

Adapter Tutorial

Tutorial 1 : Invoking Oracle Applications API through the Oracle AS Adapter

Application Programming Interfaces (APIs) are stored procedures that enable you to insert and update data in Oracle Applications. For example, using APIs, you can insert a customer record in Oracle Applications. The Oracle AS Adapter for Oracle Applications exposes the Oracle Applications API as a Service via WSDL with JCA binding. This tutorial describes how to invoke Oracle Application API interface and insert a customer record. The File Adapter read operation is used to read the customer record from a delimited file and trigger the BPEL process which inturn calls the create customer API in Oracle Applications.

Tutorial 1 : Invoking Oracle Applications API through the Oracle AS Adapter	1
Overview	1
Configuring the Oracle Applications create customer Service	2
Configuring the File Adapter Read Service	4
Understanding the generated Oracle Applications WSDL	9
Understanding the File Adapter WSDL	10
Creating the customer create BPEL process	11
Compiling and deploying the BPEL process	12
Testing the BPEL process	14
Use the Oracle Applications Forms to verify	15
Handling PL/SQL Boolean, PL/SQL Record and REFCursors	16
Troubleshooting	17
Problem	17
Solution	17

Overview

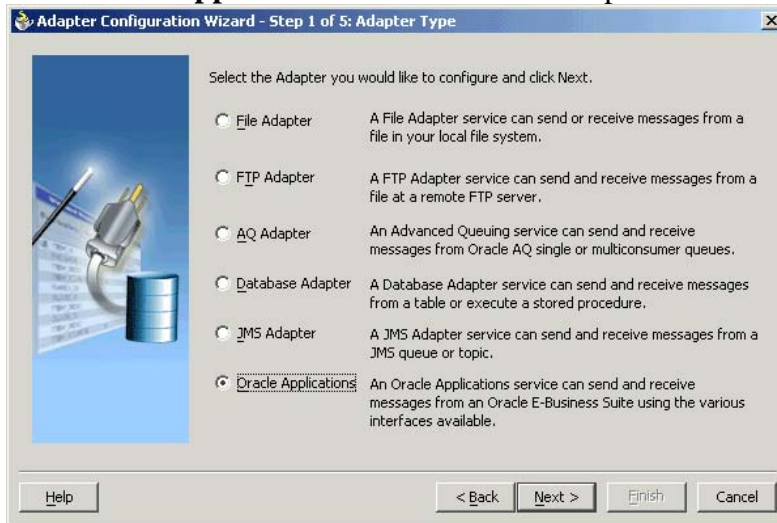
The Oracle AS Adapter for Oracle Applications is part of the BPEL PM 10.1.2 install and is available for both stand-alone as well as mid-tier installations. The Oracle as Adapter for Oracle Applications is a pure JCA 1.5 Resource Adapter and can be deployed on a J2EE container in managed mode. The Adapter Framework (AF) is used for the bidirectional integration of the JCA 1.5 resource adapter with BPEL Process Manager. Adapter FW is based on open standards and employs the Web Service Invocation

Framework (WSIF) technology for exposing the underlying JCA Interactions as Web Services. Both the JCA Inbound and Outbound Interactions are exposed as WSDL with JCA bindings.

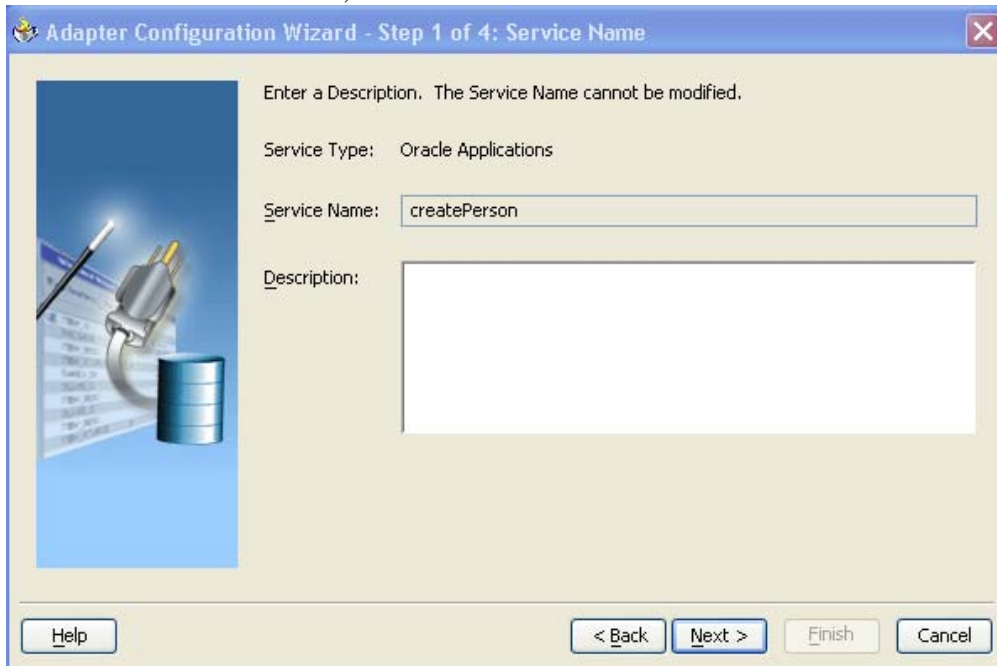
Configuring the Oracle Applications create customer Service

The Jdeveloper based design-time wizard is used for configuration of the Oracle AS Adapters and is launched from the BPEL Partner Link activity. WSDL files are created for both JCA Outbound (Request-Response service – BPEL invoke) and JCA Inbound (Event Notification – BPEL receive) Interactions. The following set of figures illustrates the configuration of the create customer service using the Jdeveloper tool.

Pick **Oracle Applications** from the list of Adapters.



