3 Creating the Purchase Order Routing Service

3.1 Introduction

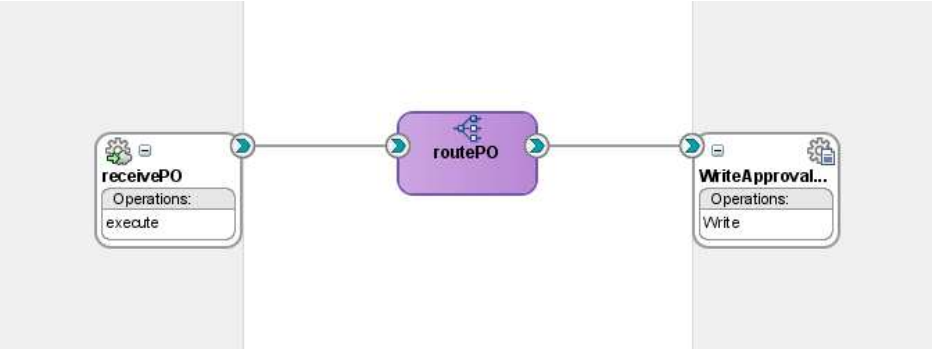
Note: The solution for this chapter can be found in `c:\po\solutions\ch3`. This solution does not require any setup.

You will now create another application that will accept new purchase orders and forward them to the ordering system's archive. In this case you will simulate the archive with a file adapter that will capture to disk all processed purchase orders.

Ultimately, you will continue adding features to this composite application and expand it to implement the full PO Processing application following the chapters of this tutorial. For now, you begin with the first part, the archive.

The implementation of this service uses a Mediator that receives orders coming over SOAP and writes them to file using the file adapter. Once completed, your Mediator flow will look like Figure 1 PO Processing service.

Figure 1 PO Processing service

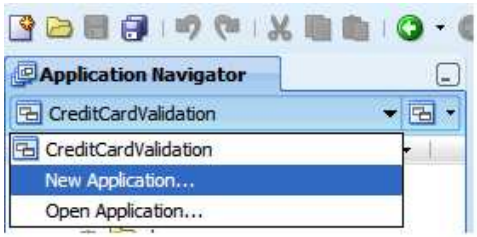


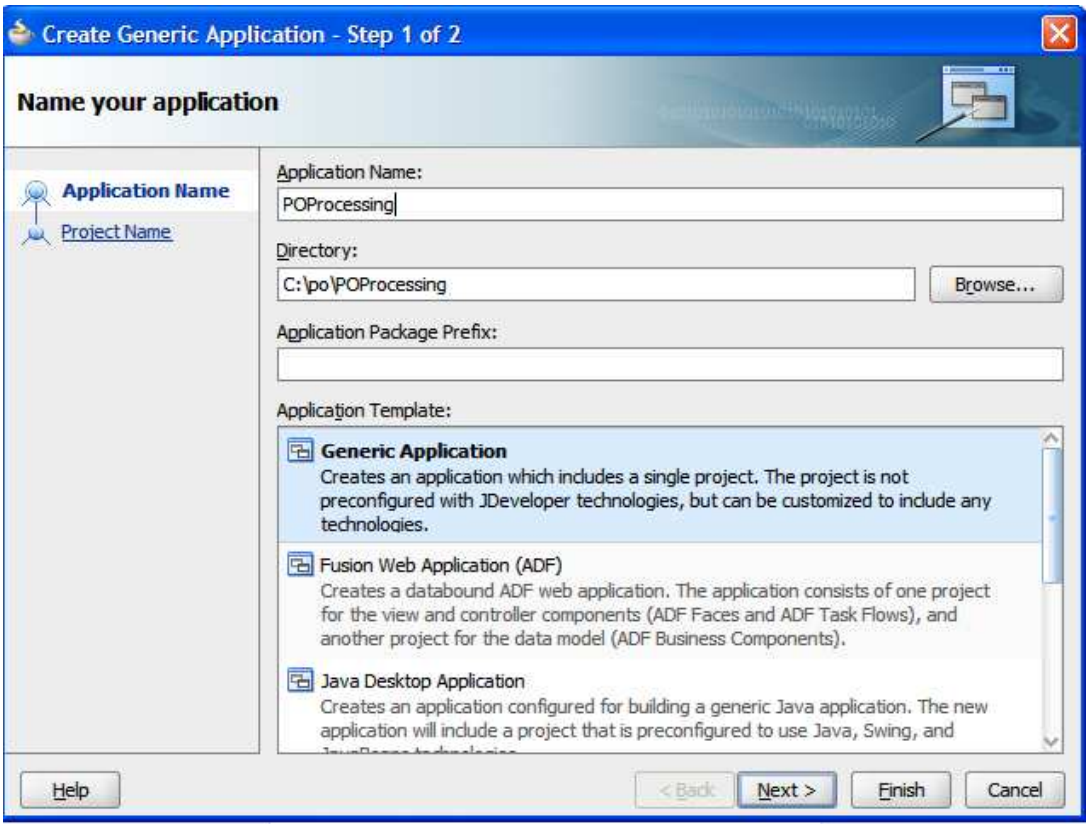
3.2 Designing the flow

This service is a one-way service, also called fire-and-forget. It does not return any value. It takes a PurchaseOrder defined in `po.xsd` and writes data to a file using type Order defined in `internalorder.xsd`. Data transformation from type PurchaseOrder to Order is aided by a pre-defined dictionary named `po2internalorders-dictionary.xml` (available in your schemas directory).

3.3 Creating a new application

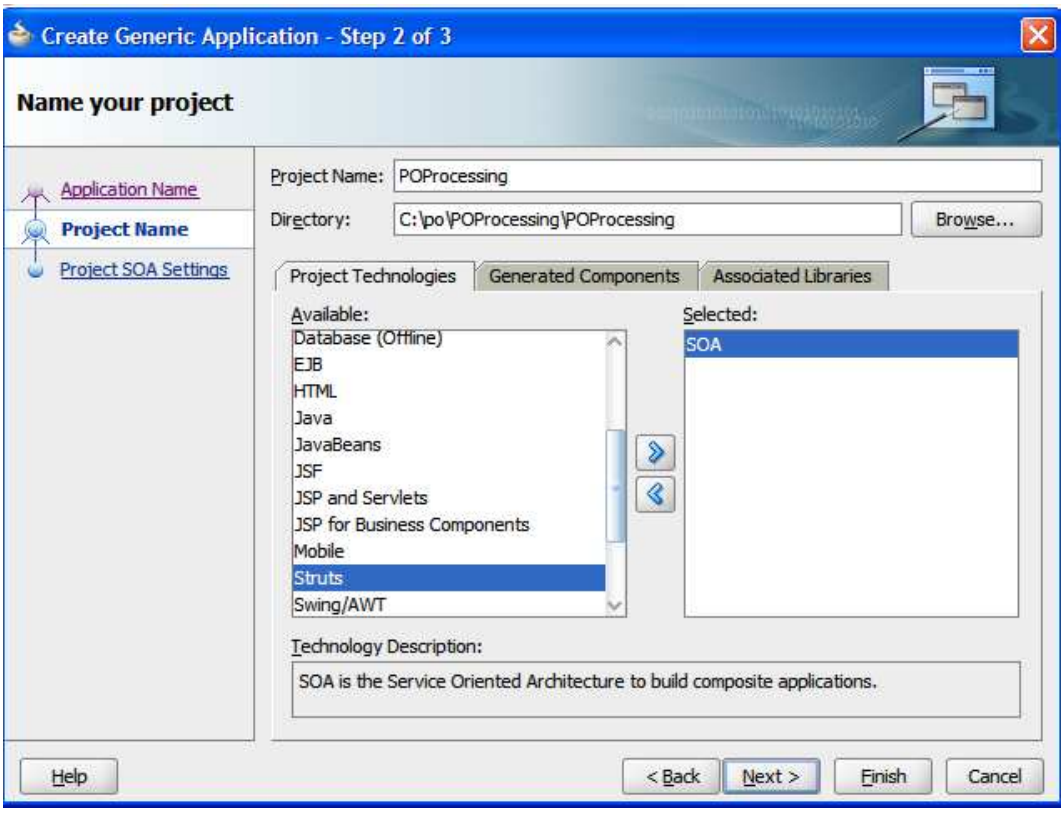
1. Create a new application for your POProcessing composite. As an alternative, instead of using the **File/New** menu, choose **New Application** from the application list menu.





