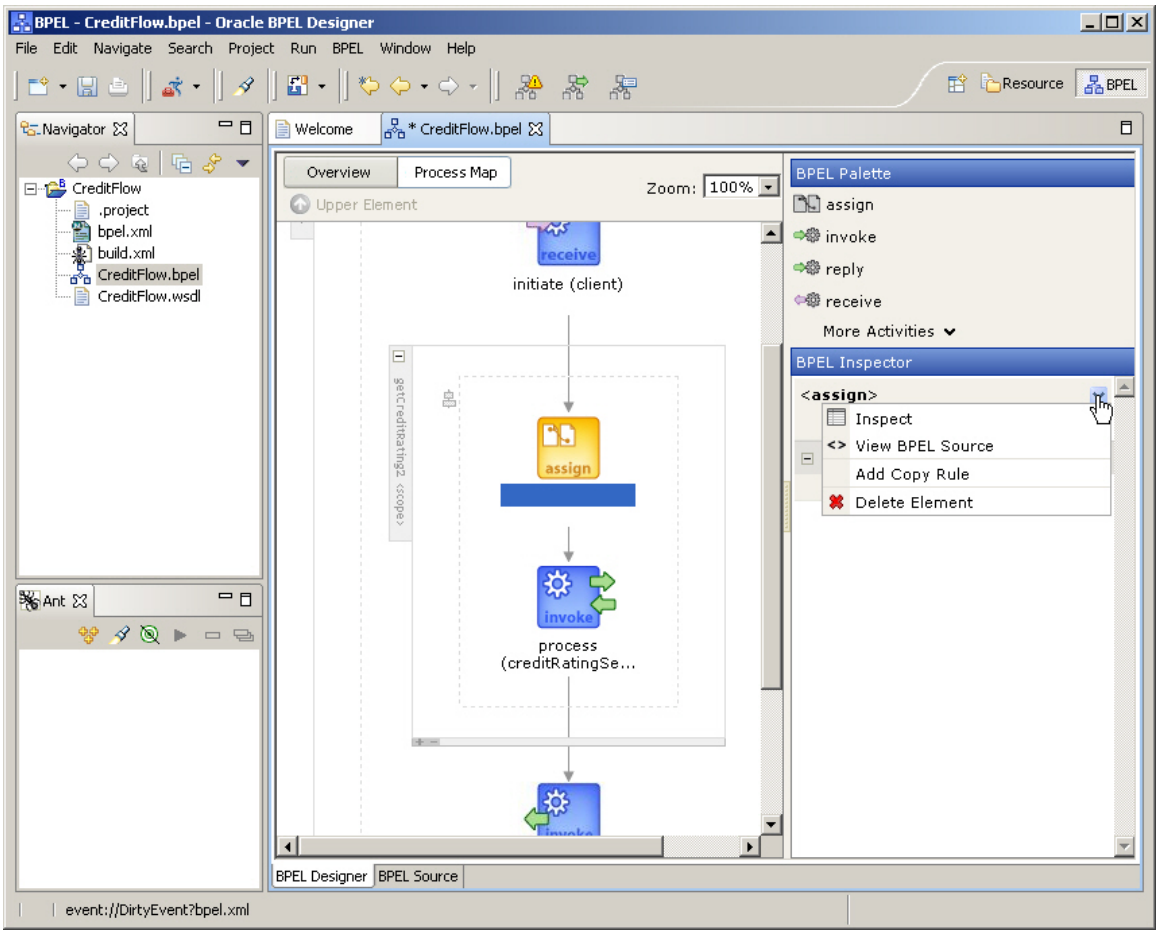
Now you will use XPath and the BPEL `<assign>` activity to perform simple data manipulation to initialize the input variable you are passing to the credit rating service.

1 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).



As before, the newly created `<assign>` activity is now selected, and you can use the BPEL Inspector to configure its attributes.

2 In the Inspector, click the **tasks** drop-down arrow in the top right of the `<assign>` section and click **Add Copy Rule**.



Now you will use the Copy Rule form to copy the `ssn` field from your flow's input message into the `ssn` field of the credit rating service's input message.