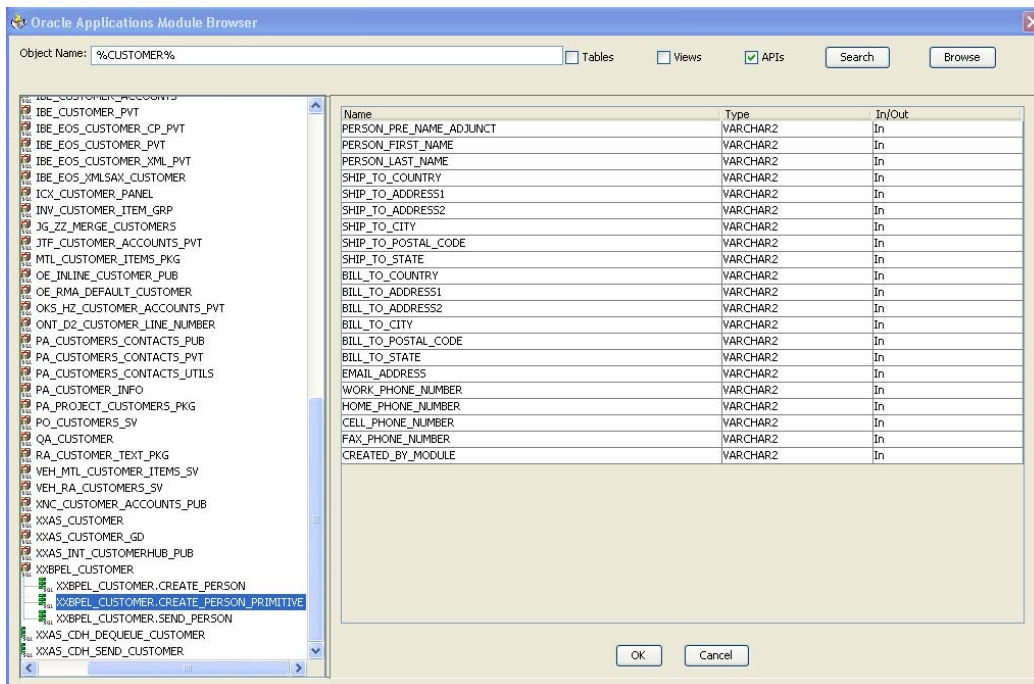
In the **Service Name** field, enter a service name.

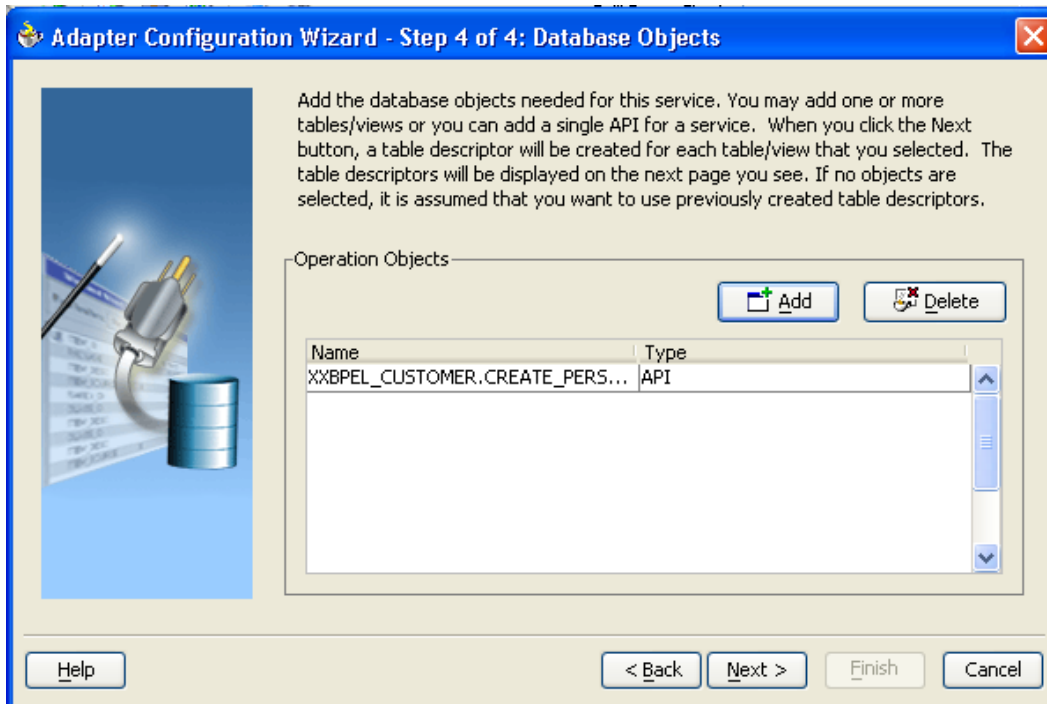


Specify the **connection parameters** as shown in the below figure. The JNDI name refers to the run-time connection and points to the **logical deployment** of the adapter.



Use the **browse** option to select the Customer record. The syntax for browsing is based on SQL 92 standard.