2. Name your application **POProcessing** and make its location in `c:\po`.
3. Click **Next**.

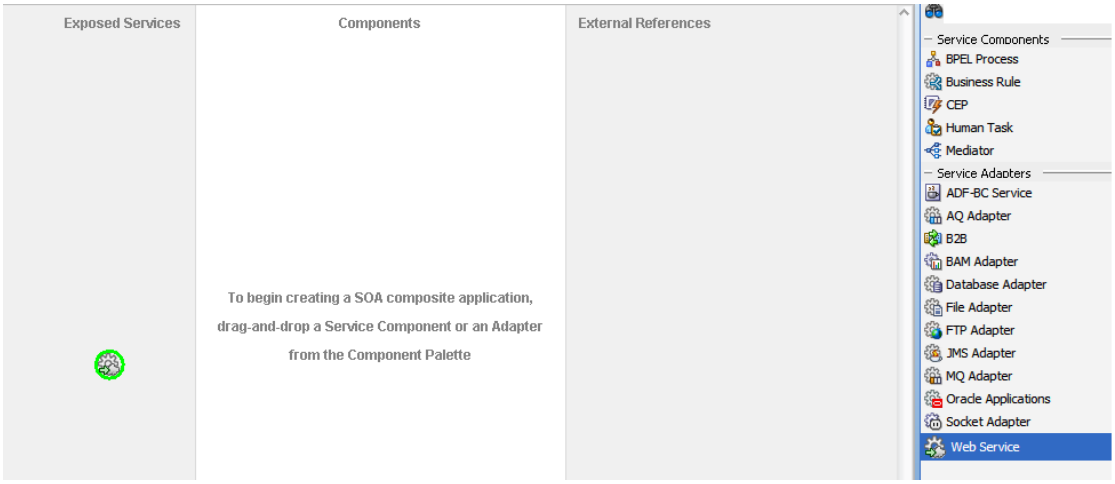
4. Name the project **POProcessing** and select **SOA** for the **Project Technologies**.



5. Click **Next** and then click **Finish** to create the project with an empty composite.

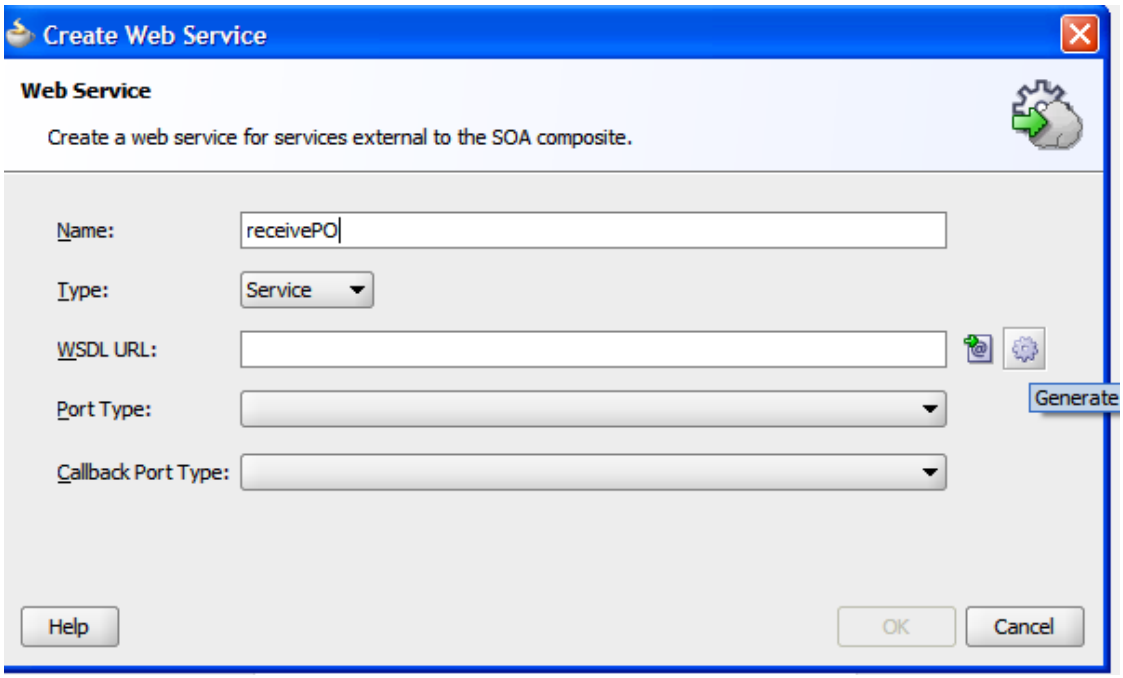
3.4 Adding the service interface

1. Start by creating the service you will use to expose this composite over SOAP. Drag and drop a **Web Service** activity onto the **Exposed Services** swim lane

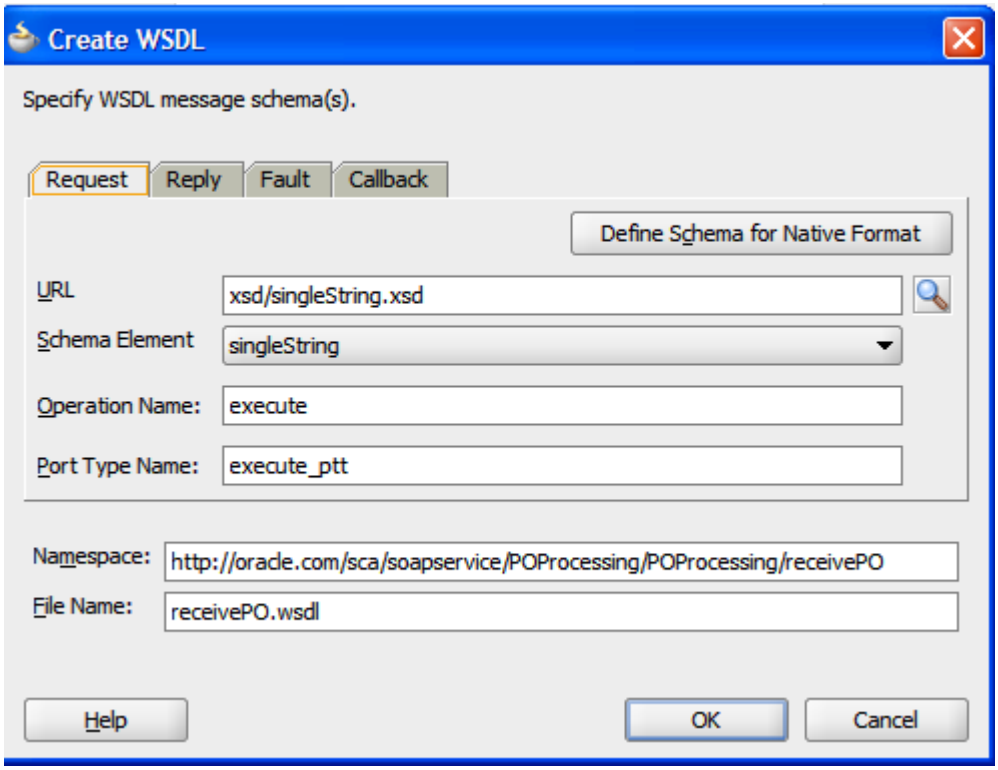


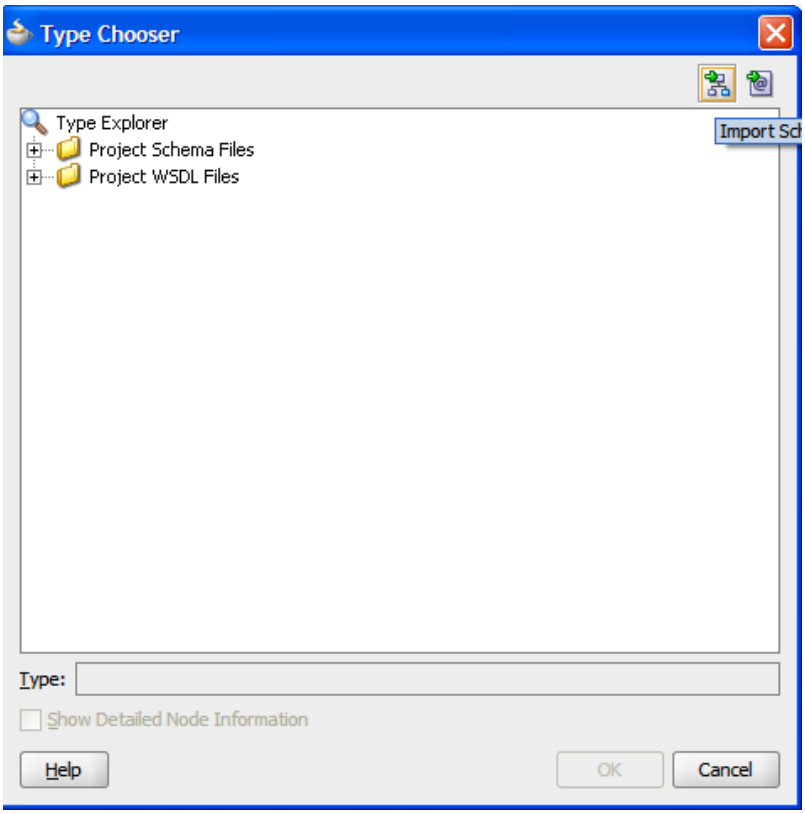
2. Name it **receivePO** and create a WSDL from a schema by clicking the cog icon.

