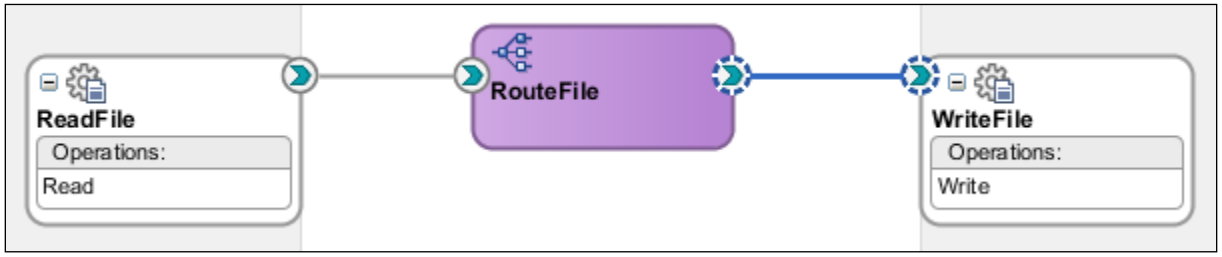
Practices for Lesson 4: Working with Data

Practices for Lesson 4: Overview

Practices Overview

In this practice, you create, deploy, and run a composite application that receives delimited data from a File adapter. This data is transformed to an XML format and passed to a Mediator. The Mediator transforms the data to a second XML format, and routes the file to an outbound File adapter, which passes the data from the composite application to the file system.



Practice 4-1: Creating the Composite Application

Overview

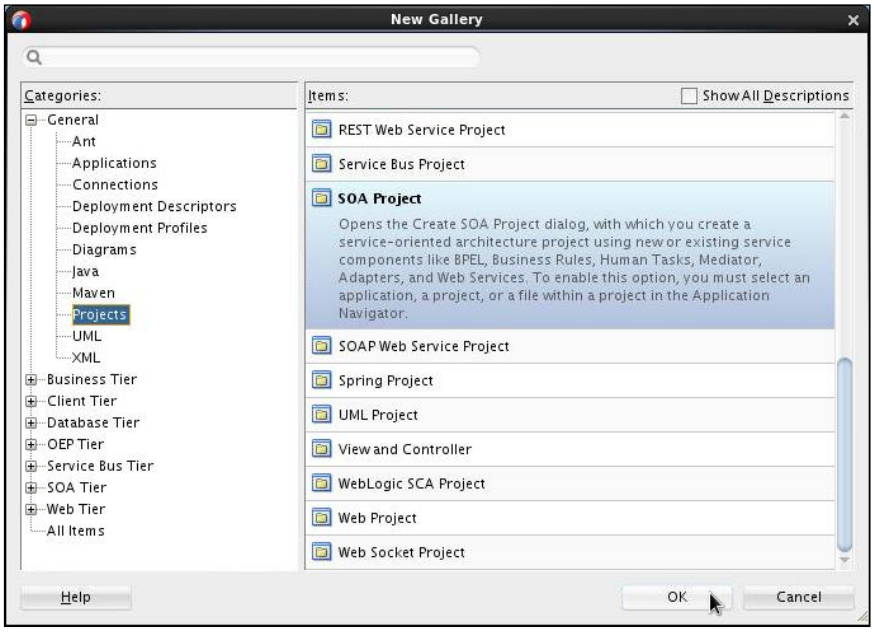
In this section of the practice, you create a new composite application and add an external service interface.

Assumptions

This practice assumes that you have started JDeveloper.

Tasks

1. Create a new SOA project named `NativeData`.
   a. In the JDeveloper Application Navigator menu, select New > Project.
      The New Gallery window appears.
   b. Select SOA Project and click OK.



      The Create SOA Project wizard appears.
   c. Name the project `NativeData` and click Next.
   d. Verify that the Empty Composite template is selected and click Finish.
      The composite overview window is opened.
      Note: Remember that it may take a minute or more to generate the new project.
   e. Save your work.

Creating and Configuring the Exposed Service Interface

This interface describes how a client can call the composite application.