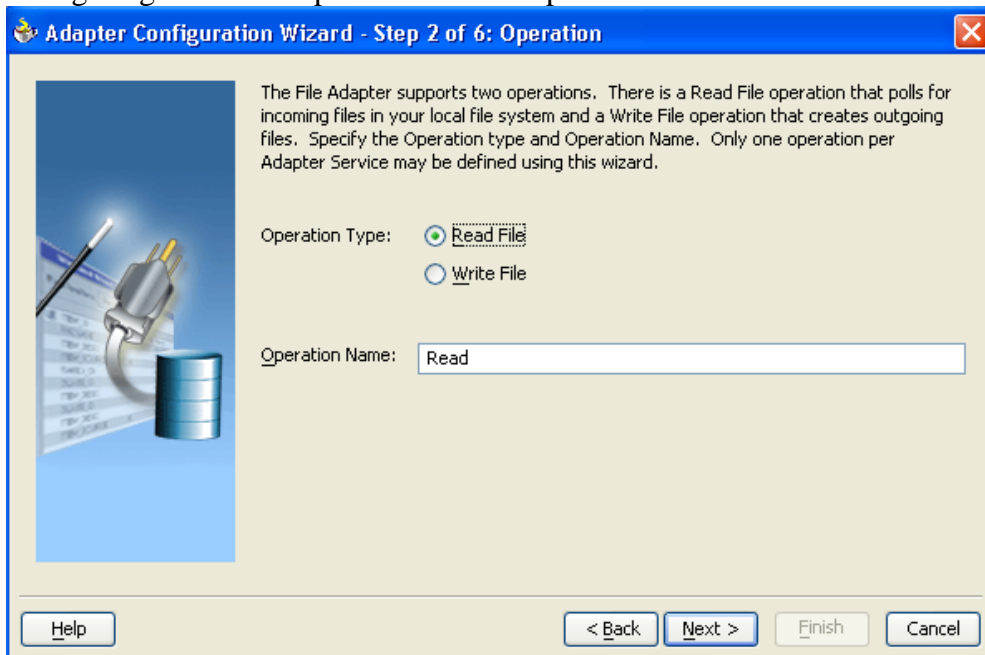


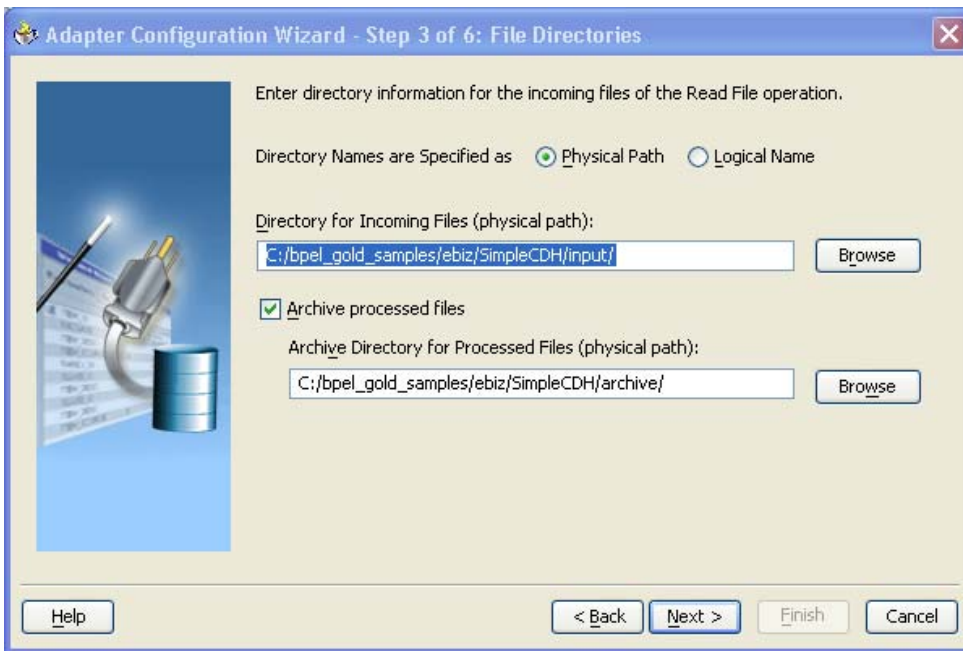


Choose the **XXBPEL_CUSTOMER_CREATE_PERSON_PRIMITIVE** API. This contains only primitive datatypes.

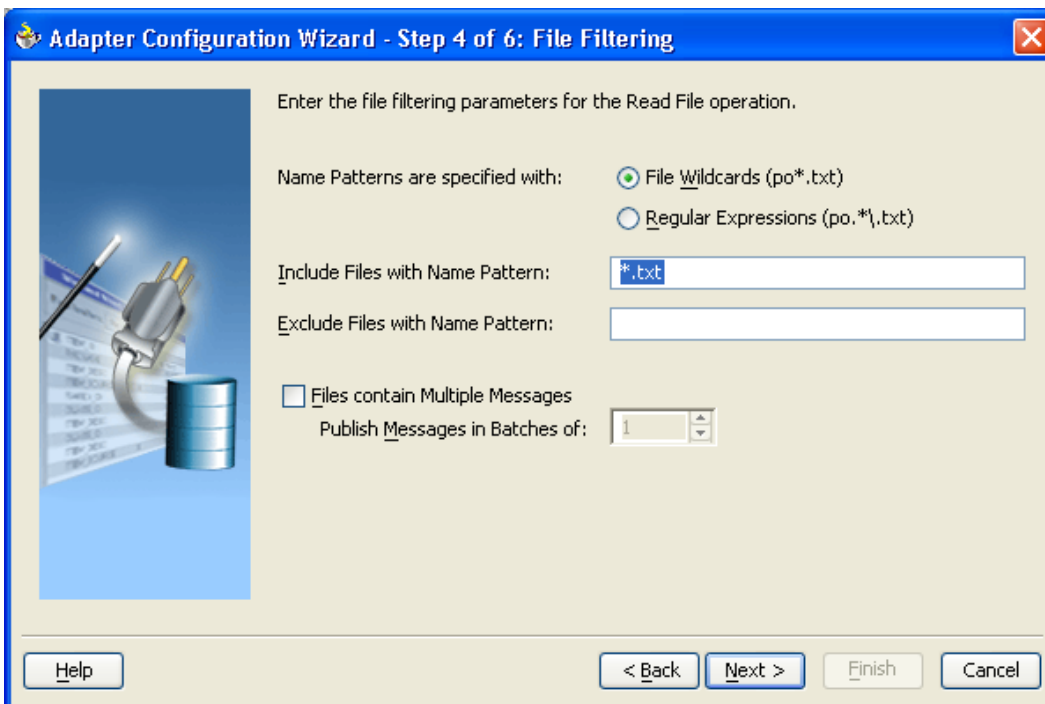
Configuring the File Adapter Read Service