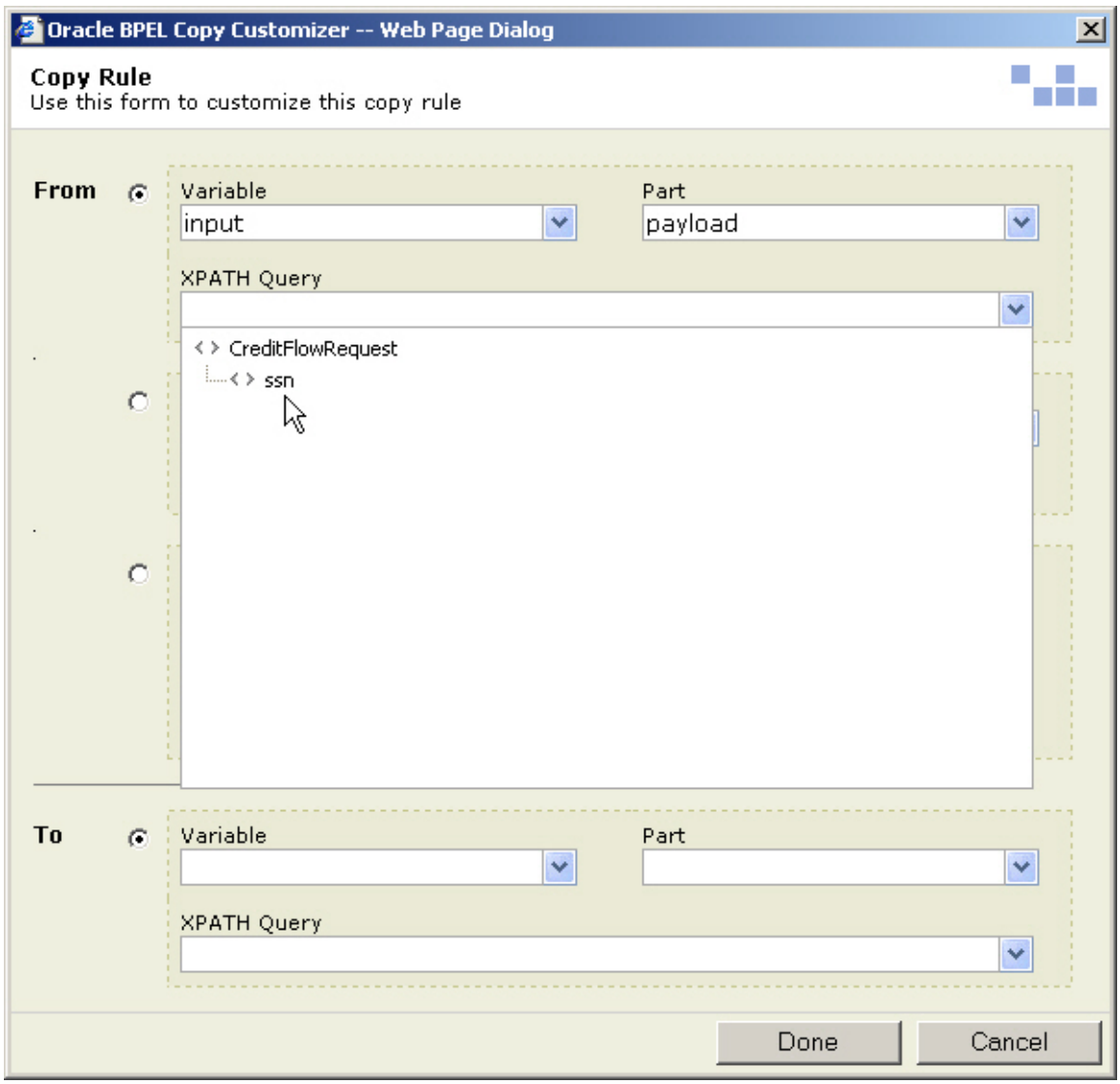
3 Fill in the **From** section of the Copy Rule form as follows:

a In the **Variable** list, select `input` (which is the variable passed as input to kick off your BPEL process).

Once you select a variable, the **Part** list is populated with the appropriate values, based on the BPEL Designer's reading of the XML Schema type definition for this variable type.

b In the **Part** list, select `payload`.

c Click the **XPATH Query** field's drop-down arrow and, in the tree view displayed by the XPath query editor, select `ssn`. This enters the query `/tns:CreditFlowRequest/tns:ssn` in the field.