2. Drag a File adapter from the Component Palette into the Exposed Services column.
   The FILE adapter configuration wizard opens.
3. To configure the File adapter, use the instructions listed in the following table:

Step	Window Description	Choices or Values
a.	File Adapter Service	Service Name: `ReadFile` Click Next.
b.	Adapter Interface	Accept the default values. Click Next.
c.	File Server Connection	Accept the default values. Click Next.
d.	Operation Type	Select the Read File radio button. Click Next.
e.	File Directories	Use the Browse button to specify the Directory for Incoming Files: `/home/oracle/labs/input` Deselect Process Files Recursively. Select Archive Processed Files. Use the Browse button to specify the Archive Directory: `/home/oracle/labs/archive` Click Next.
f.	File Filtering	Include Files with Name Pattern: `po*.dat` Click Next.
g.	File Polling	Polling Frequency: 10 seconds Click Next.
h.	Messages	Click the “Define Schema for Native Format” (gear) icon.

   The Native Format Builder opens.

   Note: So far, we have specified that the File adapter is to poll the directory `/home/oracle/labs/input` every 10 seconds. If it finds any files that match the specification `po*.dat`, it moves them to the `/home/oracle/labs/archive` directory. It also passes a copy of the file to the composite application that we are about to build.

   Continue with Practice 4-2 to define the native data format of the incoming data.

Practice 4-2: Defining the Native Data Format

Overview

In this section of the practice, you define the incoming native data format within an XSD so that the incoming adapter can reformat the incoming message as XML. The incoming message contains end-delimited purchase order data. The first line of the file includes customer and payment data associated with the purchase itself. That line of data is followed by a variable number of lines of data that describe each of the items in the order. The following example includes two item records:

```
2,100,credit,two_day,initial,VISTA,1234-1234-1234-1234
SKU301|Music Player 1Gb|45|3
SKU305|Music Player 120Gb|250|20
```

Assumptions

This practice assumes that you have completed Practice 4-1, and that the Native Format Builder is still open.

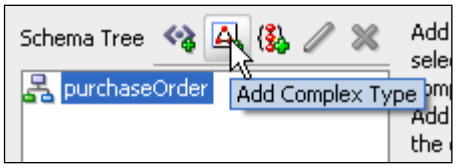
Tasks

1. To configure the handling of native data, perform the steps in the following table:

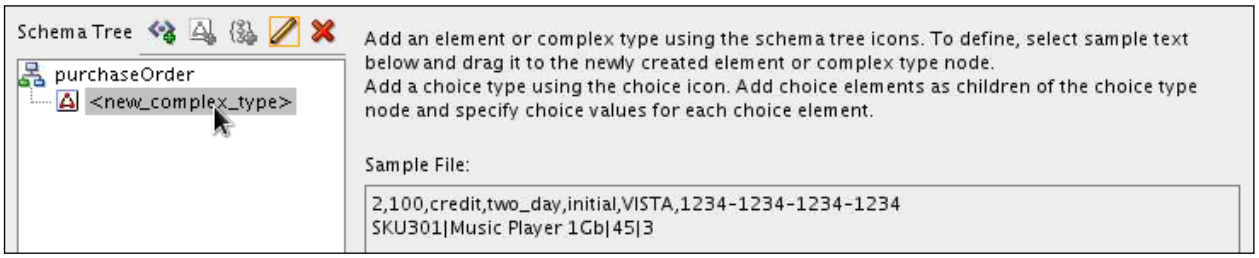
Step	Window	Choices or Values
a.	Welcome	Click Next.
b.	File Name and Directory	File Name: `nxsd_po_schema.xsd` Click Next.
c.	Choose Type	Select Complex Type. Click Next.
d.	File Description	File Name: `/home/oracle/labs/files/xsd/purchaseOrder.dat` Number of rows to sample: 2 Root element: `purchaseOrder` Click Next.

   The Design Schema pane opens.

2. Create and configure a complex type to describe customer data.
   a. Add a complex type to the schema.



b. Highlight the sample data that corresponds to the complex type (the first line of the sample file). Drag and drop the sample data onto the <`new_complex_type`> component of the schema tree.