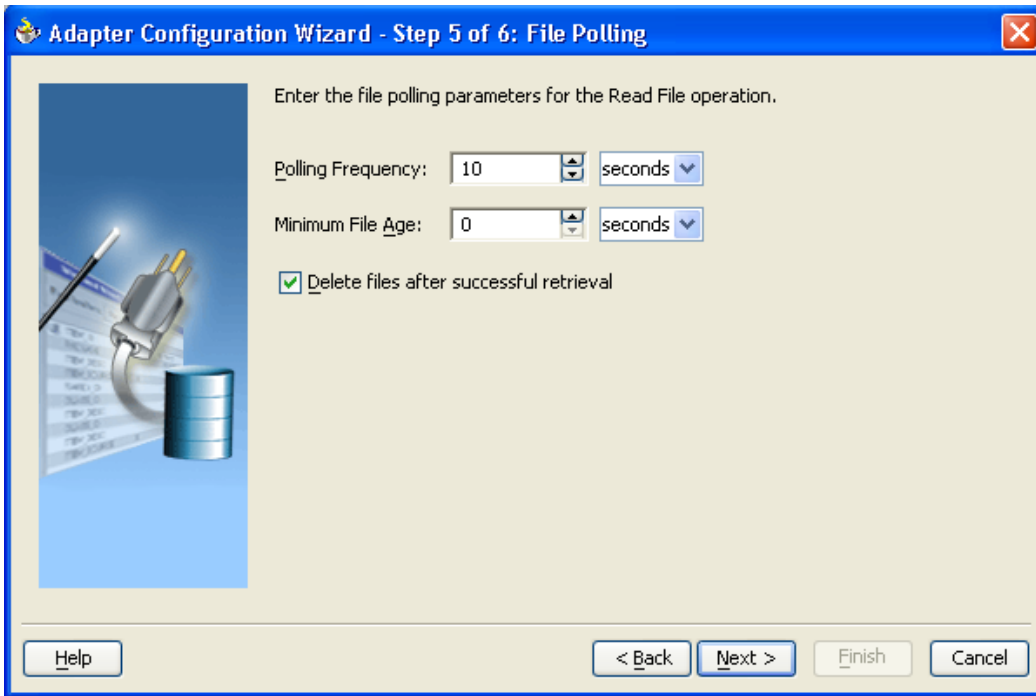
The FileAdapter service in this example is a FileRead operation. It polls the input directory for delimited files with .txt extension, translates the incoming data to an XML message based on the nXSD generated at design-time (using the Native Format Builder Wizard and supplying the sample file) and triggers the BPEL process. The steps for configuring the FileAdapter service are captured below:





Specify the file pattern and file polling characteristics as shown below:

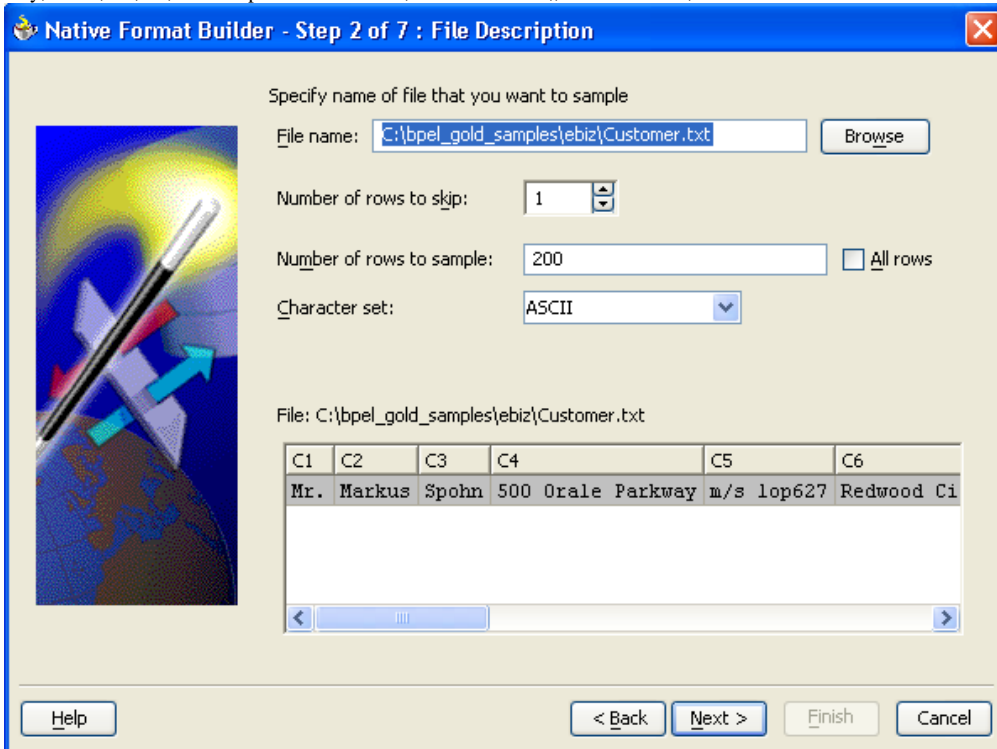




The sample file used for creation of the nXSD is as follows:

Title,FirstName,LastName,ShipToAddressLine1,ShipToAddressLine2,ShipToCity,ShipToPostalCode,ShipToState,ShipToCountry,BillToAddressLine1,BillToAddressLine2,BillToCity,BillToPostalCode,BillToState,BillToCountry,EmailAddress,WorkPhoneNumber,HomePhoneNumber,CellPhoneNumber,FaxNumber

Mr.,Markus,Spohn,500 Orale Parkway,m/s 1op627,Redwood City,94065,CA,US,500 Orale Parkway,m/s 1op627,Redwood City,94065,CA,US,markus.spohn@oracle.com,1-650-506-9977,,-415-806-1405,1-650-506-7365



Native Format Builder - Step 3 of 7 : Record Organization

Specify the file organization in terms of the records that it contains

File contains only one record
 File contains multiple record instances
 Multiple records are of different types
 Multiple records are of single type

