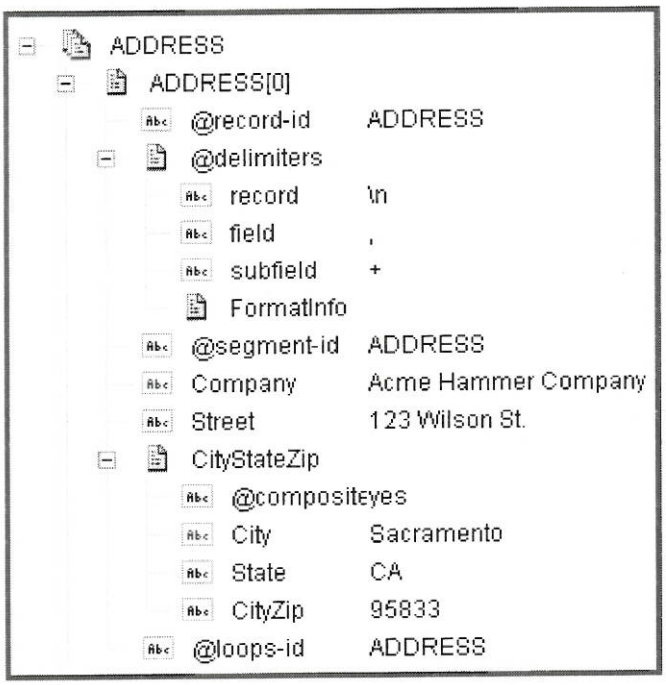
9. Save and test the new **Flat File Schema** called **address** by running it in Developer. When asked for an input file, use **address.txt** in the directory **...\IntegrationServer\packages\AcmeSupport\pub\FlatFile.** Make sure that the first ADDRESS record looks like the following:



Check Your Understanding

1. Why can't flat files be imported like XML documents? These no structure and we do not have schemas that is provided with flat file
2. What is the meaning of Nth field?

This page intentionally left blank

Exercise 14:
Create a Flat File Dictionary

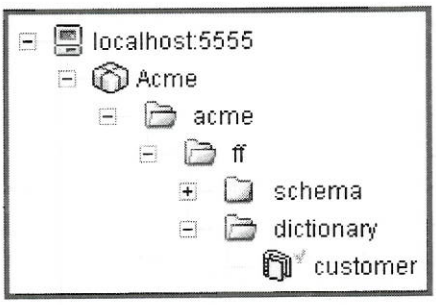
Overview

In this exercise, you will create a new Flat File Dictionary, create a reusable record definition in the Dictionary, reference this record definition in a new Flat File Schema, and test the Flat File Schema.

Just like the last exercise, you have to do this exercise using the developer tool.

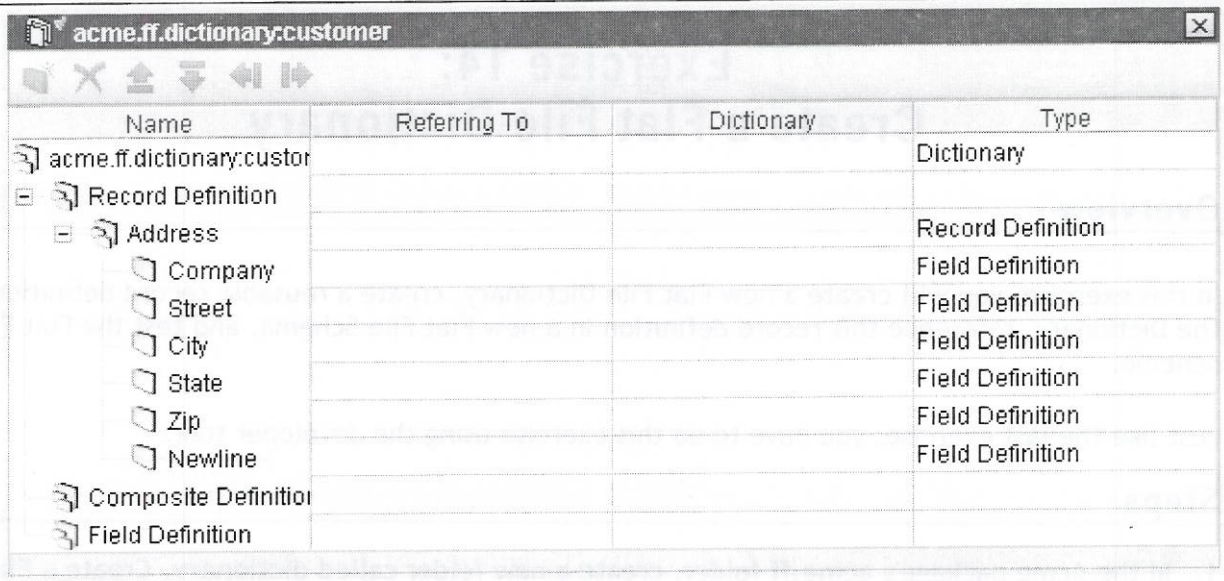
Steps

1. In the Acme package's **acme.ff** folder, create a new folder called **dictionary**. Create a **Flat File Dictionary** called **acme.ff.dictionary:customer.**