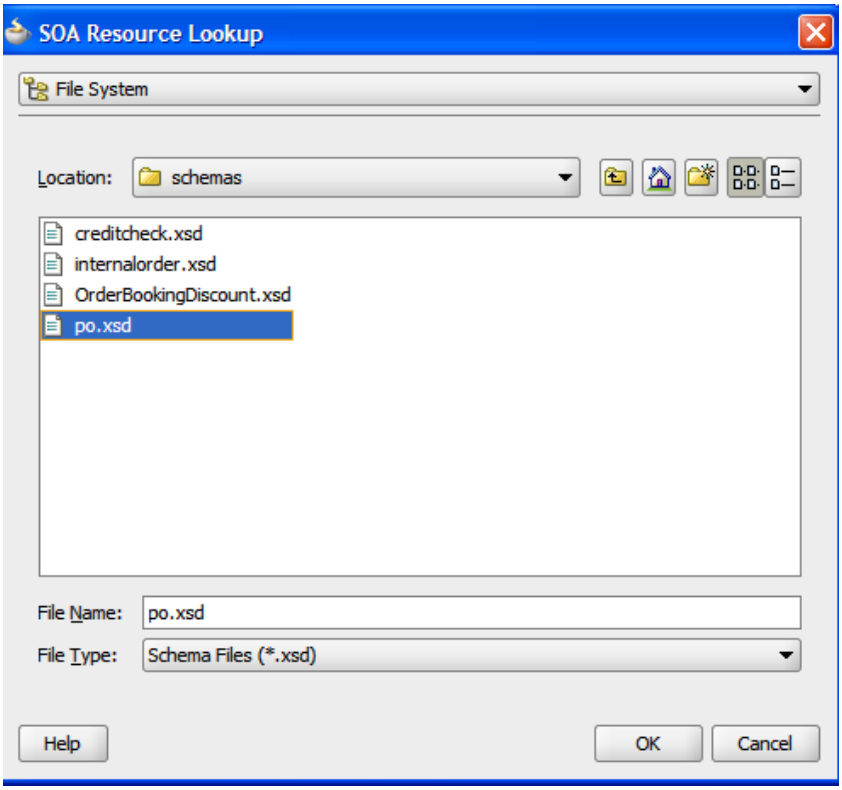
   Creating a WSDL from a schema allows you to define the input data that your service expects and at the same time, automatically create the WSDL interface.



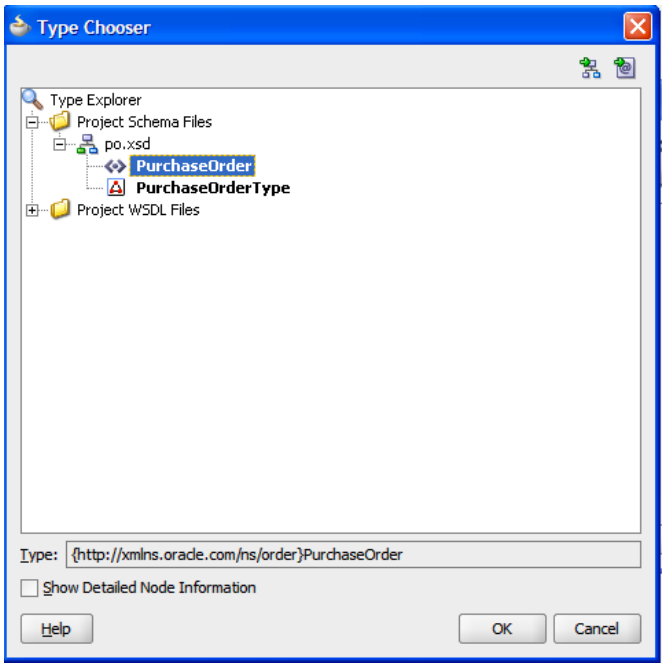
3. Browse for the `po.xsd` schema (which you'll find under `c:\po\schemas`), copy it to your project and open it to select the type.



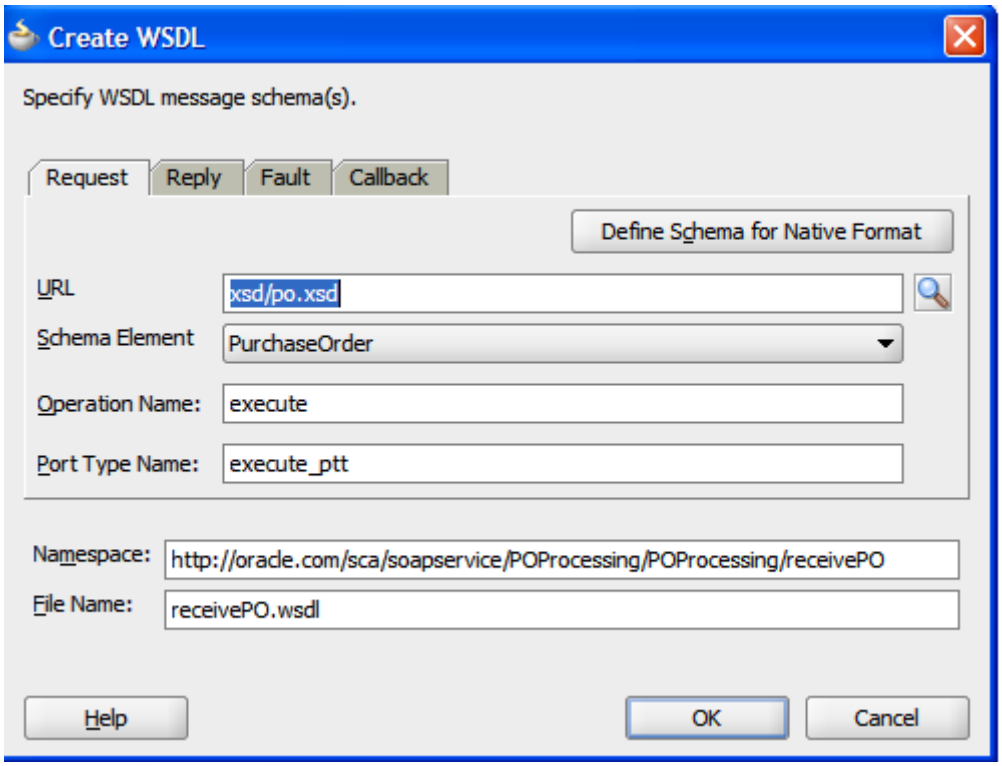

Type Chooser
Type Explorer
Project Schema Files
Project WSDL Files
Import Sc
Type:
Show Detailed Node Information
Help
OK
Cancel


SOA Resource Lookup
File System
Location:
schemas
creditcheck.xsd
internalorder.xsd
OrderBookingDiscount.xsd
po.xsd
File Name:
po.xsd
File Type:
Schema Files (*.xsd)
Help
OK
Cancel

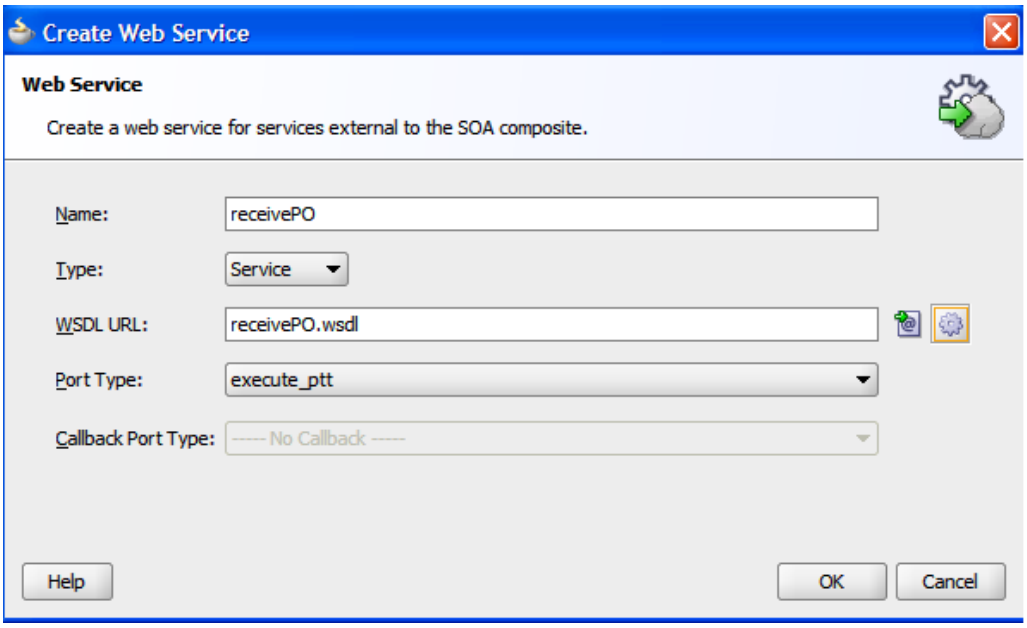
4. Select the **PurchaseOrder** element. This is the input data for your service.