File: C:\bpel_gold_samples\ebiz\Customer.txt

C1	C2	C3	C4	C5	C6
Mr.	Markus	Spohn	500 Oracle Parkway	m/s 1op627	Redwood Cit

Native Format Builder - Step 4 of 7 : Specify Elements

Specify target namespace and element names of native format file

Namespace:

Element name specified here will represent a record in native format

Enter name of element containing multiple records:

Enter a name for element that will represent record:

File: C:\bpel_gold_samples\ebiz\Customer.txt

C1	C2	C3	C4	C5	C6
Mr.	Markus	Spohn	500 Oracle Parkway	m/s 1op627	Redwood C:

Native Format Builder - Step 5 of 7 : Specify Delimiters

Records delimited by:

Fields

Delimited by:

Optionally enclosed by:

File: C:\bpel_gold_samples\ebiz\Customer.txt

C1	C2	C3	C4	C5	C6
Mr.	Markus	Spohn	500 Oracle Parkway	m/s 1op627	Redwood Ci

Native Format Builder - Step 6 of 7 : Field Properties

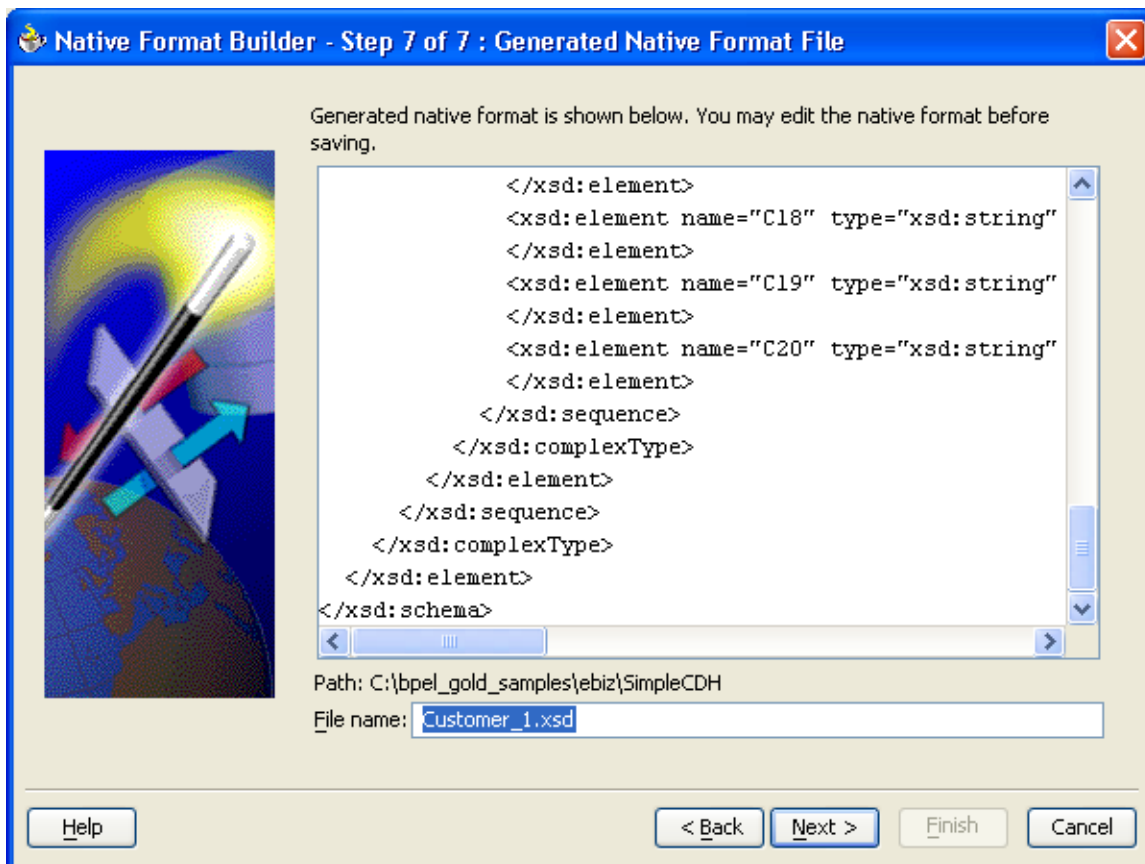
Specify the field names and field properties

Use the first record as the field names

Name	Type	Delimiter
C1	string	Comma (,)
C2	string	Comma (,)
C3	string	Comma (,)
C4	string	Comma (,)
C5	string	Comma (,)
C6	string	Comma (,)
C7	int	Comma (,)
C8	string	Comma (,)

File: C:\bpel_gold_samples\ebiz\Customer.txt

C1	C2	C3	C4	C5	C6
Mr.	Markus	Spohn	500 Oracle Parkway	m/s 1op627	Redwood Ci



Understanding the generated Oracle Applications WSDL

The createPerson.wsdl generated is different from a regular WSDL. It does not contain SOAP artifacts but contains JCA artifacts to invoke the underlying JCA operation. The WSDL JCA extension contains JCA artifacts that are required by Adapter SDK during runtime to translate Web Service messages (from BPEL PM) to JCA Interactions and back. The WSDL JCA binding elements contain the metadata for Adapter SDK to call JCA Outbound Interaction and activate any inbound JCA 1.5 endpoint to receive inbound events. The JCA extension element for the Outbound Interaction contains the JNDI location and InteractionSpec details for calling an Outbound Interaction.

```

<service name="createPerson">
  <port name="createPerson_pt"
    binding="tns:createPerson_binding">
    <!--
      Your runtime connection is declared in
      J2EE_HOME/application-
      deployments/default/AppsAdapter/oc4j-ra.xml
      These mcf properties here are from your design
      time connection and
      save you from having to edit that file and restart
      the application server
      if eis/Apps/apps1 is missing.
      These mcf properties are safe to remove.
    -->