4 If you want to understand where the XPath query above comes from, take the following steps to view related parts of the source code (or simply read along and look at the extracted code below).

   a Click **Done** to close the Copy Rule form in its half-finished state.

   b Click the **BPEL Source** tab to view the `CreditFlow.bpel` source code, and notice that the `input` variable is of type `CreditFlowRequestMessage`.

```
<!-- Reference to the message passed as input during initiation -->
<variable name="input" messageType="tns:CreditFlowRequestMessage"/>
```

c In the Navigator, double-click `CreditFlow.wsdl` and notice that the `CreditFlowRequestMessage` message type is defined as follows:

```
<message name="CreditFlowRequestMessage">
        <part name="payload" element="tns:CreditFlowRequest"/>
</message>
```

From this you can see that the part named `payload` will return an XML element of type `CreditFlowRequest`, where `CreditFlowRequest` is defined in the WSDL file as:

```
<element name="CreditFlowRequest">
       <complexType>
           <sequence>
               <element name="ssn" type="string"/>
           </sequence>
       </complexType>
    </element>
```

From the definitions above, you can see that the XPath query to get from the part named `payload` to the `ssn` field is `/CreditFlowRequest/ssn`.

d Return to the `CreditFlow.bpel` window, click the **BPEL Designer** tab, and bring back the Copy Rule form by clicking **copy** under **Copy Rules** in the Inspector.

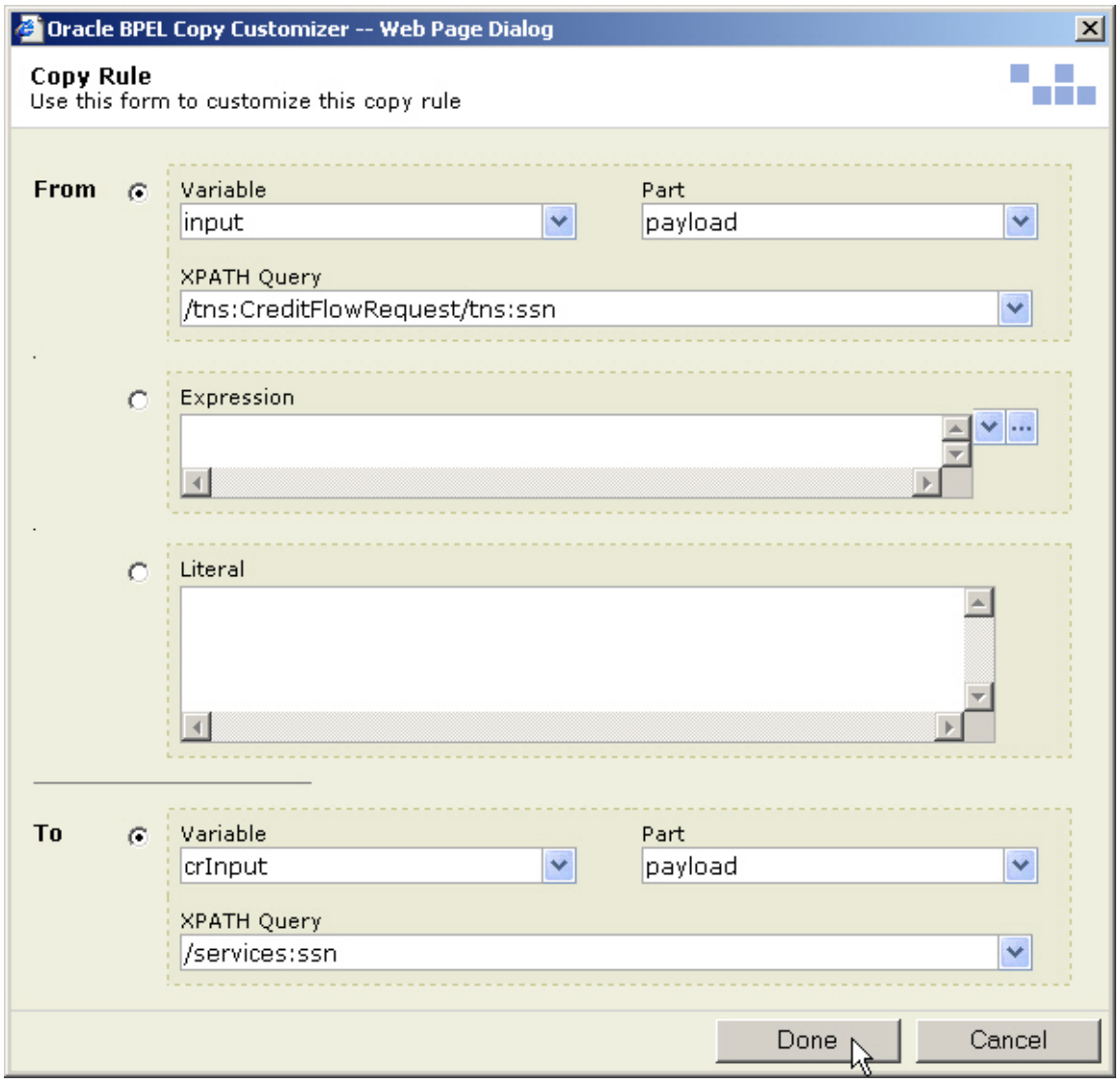
5 Fill in the **To** section of the Copy Rule form as follows:

a In the **Variable** list, select `crInput`.

b In the **Part** list, select `payload`.

c In the XPath query editor, select `ssn` (which will enter the query `/services:ssn` in the XPath query field).

6 In the Copy Rule form, click **Done**.



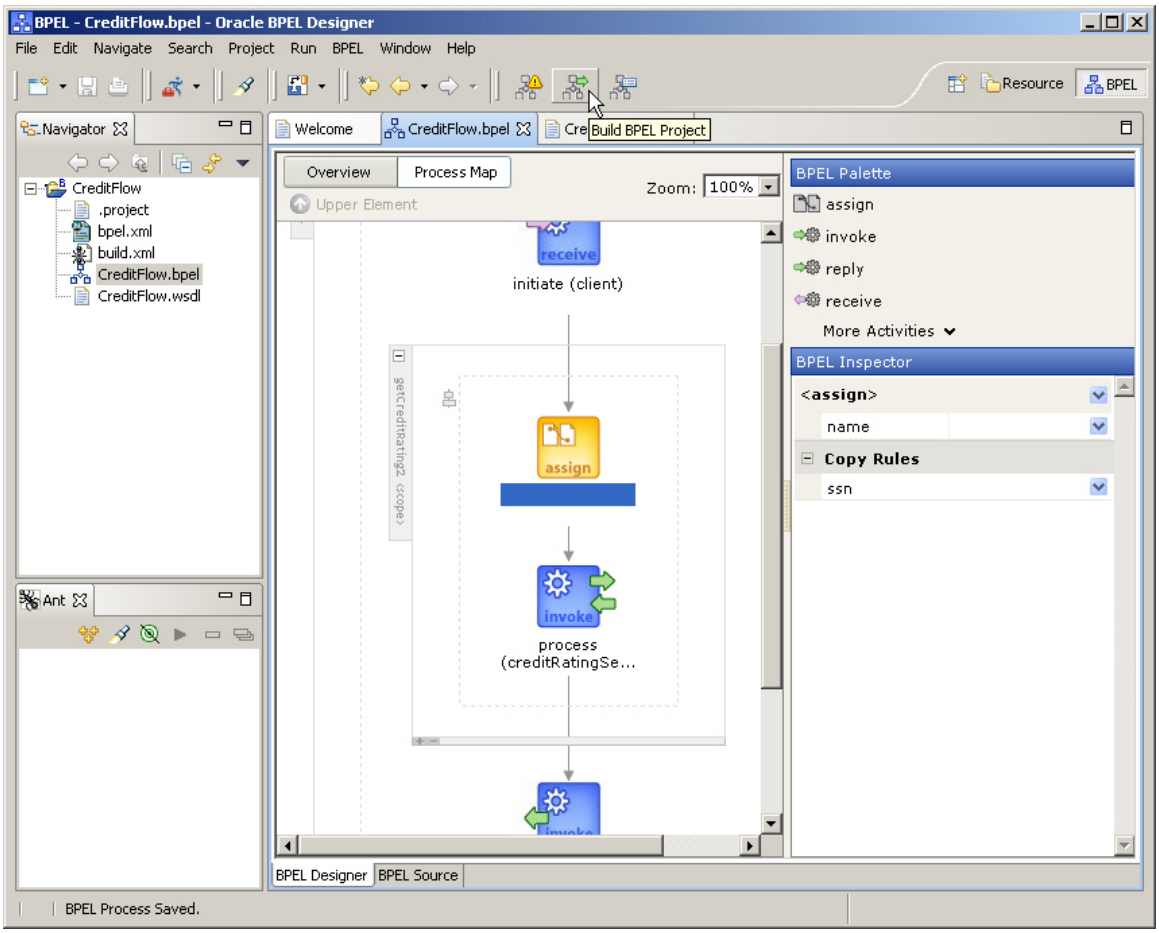
Compile, Deploy, and Test Your BPEL Process

Although you have not yet wired up the return value from the credit rating service to the return value of your flow, you can still test your flow. In this section of the tutorial, you will compile, deploy, and test your BPEL process.