This service is a one-way invocation type, also known as a fire-and-forget service. So there is no need to specify a reply or callback.



5. Click **OK**.

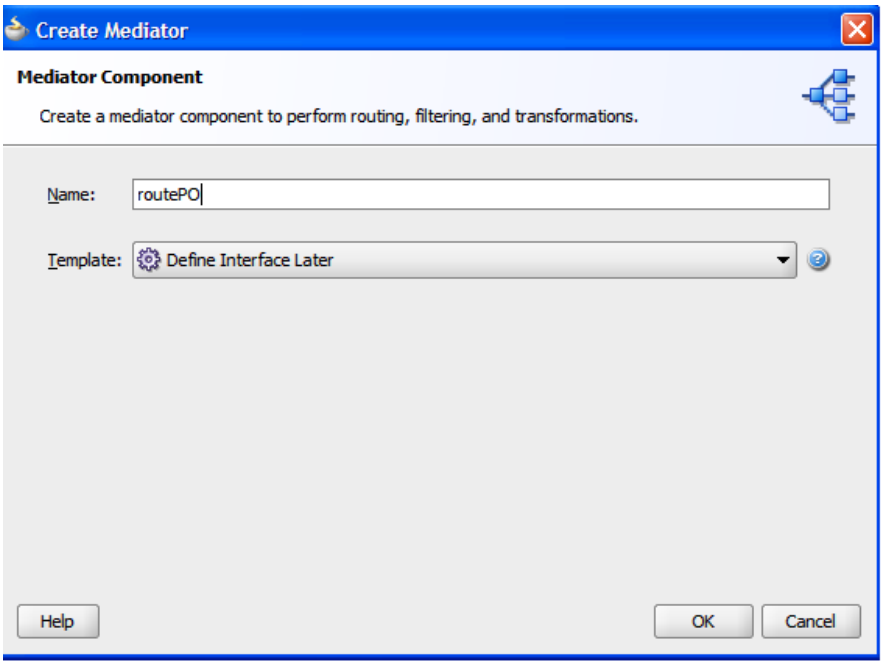


6. Click **OK** to close the web service binding dialog.

3.5 Adding the routing component

1. Now add a Mediator to the composite by dragging one from the palette to the canvas. Call it **routePO** and select the **Define Interface Later** template.

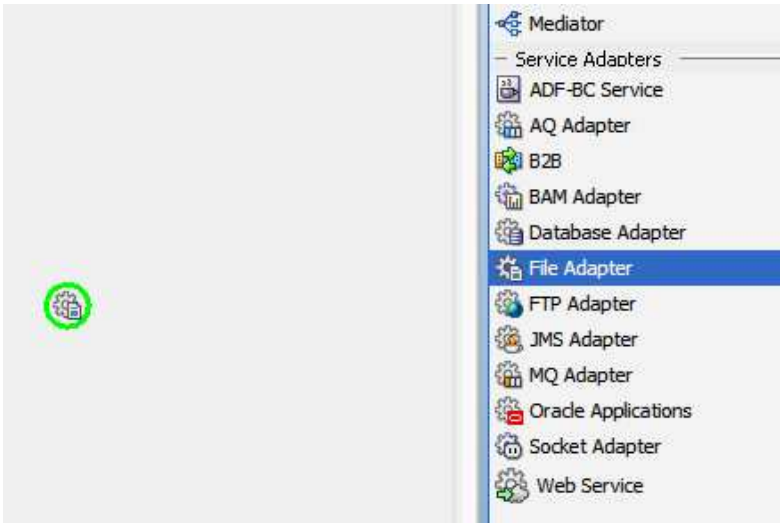
   When you define the interface later, you can graphically connect the SOAP service front end and file adapter service reference. The interfaces are implicit.



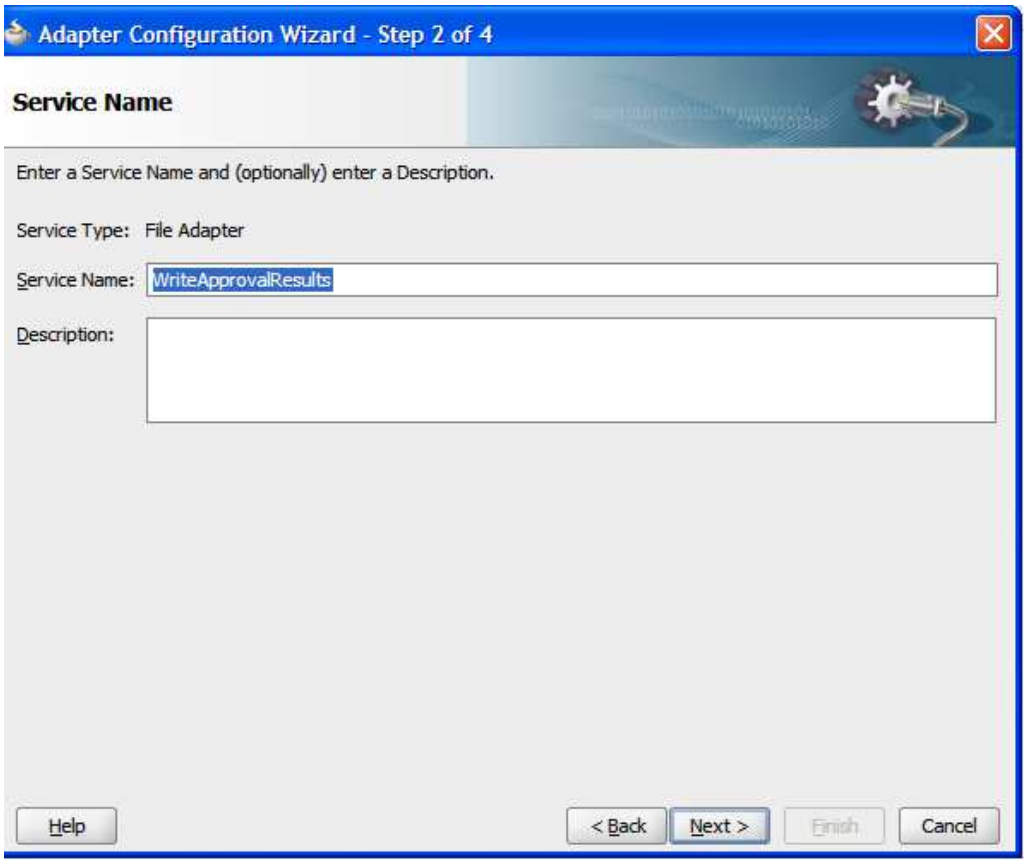
2. Click **OK**.

3.6 Adding the File Adapter

1. Drag-and-drop a **File Adapter** to the **External References** swim lane. This file adapter will write each new order to a text file.



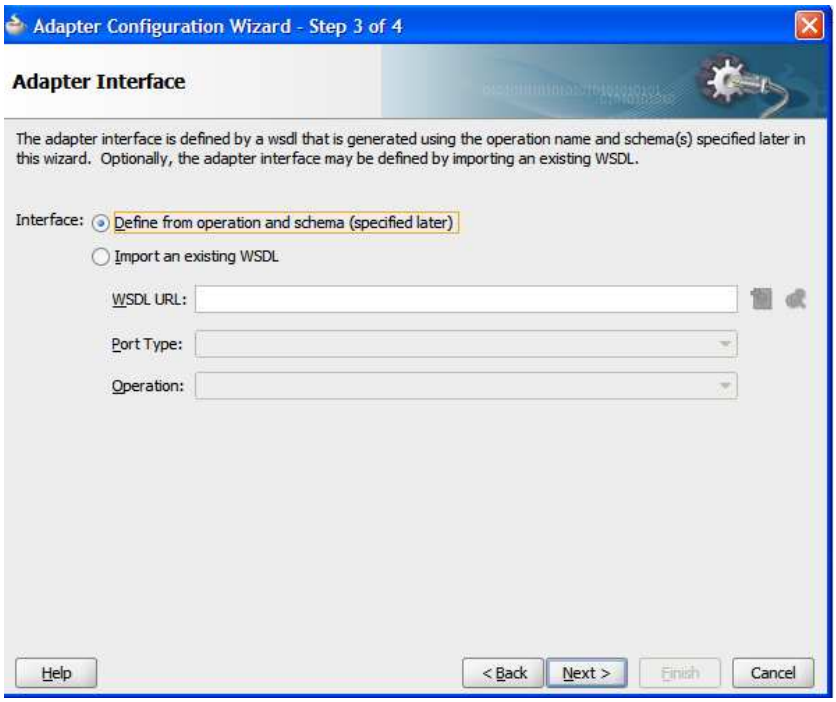
2. Name the service **WriteApprovalResults**.