```

```

    <jca:address location="eis/Apps/apps1"
      UIConnectionName="OracleApplications"
      UIOracleAppType="DBOBJECT"
      ManagedConnectionFactory="oracle.tip.adapter.apps.
      AppsManagedConnectionFactory"
      mcf.ConnectionString="jdbc:oracle:thin:@la2037.orac
      leads.com:1521:la2037" mcf.UserName="apps"
      mcf.Password="53CB0F044A0D3DD2C063679F18F89
      870" />
  </port>
</service>

```

The WSDL extension has 2 parts : a service part defining the address or location of a service and the binding part that defines the implementation of the service. The above figure points to the **<service>** element. The **<jca:address>** tag contains the JNDI location of the ManagedConnectionFactory of the JCA 1.5 Resource Adapter and has to be match with one of the **<connector-factory>** elements of the corresponding oc4j-ra.xml. The WSDL **<service>** element also contains the design-time parameters. This is used for logical deployment of the JCA 1.5 Resource Adapter in the non-managed mode if there is no associated **<connector-factory>** element present in the oc4j-ra.xml for the JNDI run-time connection. The Adapter defaults to the non-managed mode of operation and the connection management is done by the Adapter and not by the OC4J container. This method of operation is not recommended for production situations as the Adapter connection management is not as sophisticated and robust when compared to the OC4J connection management. The non-managed mode of operation is suitable for development and testing purposes only.

```

<binding name="createPerson_binding"
  type="tns:createPerson_ptt">
  <jca:binding />
  <operation name="createPerson">
    <jca:operation SchemaName="APPS"
      PackageName="XXBPEL_CUSTOMER"
      ProcedureName="CREATE_PERSON_PRIMITIVE"
      InteractionSpec="oracle.tip.adapter.apps.AppsStored
      ProcedureInteractionSpec" />
    <input />
  </operation>
</binding>

```

The **<binding>** element in the above WSDL defines the JCA interactions and contains the **<jca:operation>** element. The **<jca:operation>** element has contains the InteractionSpec classname and the name-value pairs for the InteractionSpec parameters. The Adapter SDK creates an InteractionSpec Java bean and calls the appropriate JCA Outbound Interaction method.

Understanding the File Adapter WSDL

The ReceiveData.wsdl generated is different from a regular WSDL. It does not contain SOAP artifacts but contains JCA artifacts to raise the JCA Inbound Interaction event to the BPEL process receive activity. The WSDL JCA binding elements contain the metadata for Adapter SDK to activate any inbound JCA 1.5 endpoint to receive inbound events. The JCA extension element for the Inbound Interaction usually contains the JCA Resource Adapter class name and the ActivationSpec class. However, for technology and Oracle Application Adapters the JCA extension contains the JNDI location of the ManagedConnectionFactory class instead of the Resource Adapter class name. The WSDL <service> element for the ReceiveData.wsdl is shown below.

```

<service name="ReceiveData">
  <port name="Read_pt" binding="tns:Read_binding">
    <jca:address location="eis/FileAdapter" UIincludeWildcard="*.txt" />
  </port>
</service>

```

The WSDL <binding> element for the ReceiveData.wsdl is shown below. This contains the ActivationSpec classname and the name-value parameters for creating the ActivationSpec class.

Note: Two FileAdapter endpoints cannot point to the same Input Directory. The File Adapter does not do duplicate detection of endpoints and the behavior is undeterministic.