To compile and deploy your BPEL process:

1 Save the process.

2 Click the Build BPEL Project button on the toolbar, as shown below, to compile the process and deploy it to your local server's default domain.

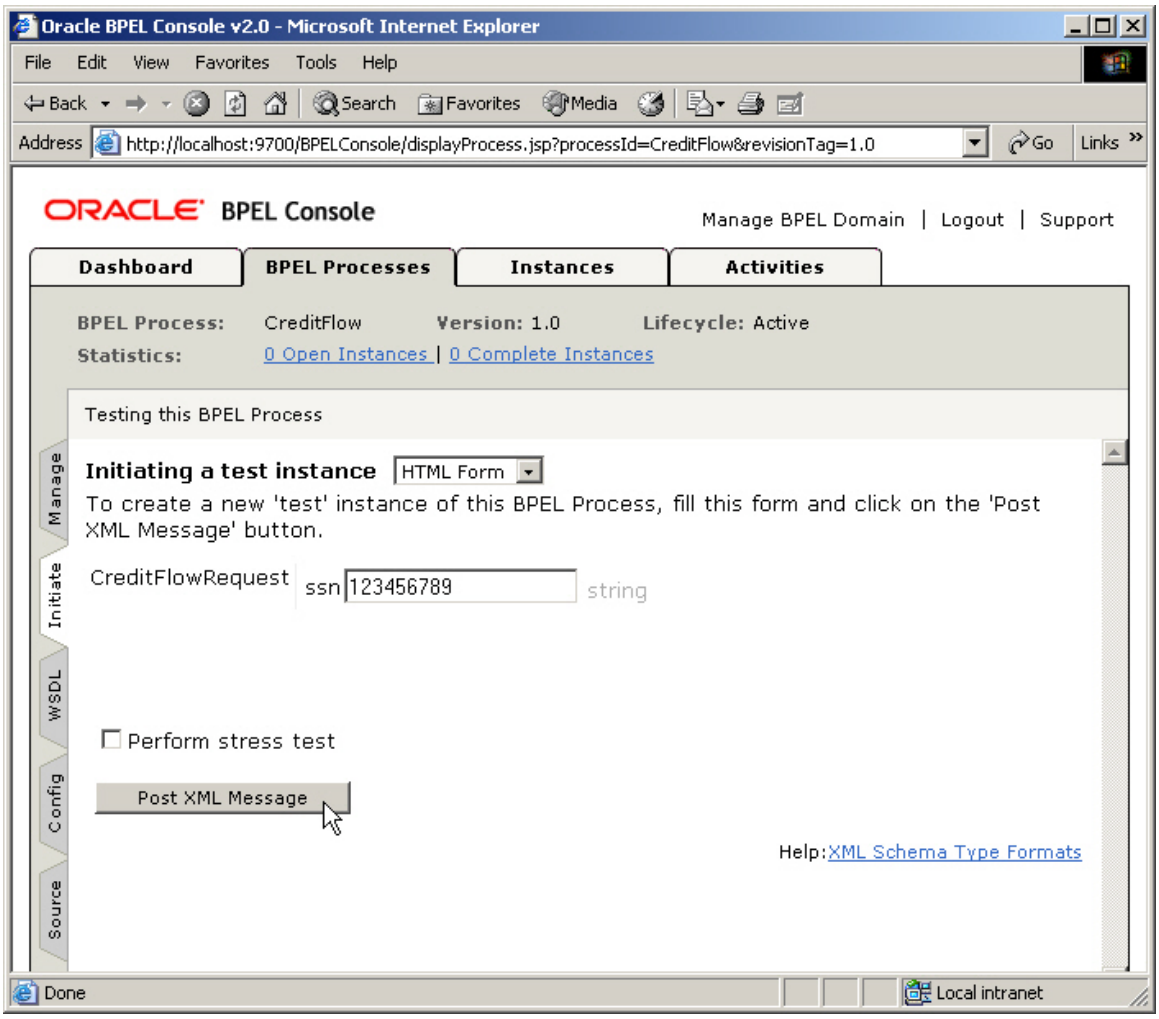


As the final step, you will test your BPEL process by using the automatically generated test interface in the BPEL Console (similar to what you did at the beginning of this tutorial, if you tested the deployed credit rating service).