3. Click **Next**.

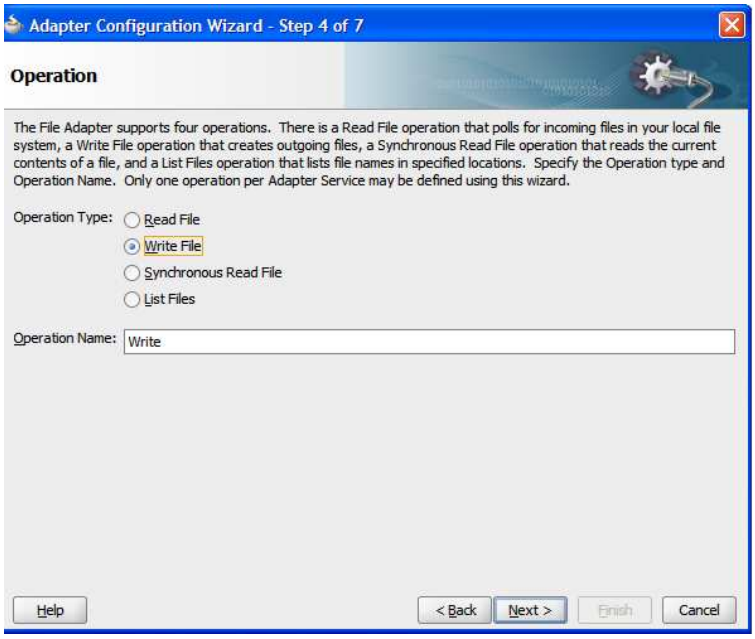
4. Select the option to **Define from operation and schema**

   This feature allows you to use the file operation and schema to define the WSDL automatically. Alternatively, you could use an existing WSDL file if you had one already created for the service reference.



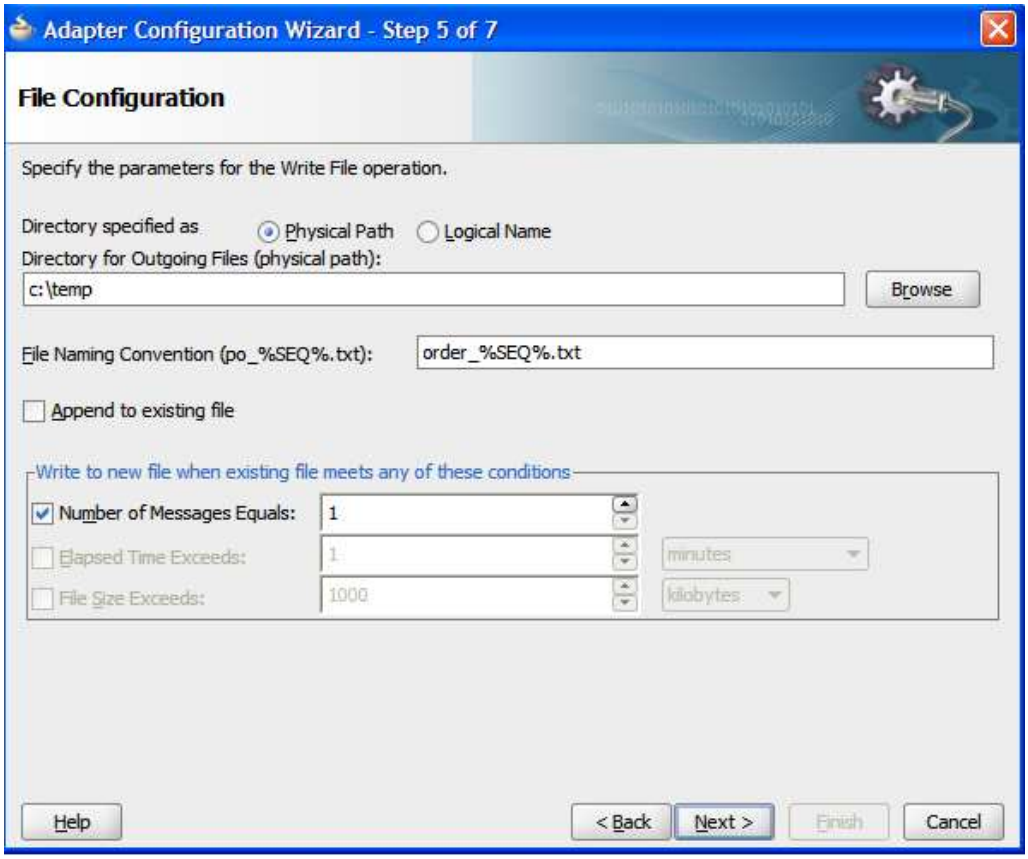
5. Click **Next**

6. Select the **Write File** operation.



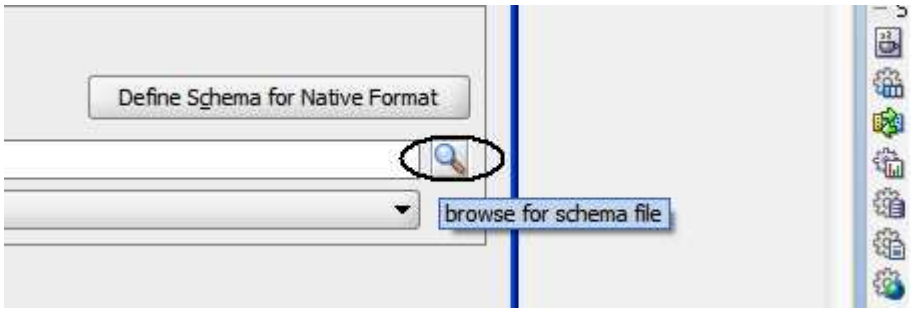
7. Click **Next**.

8. Specify the following settings, leaving all others with their default values:
   - **Directory for Outgoing Files:** *c:\temp* (or Linux notation if using Linux)
   - **File Naming Convention:** *order_%SEQ%.txt* to write the files with increasing sequence numbering. You can see additional options for numbering files in a drop down menu as soon as you enter % in the field.