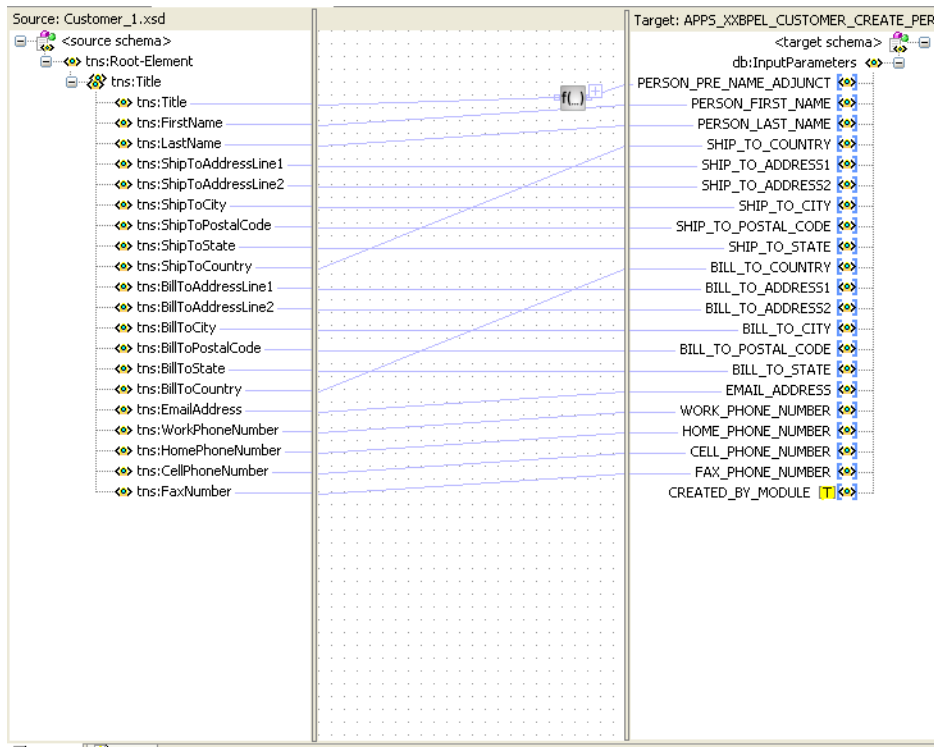
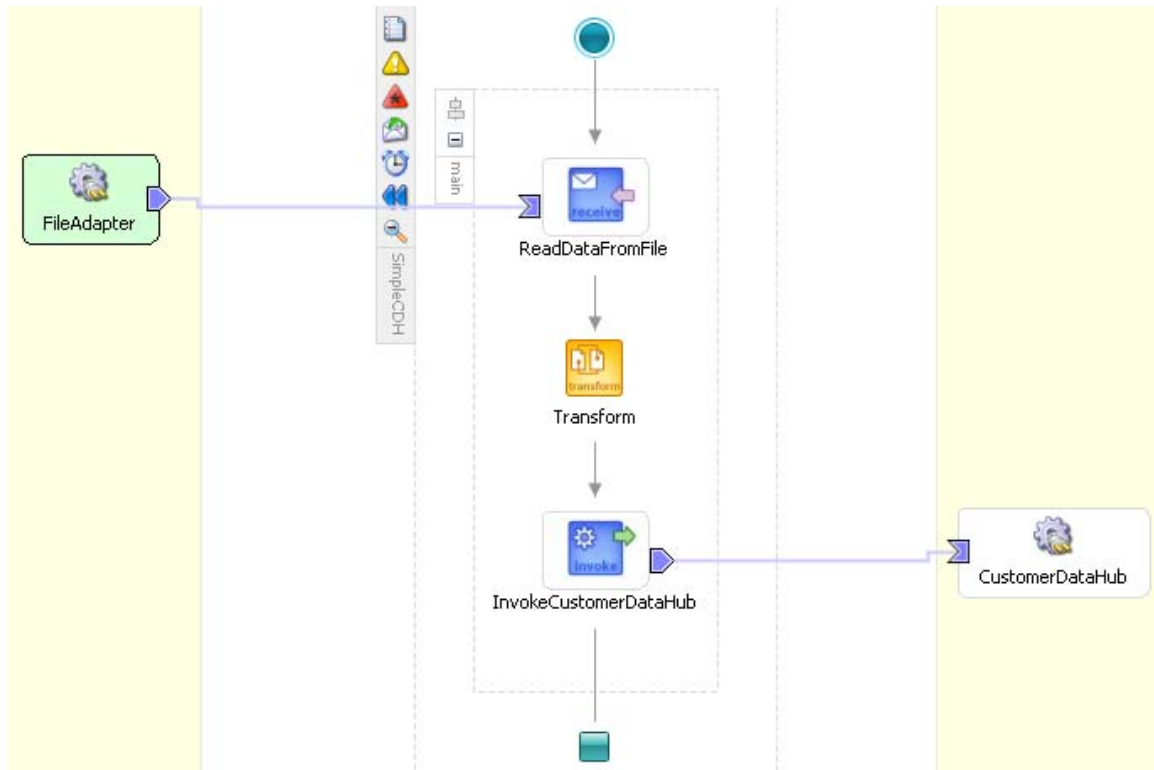
```

<binding name="Read_binding" type="tns:Read_ptt">
  <pc:inbound_binding />
  <operation name="Read">
    <jca:operation PhysicalDirectory="C:/bpel_gold_samples/ebiz/SimpleCDH/input/"
      ActivationSpec="oracle.tip.adapter.file.inbound.FileActivationSpec"
      PhysicalArchiveDirectory="C:/bpel_gold_samples/ebiz/SimpleCDH/archive/"
      IncludeFiles="*\.txt"
      PollingFrequency="10"
      MinimumAge="0"
      DeleteFile="true"
      OpaqueSchema="false" />
    <input>
      <jca:header message="hdr:InboundHeader_msg"
        part="inboundHeader" />
    </input>
  </operation>
</binding>

```

Creating the customer create BPEL process

Drag and drop a BPEL receive activity to the FileAdapter partner link. Select the “createInstance” option. Drag and drop a BPEL invoke activity to the CustomerDataHub partner link. Drag & drop a transform activity to map the FileAdapter message to the Oracle Applications Adapter API request message.

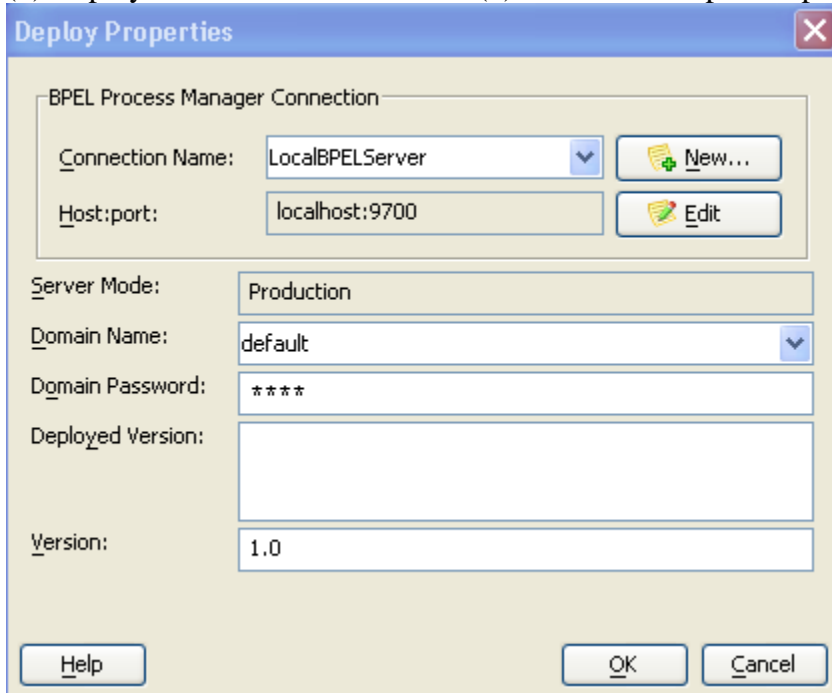


Compiling and deploying the BPEL process

Build the BPEL process.

```
Messages | BPEL Messages | Mapper Messages
Internal compilation error, terminated with a fatal exception
Internal compilation error, terminated with a fatal exception
Compiling...
Compiling C:\bpel_gold_samples\ebiz\SimpleCDH\SimpleCDH.bpel
[BPEL Compiler] Initializing compiler for first time use...
BPEL suitcase generated in: C:\bpel_gold_samples\ebiz\SimpleCDH\output\bpel_SimpleCDH_v2005_04_26_39616.jar
[11:00:31 AM] Successful compilation: 0 errors, 0 warnings.
Compiling...
Compiling C:\bpel_gold_samples\ebiz\SimpleCDH\SimpleCDH.bpel
BPEL suitcase generated in: C:\bpel_gold_samples\ebiz\SimpleCDH\output\bpel_SimpleCDH_v2005_04_26_46913.jar
[1:01:56 PM] Successful compilation: 0 errors, 0 warnings.
```