To initiate a test instance of your BPEL process:

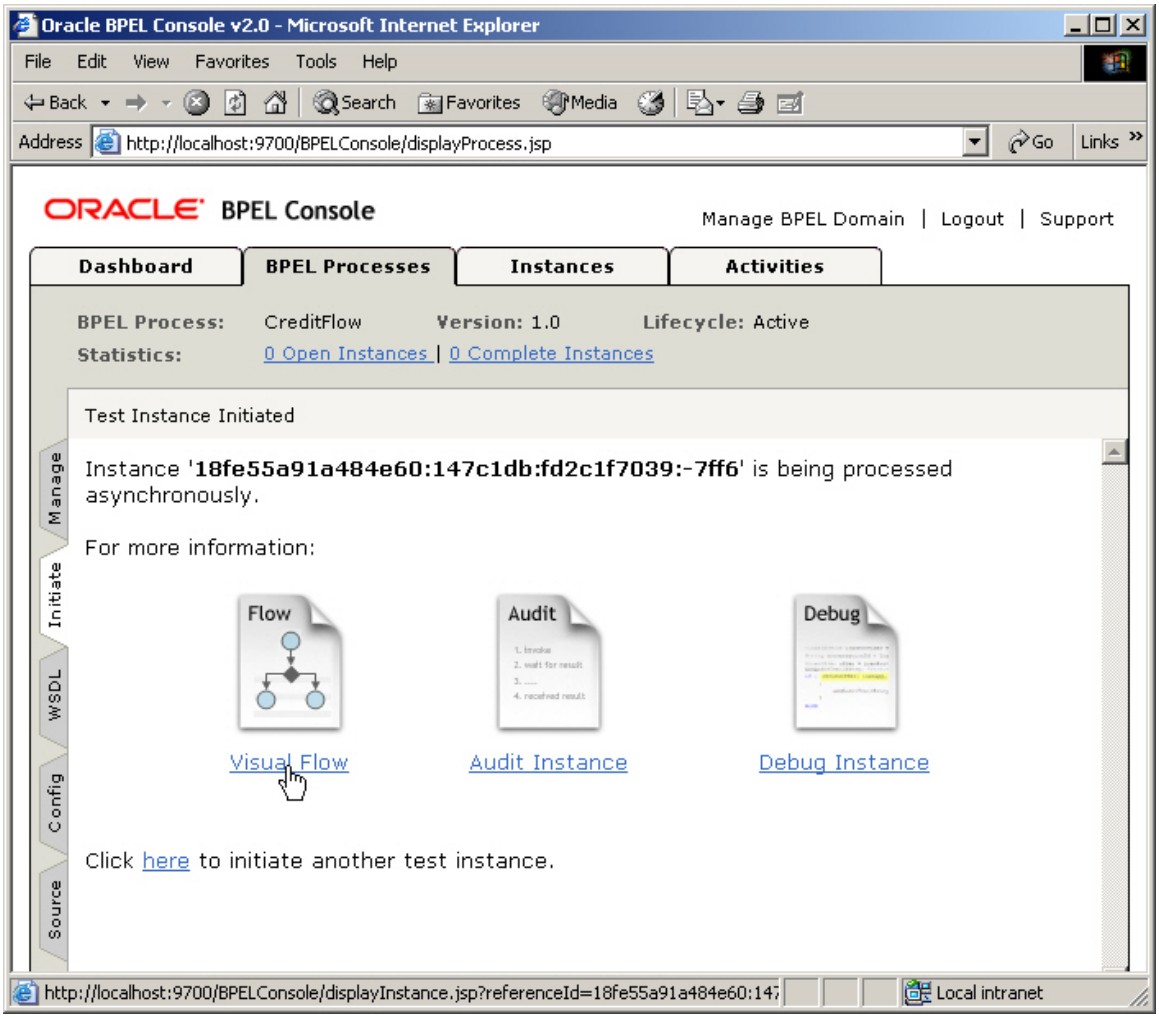
1 Bring the BPEL Console into view and click `CreditFlow` on the Dashboard.