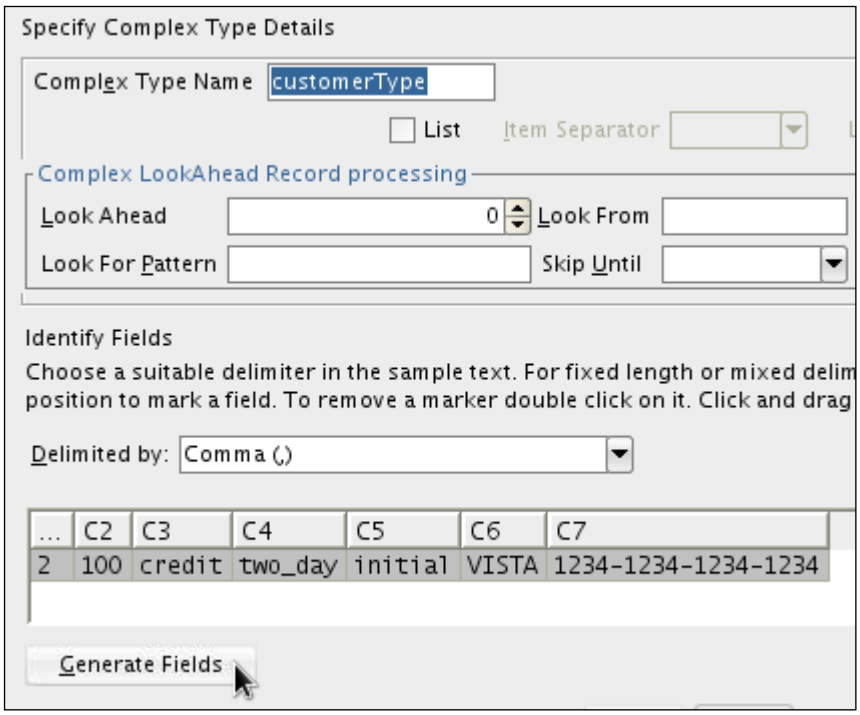
The Complex Type Details pane opens.

c. Name the complex type `customerType`.

d. Specify that the data is delimited by Comma (,).

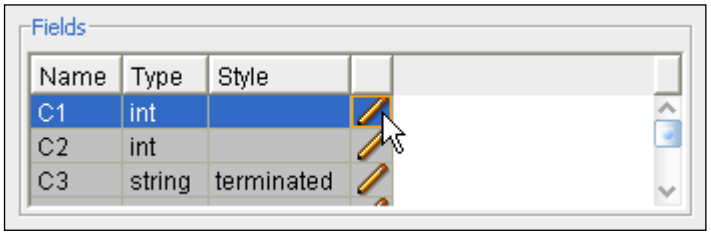
The sample data is parsed.

e. Click Generate Fields.



The wizard generates a series of fields based on the sample data and the parameters provided thus far. The properties of each of the fields must be edited manually.

f. Click the Pencil icon to edit the first field.



The field properties editor is displayed.

g. Use the data in the following table to update the Field Name, Type, Style, and Terminated By fields as needed for the complex type `customerType`.

Field Name	Type	Style	Terminated by
`custID`	string	terminated	,
`ID`	string	terminated	,
`payOption`	string	terminated	,
`shipChoice`	string	terminated	,
`status`	string	terminated	,
`ccType`	string	terminated	,
`ccNumber`	string	terminated	${eol}

h. Click OK to close the Complex Type Details window.

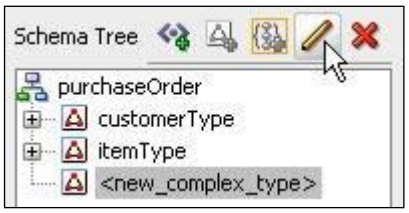
3. Create and configure a complex type to describe item data.
   a. Highlight the schema root node (`purchaseOrder`).
   b. Add another complex type to the schema.
   c. Highlight the sample item data that corresponds to the complex type (the second line of the file). Drag and drop it onto the <`new_complex_type`> component of the schema tree.
   d. Name the complex type `itemType`.
   e. Specify that the data is delimited by the pipe character |. (Enter the pipe character into the “Delimited by” field.)

      The sample data is parsed.
   f. Click Generate Fields.
   g. Use the information in the following table to edit the individual field properties:

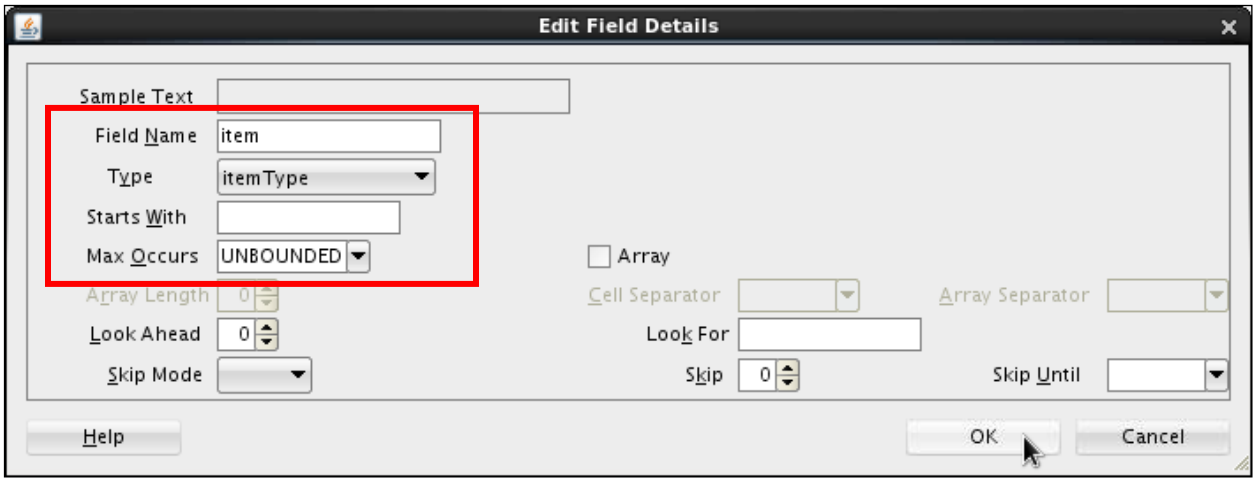
Field Name	Type	Style	Terminated by
`prodID`	string	terminated	|
`prodName`	string	terminated	|
`price`	decimal	terminated	|
`quantity`	int	terminated	${eol}

   h. Click OK to close the Complex Type Details window.

4. Create and configure a complex type to describe the list of item data.
   a. Add a complex type to the schema tree.
   b. Click the Pencil icon to edit the new complex type properties.



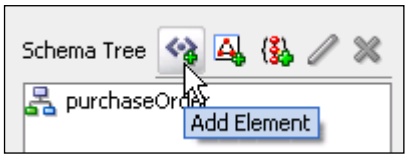
   c. Name the new type `itemListType`.
   d. Edit the field C1 to define an element named `item` of type `itemType`.
   e. Set the `maxOccurs` property of `item` to UNBOUNDED.



Note: The `UNBOUNDED` value specifies that an unlimited number of `item` records can be included in the data.

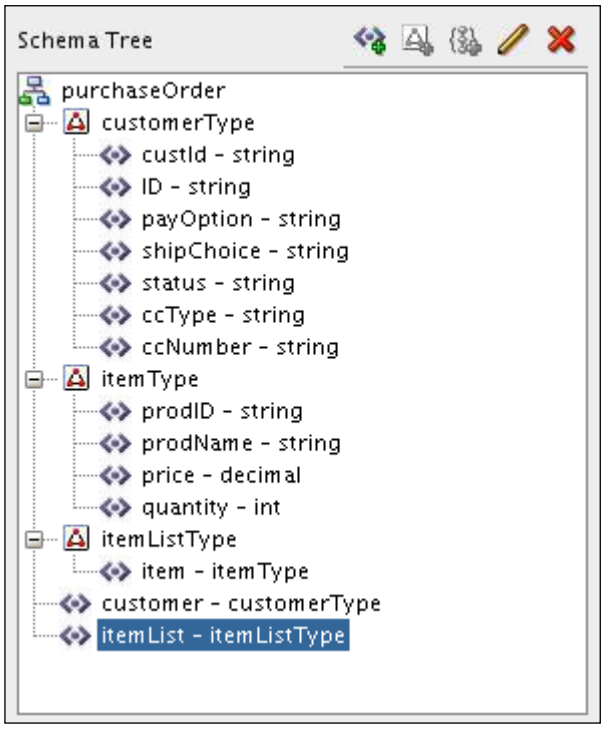
   f. Click OK (twice).