Deploy the BPEL process. There are 3 ways to perform this (1) Deploy from Jdeveloper (2) Deploy from BPEL Console and (3) Use obant script to deploy.



```
Messages | BPEL Messages | Mapper Messages
Beginning Deployment Process...
Compiling C:\bpel_gold_samples\ebiz\SimpleCDH\SimpleCDH.bpel
Compiling...
BPEL suitcase generated in: C:\bpel_gold_samples\ebiz\SimpleCDH\output\bpel_SimpleCDH_1.0.jar
[1:06:20 PM] Successful compilation: 0 errors, 0 warnings.
Deploying to http://localhost:9700 domain: default. Please wait....
[1:06:22 PM] bpel_SimpleCDH_1.0.jar deployed successfully .
```

For the above example, you need to have a matching <connector-factory> element for the JNDI run-time connection. The creation of a <connector-factory> element and associating it with a JNDI name is referred to as logical deployment of the Adapter. The OC4J container instantiates a ManagedConnectionFactory class and associated it with the JNDI name specified for every <connector-factory> element. This is also referred to as managed mode of JCA Adapter and the OC4J container manages the connections of the Resource Adapter in this case. The oc4j-ra.xml file for the Oracle Applications Adapter is shown below.

```
<?xml version="1.0"?>
<!DOCTYPE oc4j-connector-factories PUBLIC "-//Oracle//DTD Oracle Connector 9.04//EN"
"http://xmlns.oracle.com/ias/dtds/oc4j-connector-factories-9_04.dtd">

<oc4j-connector-factories>
  <connector-factory location="eis/Apps/apps1"
    connector-name="Oracle Applications Adapter">
    <config-property name="connectionString"
      value="jdbc:oracle:thin:@la2037.oracleads.com:1521:la2037"/>
    <config-property name="userName" value="apps"/>
    <config-property name="password" value="apps"/>
    <config-property name="usesExternalConnectionPooling" value="false"/>
    <config-property name="dataSourceName" value=""/>
    <config-property name="usesExternalTransactionController" value="false"/>
  </connector-factory>
</oc4j-connector-factories>
```

The oc4j-ra.xml for the File Adapter looks as below:

```
<?xml version="1.0"?>
<!DOCTYPE oc4j-connector-factories PUBLIC "-//Oracle//DTD Oracle Connector 9.04//EN"
"http://xmlns.oracle.com/ias/dtds/oc4j-connector-factories-9_04.dtd">

<oc4j-connector-factories>
  <connector-factory location="eis/FileAdapter" connector-name="File Adapter">
  </connector-factory>
</oc4j-connector-factories>
```

Testing the BPEL process

The BPEL process in this example can be tested by placing an input file in the input polling directory of the File Adapter. The file gets picked up, translated into an XML message and fed as input to the Oracle Applications Adapter service. The Audit tab shows the message received from the File Adapter and the message sent to the Oracle Applications Adapter.

Log into the Oracle Applications Web Page.

After logging in you'll be routed to the home page

- Click on '**Customer Home**'
- In the '**Search**' section, pick '**Person**' from the picklist and enter the name of the person.
- Hit '**Search**'

The screenshot shows the Oracle Applications web interface. At the top, there's the Oracle logo and 'Customers Online' header. A navigation bar includes 'Home', 'Customers', 'Reports', and 'Administration'. Below this, a sub-navigation bar shows 'Organizations', 'Persons', and 'Hierarchies'. The main content area is titled 'Jack Turner: Overview' and features a 'Switch to' dropdown menu set to 'Jack Turner' with a 'Go' button. A table displays 'Primary Information' for Jack Turner, with fields for Name, Pronunciation, Alias, Registry ID, Status, Primary Address, Primary Phone, Primary Email, Gender, Date of Birth, and Tax ID. A 'Visualize Relationships' button is located at the bottom right of the page.

Handling PL/SQL Boolean, PL/SQL Record and REFCursors

These datatypes are not supported by the Oracle JDBC Provider and hence requires writing wrappers that inturn call the actual Procedures.

For **PL/SQL Boolean** you need to do the following:

The solution is to define a wrapper procedure that substitutes a supported datatype, such as INTEGER, for each BOOLEAN parameter. For example, suppose you defined the following stored procedure:

```
CREATE PROCEDURE BOOLPROC (B BOOLEAN) AS BEGIN...END;
```

A wrapper procedure can be written as follows:

```
CREATE PROCEDURE BOOLWRAP (X INTEGER) AS  
BEGIN  
IF (X = 1) THEN BOOLPROC (TRUE);  
ELSE BOOKPROC (FALSE);  
END IF;  
END;
```

For **PLSQL Record** type refer to the example:

[<BPEL_HOME>\integration\orabpel\samples\tutorials\122.DBAdapter\JpublisherWrapper](#)

For **REFCURSOR** refer to the example:

<BPEL_HOME>\integration\orabpel\samples\tutorials\122.DBAdapter\ResultSetConverter

Troubleshooting

Problem

Exception Description: java.sql.SQLException: Io exception: The Network Adapter could not establish the connection

Internal Exception: java.sql.SQLException: Io exception: The Network Adapter could not establish the connection

Error Code: 17002.

Solution

Ensure that the connection parameters specified in the oc4j-ra.xml is right. You need to bounce the OC4J container for any changes made to the oc4j-ra.xml to take effect.