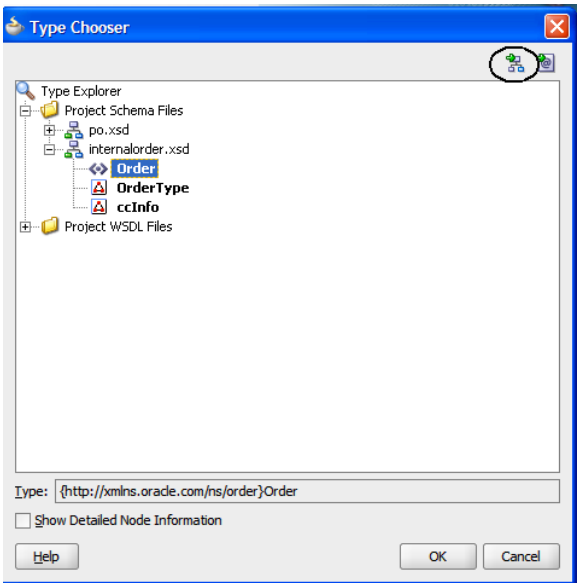


9. Click **Next**.

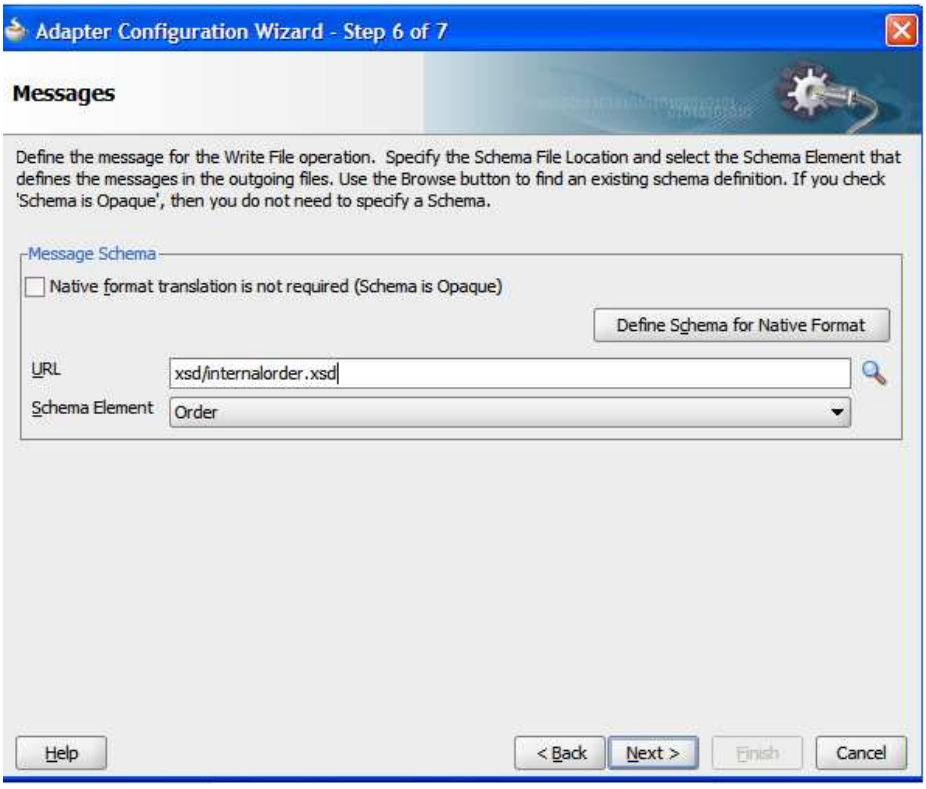
10. Browse for the schema that represents the content we will write.



11. Click the **Import Schema** File button and navigate to `c:\po\schemas\internalorder.xsd` to select and copy the schema file, and then select **Order** from the **Type Chooser** dialog.



12. Click **OK**

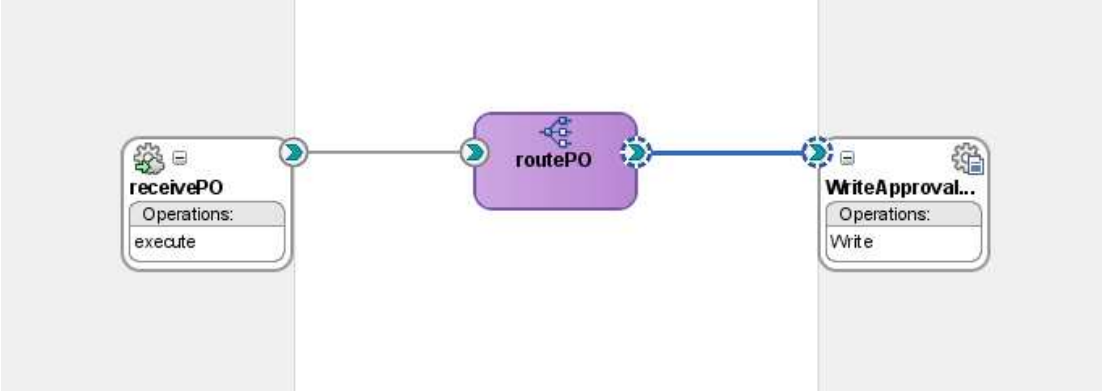


13. Click **Next**, then **Finish** to complete the File Adapter wizard and return to the composite.

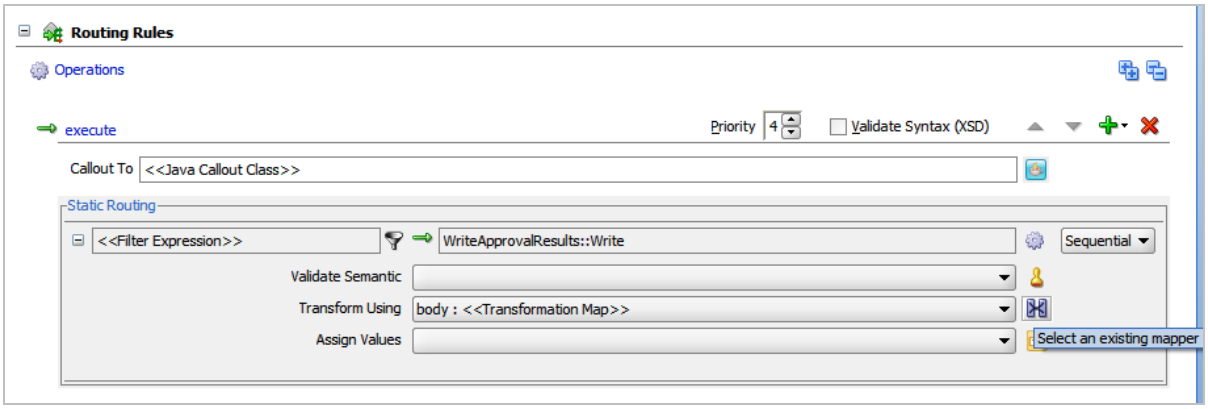
3.7 Wiring the components and adding a transformation

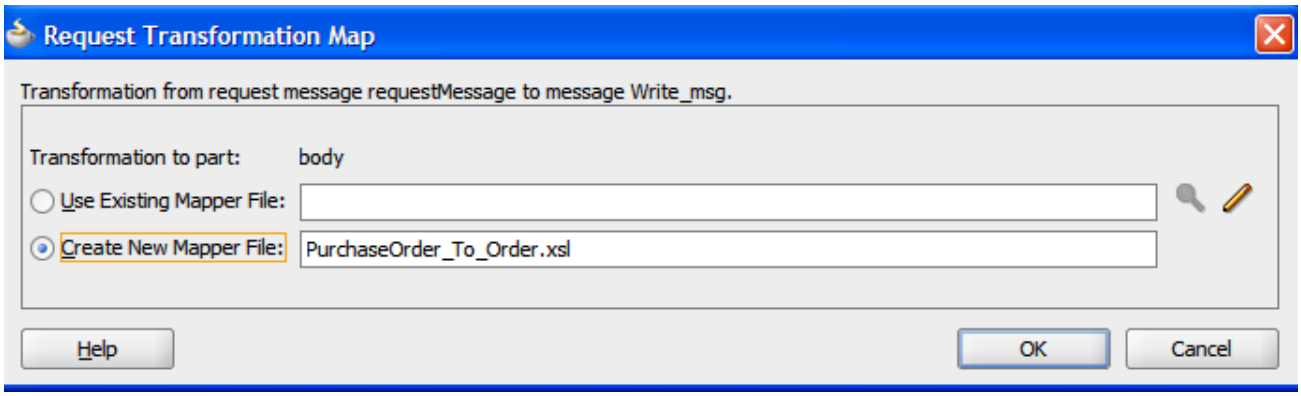
1. Next, wire the components to connect the service and reference to the mediator and complete the interface definitions.

   The wired components look like this.

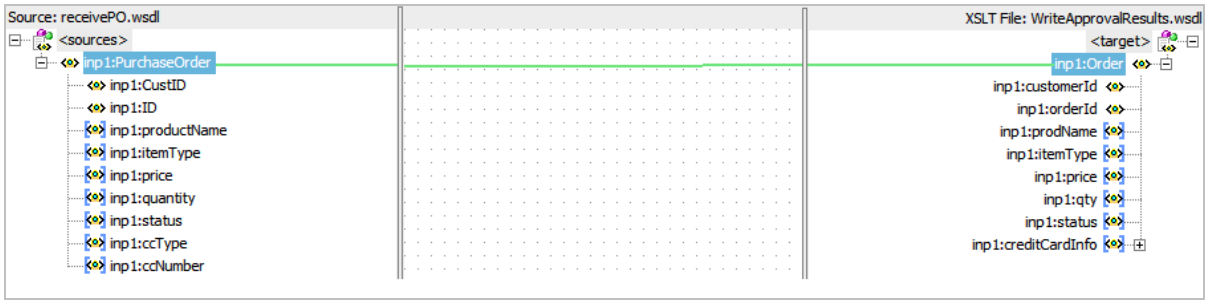


2. Double-click the Mediator component to open the Mediator editor and create the mapping between the inbound PO and the internal order format that you use to log to file.