5. Add the following two *elements* (not complex types) to `purchaseOrder` and configure them:

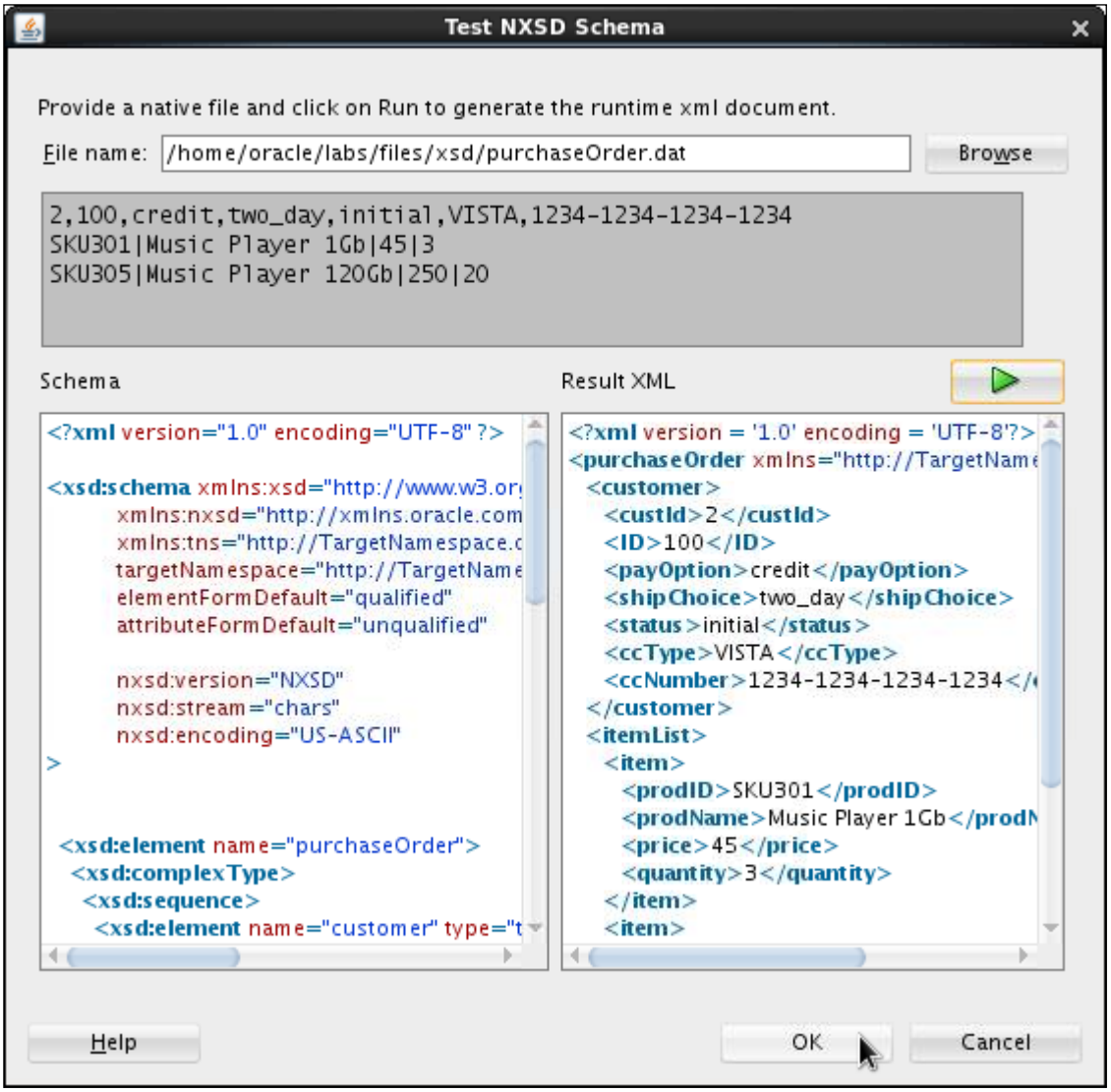


Name	type	maxOccurs
`customer`	customerType	1
`itemlist`	itemListType	1

6. Verify your work.



7. Click Next.
8. Click Test.
9. Provide the native file name `/home/oracle/labs/files/xsd/purchaseOrder.dat`.
10. Click the test (green right arrow) button.
    The sample data is mapped to an XML format.
11. Verify your results.



12. Click OK.

    The Test NXSD Schema window closes.

13. Click Next, and then Finish.

    The Native Format Builder closes.

14. Click Next, and then Finish.

    The File Adapter Configuration wizard closes.

15. Save your work.

Practice 4-3: Creating the Service Components and External References

Overview

In this section of the practice, you add a Mediator service component, and another File adapter as an external reference (to write the outbound message).

Assumptions

This section of the practice assumes that you have completed Practice 4-2 successfully.

Tasks

1. Create a Mediator component.
   a. In the Composite Editor, drag a Mediator component from the Component Palette into the Components column.

      The Create Mediator dialog box opens.
   b. Set the name to `RouteFile`.
   c. Accept the Define Interface Later template. Click OK.

      Note: The interface is automatically defined in the next step, when the wire is created from the exposed service to the Mediator.
   d. Create a wire from the ReadFile exposed service icon to the RouteFile Mediator icon.
   e. Save your work.
2. Create and configure a File adapter as an external reference.
   a. Add a File adapter to the External References column.