2 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.



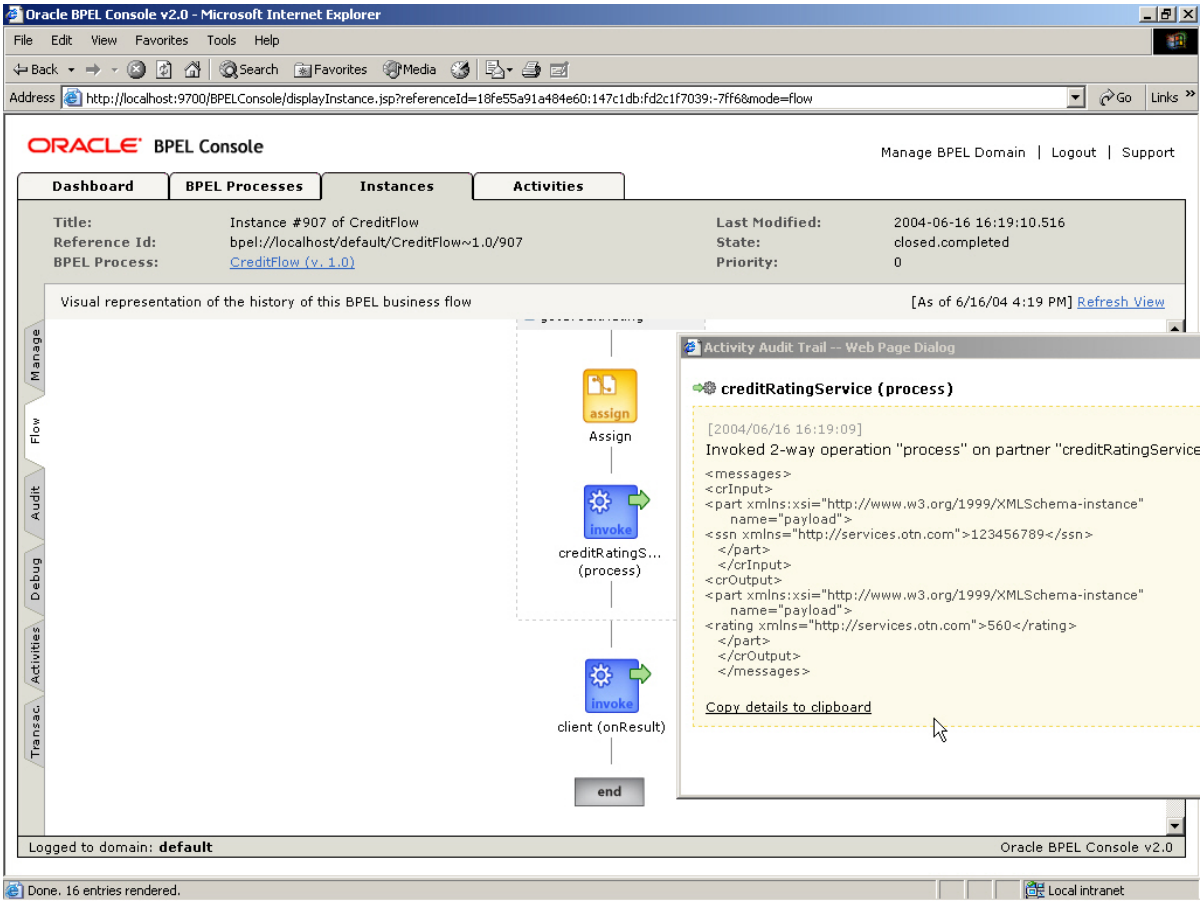
To view the visual audit trail of the instance:

1 Click the **Visual Flow** link to see a visual audit trail representing the current state of the process instance.



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.

2 Click the `creditRatingService` `<invoke>` activity in the audit trail (a few boxes down) to see the messages sent to and received from the credit rating service.



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 `crOutput` variable) into the `creditRating` field for the result of your process itself (the `output` variable). We leave this as an exercise for you.

If you experience any difficulties in completing the flow or have any questions or comments regarding this tutorial, please see http://otn.oracle.com/bpel.