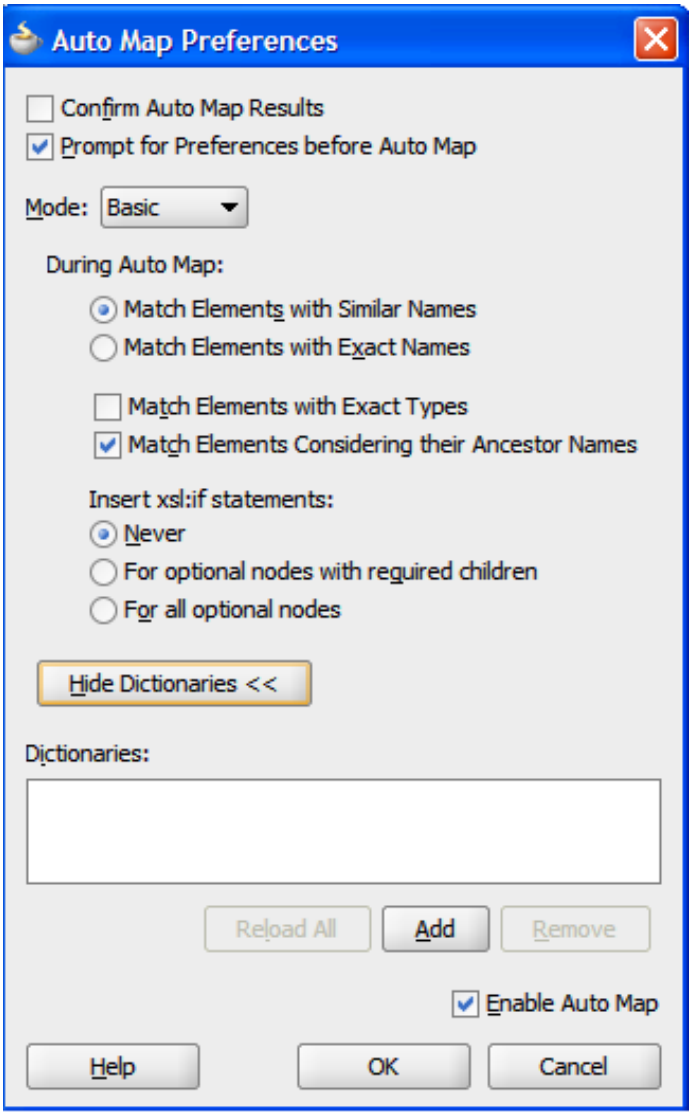




3. Drag a wire from **Purchase Order** on the source side to **Order** on the target side.



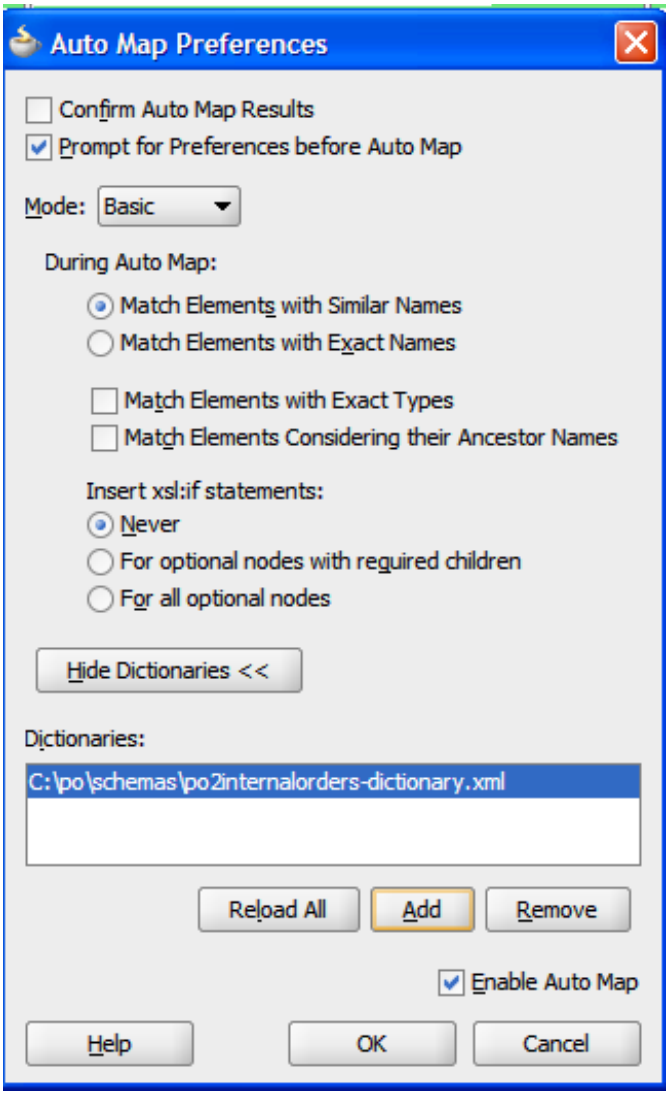
4. You are prompted for auto-mapping preferences:



To help in the mapping, you are going to leverage a dictionary created by the business analysts and that lists common synonyms in use across your data objects

(e.g., “qty” is sometimes used instead of “quantity”, some departments use “ID” instead of “orderId”, etc.).

5. Uncheck **Match Elements Considering their Ancestor Names**
6. Click on **Show Dictionaries**.
7. Click **Add**.
8. Browse for `c:\po\schemas\po2internalorders-dictionary.xml`.
9. Click **Open**

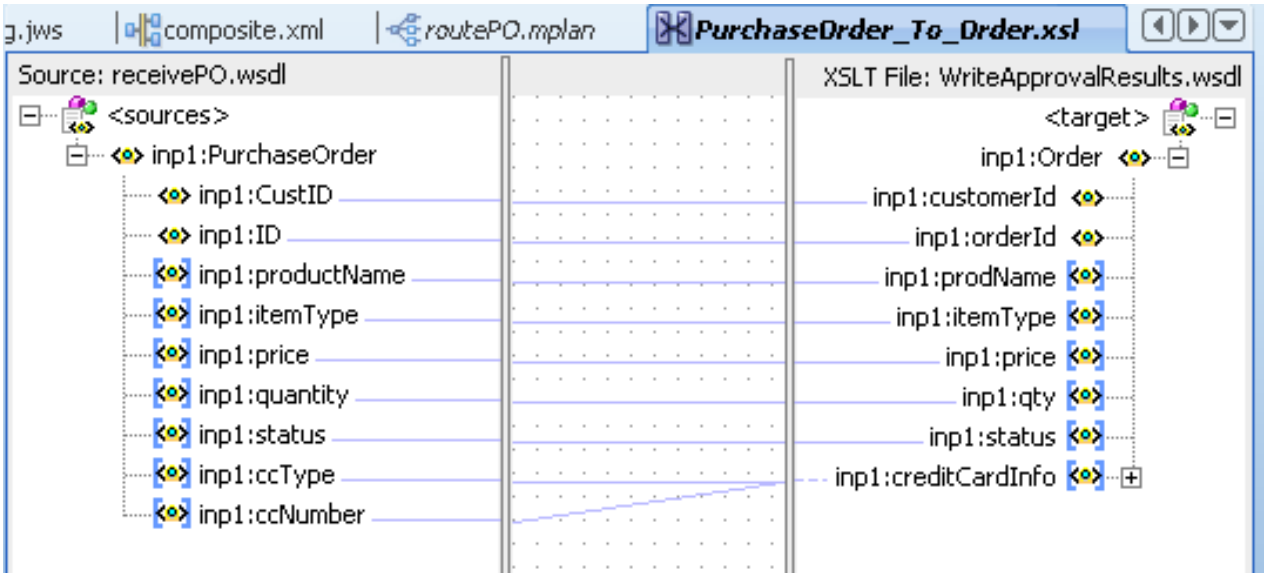


10. Click **Ok**.

    The dictionary has helped match fields such as CustID to customerId and productName to prodName.

    Note that even without a dictionary, the auto-mapping feature will be able to identify and automatically take care of many of these fields, but a dictionary, customized to your company helps improve its accuracy.

The resultant mapping looks like this:



11. Save and close both the mapping and the Mediator editor to return to the composite.
12. At this point in time you have a fully-functional Mediator flow that can be deployed and tested. In later chapters, you will add more components to it.

3.8 Deploying the application

Deploy the application in the same way as before. Read **Appendix A Deploying and Running a Composite Application** to refresh your memory on how to deploy if you need to.

3.9 Testing the application

Test your new application using the instructions here. Read **Appendix A Deploying and Running a Composite Application** for more details.