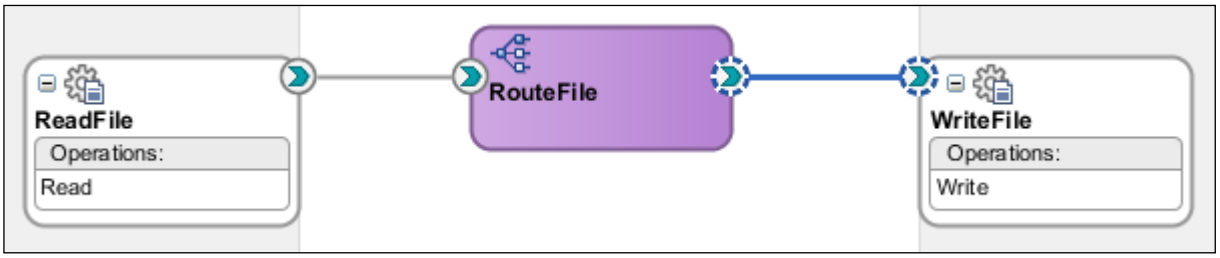
      The FILE Adapter Configuration wizard opens.
   b. Use the instructions in the following table to configure the File adapter:

Step	Window	Choices or Values
a.	File Adapter Reference	Name: `WriteFile` Click Next.
b.	Adapter Interface	Accept “Define from operation and schema (specified later)”. Click Next.
c.	File Server Connection	Accept the File Server JNDI name: `eis/FileAdapter` Click Next.
d.	Operation Type	Select the Write File option. Click Next.
e.	File Configuration	Directory for Outgoing Files: `/home/oracle/labs/output/podata`

Step	Window	Choices or Values
		File Naming Convention: `po_%yyMMddHHmmss%.xml` Click Next.
f.	Messages	Click the “Browse for schema file” icon.
g.	Type Chooser	Click the Import Schema icon.
h.	SOA Resource Browser	With File System selected, navigate to the `/home/oracle/labs/files/xsd` folder and select `internalorder.xsd`. Click OK.
i.	Localize Files	Click OK.
j.	Type Chooser	Expand the Project Schema Files > `internalorder.xsd` entry (if needed), and select `order`. Click OK.
k.	Messages	Click Next.
l.	Finish	Click Finish.

The Adapter Configuration wizard closes.

3. Create a wire from the Mediator component to the File adapter.
4. Verify and save your work.



Note: JDeveloper may list a number of error and warning messages during the creation of your project. It performs validation checks each time you save, and these errors and warnings simply indicate that the project is not fully configured. Error messages should be taken more seriously if they are seen when you attempt to deploy your project.

5. **Things to Consider:** What files were created when you added the Mediator? The File adapter? What does each of the files contain?

   Note: If you are not sure, refer to the slides titled “Creating a Mediator” and “Creating a File Adapter” in the lesson titled “Getting Started with Composite Applications.”

Practice 4-4: Adding a Routing Rule to the Mediator

Overview

In this section of the practice, you configure the Mediator by adding a routing rule. This rule includes a transformation to map the incoming message format to the outgoing message format. You use the Auto Map feature, use XPath functions to perform type conversions, and create a data dictionary.

Assumptions

This practice assumes that you have completed Practice 4-3 successfully.

Tasks

1. Right-click the `RouteFile` Mediator and select Edit.

   The Mediator configuration editor opens.
2. Click the “Select an existing mapper file or create a new one” icon next to the Transform Using field.

   The Request Transformation Map dialog box appears.
3. Click the Create Mapping icon.

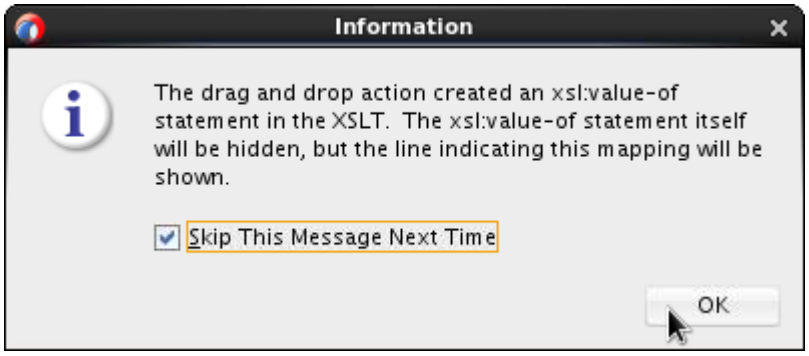
   The Create Transformation Map dialog box appears.
4. Click OK.
5. Click OK to close the Request Transformation Map dialog box.

   The XSLT Mapper (`purchaseOrder_To_order1.xsl`) window opens.
6. Drag the PurchaseOrder element in the source column and drop it onto the Order element in the target column.

   The Auto Map Preferences window opens.
7. Click OK to accept the defaults.

   The Auto Map feature finds no matches. (Points to Consider: Why was the source node `status` not mapped to the target node `status`?)
8. Manually map the leaf nodes. Use the table that follows if needed. (Note that there is no mapping to the target node `orderTotal`.)

   The first time you map a field, the following information dialog box appears. Select the Skip This Message Next Time check box to avoid repeated display of this message.



Source	Target
custID	customerId
ID	orderId
payOption	payMethod
shipChoice	shipMethod
status	status
ccType	cardType
ccNumber	cardNumber

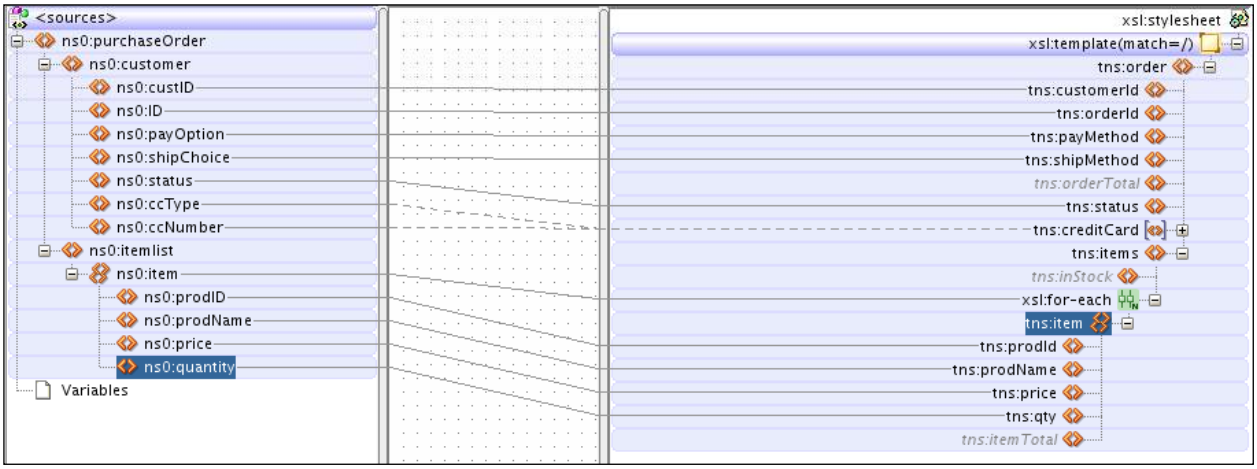
9. Map the node `itemlist` to the node `items`.

   The Auto Map Preferences window appears again.

10. Click OK to accept the defaults.

    Note: Something very different happens this time. First, Auto Map finds several matches. (Point to Consider: *What is different this time*?) Secondly, a `for-each` node is added to the target message structure, and the repeating element `item` is mapped to it. The `for-each` node is a graphic representation of an XSL construct that allows processing of repeating message elements in a loop.

11. Manually map the node `quantity` to `qty`.
12. Verify and save your work.



13. Close all editors, except the NativeData overview window.

Practice 4-5: Deploying and Running the Project

Overview

In this section of the practice, you deploy and test the project.