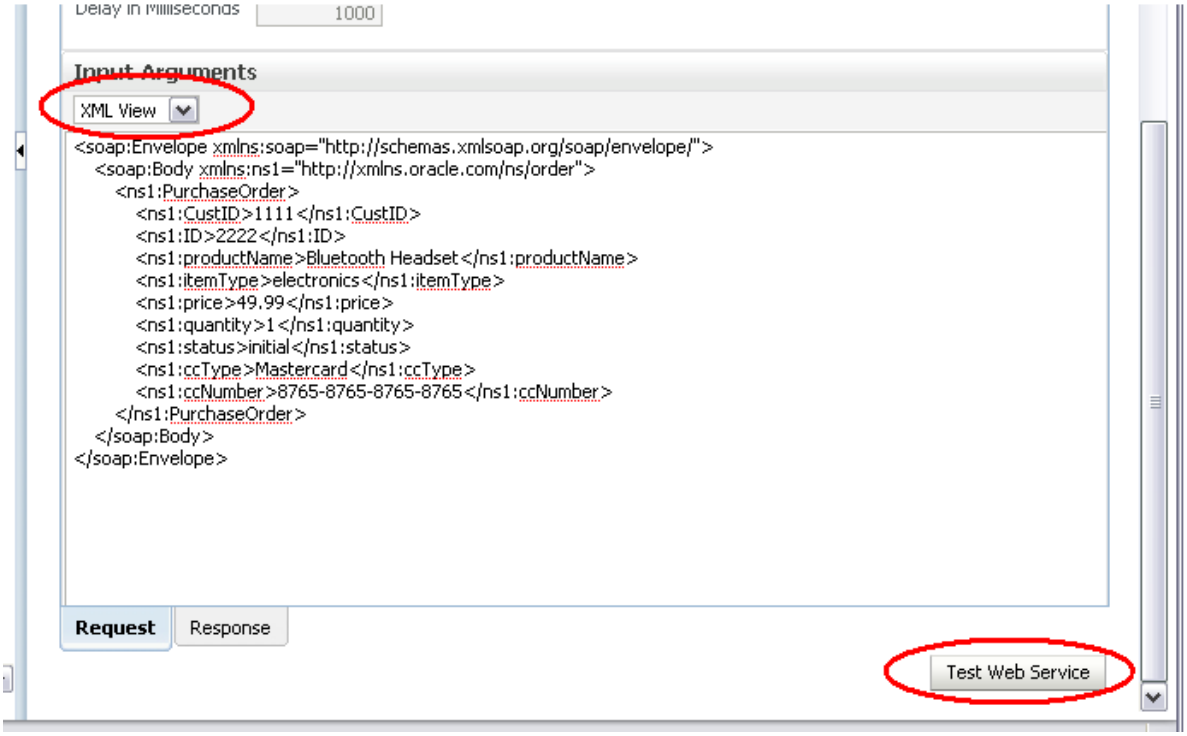
1. After you have deployed your composite, open Enterprise Manager at the following link: http://localhost:7001/em
2. Click on **POProcessing** and the **Test** button to test your service.

3. Enter a small order by doing one of the following:

   - Type the values into in to the HTML form.
   - Click **XML View** so you can paste in the XML payload. This is the recommended way. Open the following file in a text editor:

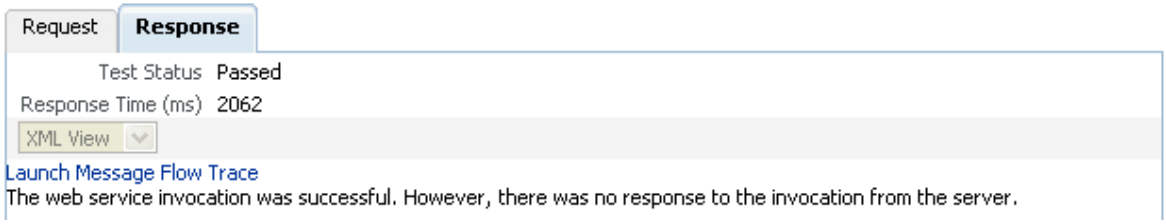
     `c:\po\input\po-small-Headsetx1.xml`

     Copy the entire contents and paste them into the large text field in your browser, replacing the blank xml form that is there:

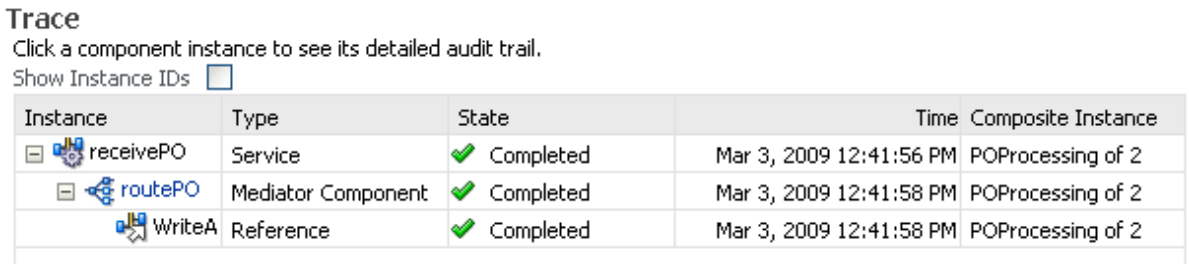


4. Click **Test Web Service**.

The **Test Result** screen won't have any response because this is a one-way invocation with no reply or callback.

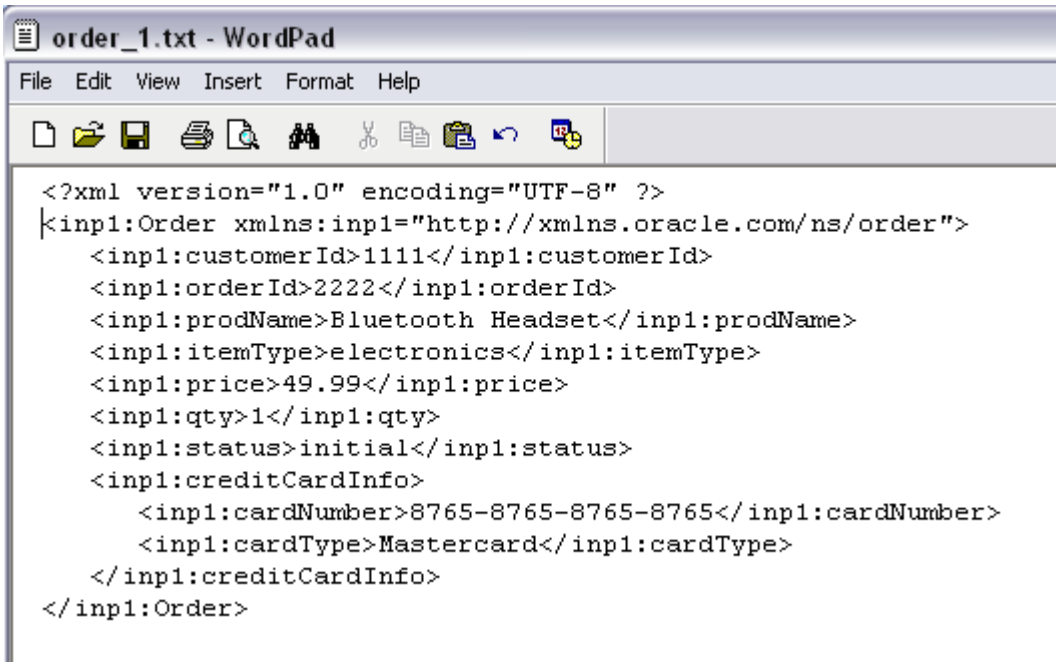


5. You can view the **Flow Trace** by selecting **Launch Message Flow Trace**.



In addition, the call to the File Adapter service will have resulted in a new file in `c:\temp`, called `order_n.txt`, where *n* is a sequence number like 1, 2, 3, etc.

You can open that file with a text editor and examine it. Notice how field names have been translated by the mapping and are different from the input xml.



3.10 Operations and naming

This section gives you all of the operations and names for objects created in this chapter. Experienced users can use this for creating the objects in this chapter quickly. Any questions on details for a particular operation listed here can be found in the preceding sections. The information is divided by the sections in this document.

Section 3.3: Creating a new application

- **Application Name:** POProcessing
- **Directory:** c:\po
- **Project Name:** POProcessing
- **Project Technologies:** SOA
- **Empty Composite**

Section 3.4: Adding the service interface

- **Service:** receivePO
- **Type:** Service
- **WSDL based on:** po.xsd
- **Request:** PurchaseOrder

Section 3.5: Adding the routing component

- **Mediator:** routePO
- **Define Interface Later**

Section 3.6: Adding the File Adapter

- **File Adapter:** WriteApprovalResults
- **Directory for Outgoing Files:** c:\temp (or Linux notation if using Linux)
- **File Naming Convention:** order_%SEQ%.txt
- **Schema file: c**:\po\schemas\internalorder.xsd
- **Schema:** Order

Section 3.7: Wiring the components and adding a transformation

- **Wire:** service to mediator to adapter
- **Mediator transform:** Map Purchase Order to Order
- **In Auto-complete:** Uncheck Match Elements Considering their Ancestor
- **In Auto-complete:** Add Dictionary
- **Dictionary: c**:\po\schemas\po2internalorders-dictionary.xml

The application is completed. Continue with Section 3.8 above to deploy and test your application.