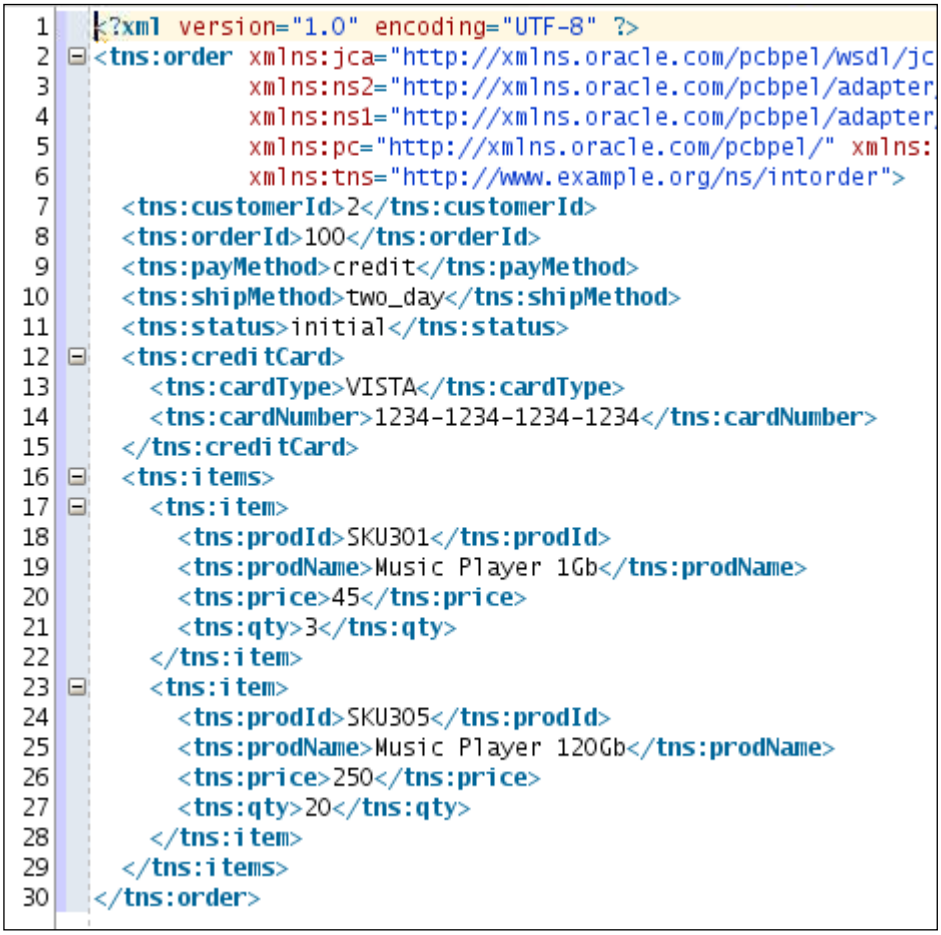
Assumptions

This practice assumes that you have completed Practice 4-4 successfully.

Tasks

1. Deploy the NativeData project to the `IntegratedWebLogicServer` Application Server.
   a. Remember to select the check box to overwrite any existing composites with the same revision ID.
   b. Verify successful deployment in the Message pane.
2. Copy the file `/home/oracle/labs/files/xsd/purchaseOrder.dat` to `/home/oracle/labs/input/po.dat`.
3. Confirm that within 10 seconds, the following occur:
   - The file is removed from the input directory.
   - The original file is moved to `/home/oracle/labs/archive/po.dat_<`*`UUID`*`>`.
   - A copy of the file is passed to the composite application, and then, after transformation, it is written to `/home/oracle/labs/output/podata/po_<`*`dateTimeStamp`*`>.xml` in the application.

4. Use JDeveloper or a text editor to open the output file and verify that the data is correctly formatted.



```
<?xml version="1.0" encoding="UTF-8" ?>
<tns:order xmlns:jca="http://xmlns.oracle.com/pcbpel/wsdl/jc
           xmlns:ns2="http://xmlns.oracle.com/pcbpel/adapter
           xmlns:ns1="http://xmlns.oracle.com/pcbpel/adapter
           xmlns:pc="http://xmlns.oracle.com/pcbpel/" xmlns:
           xmlns:tns="http://www.example.org/ns/intorder">
  <tns:customerId>2</tns:customerId>
  <tns:orderId>100</tns:orderId>
  <tns:payMethod>credit</tns:payMethod>
  <tns:shipMethod>two_day</tns:shipMethod>
  <tns:status>initial</tns:status>
  <tns:creditCard>
    <tns:cardType>VISTA</tns:cardType>
    <tns:cardNumber>1234-1234-1234-1234</tns:cardNumber>
  </tns:creditCard>
  <tns:items>
    <tns:item>
      <tns:prodId>SKU301</tns:prodId>
      <tns:prodName>Music Player 1Gb</tns:prodName>
      <tns:price>45</tns:price>
      <tns:qty>3</tns:qty>
    </tns:item>
    <tns:item>
      <tns:prodId>SKU305</tns:prodId>
      <tns:prodName>Music Player 120Gb</tns:prodName>
      <tns:price>250</tns:price>
      <tns:qty>20</tns:qty>
    </tns:item>
  </tns:items>
</tns:order>
```