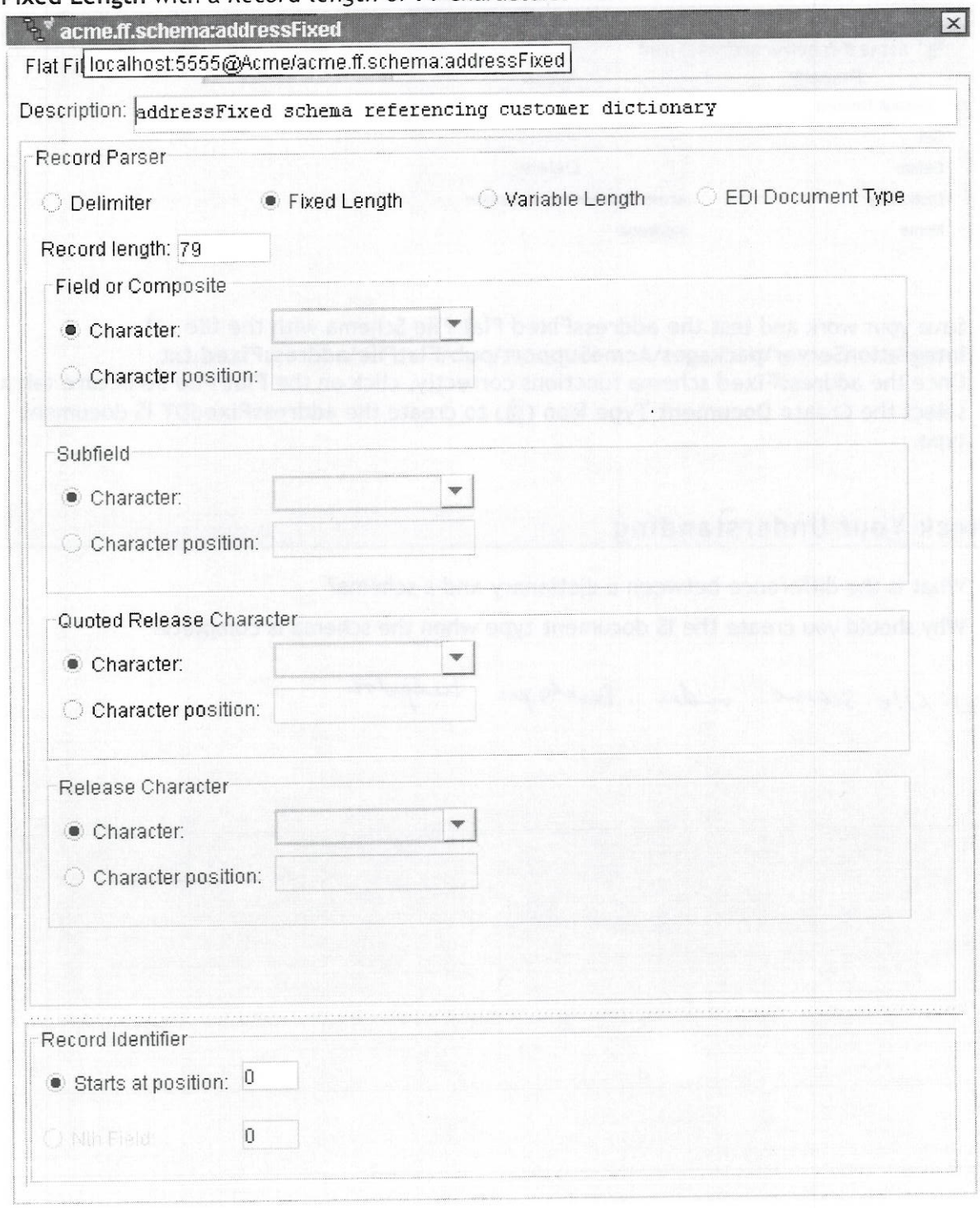


2. Add a record definition called **Address** (case sensitive) to the new **Flat File Dictionary**. Using an **Extractor Type** of **Fixed Position**, add 6 new field definitions to the **Address** record as shown in the table below:

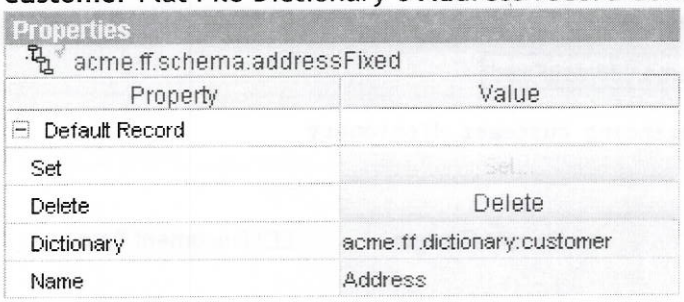
Name	Start	End
Company	0	30
Street	30	55
City	55	70
State	70	72
Zip	72	77
Newline	77	79



3. Create a new **Flat File Schema** called **acme.ff.schema:addressFixed**. Specify that it is **Fixed Length** with a *Record length* of **79** characters.



4. In the **"Default Record"** property of the **addressFixed** schema, add a reference to the **customer** Flat File Dictionary's **Address** record definition.



5. Save your work and test the **addressFixed** Flat File Schema with the file **...\ IntegrationServer\packages\AcmeSupport\pub\FlatFile\addressFixed.txt**
Once the **addressFixed** schema functions correctly, click on the **Flat File Structure** tab and select the **Create Document Type** icon ( ) to create the **addressFixedDT** IS document type.

Check Your Understanding

1. What is the difference between a dictionary and a schema?
2. Why should you create the IS document type when the schema is complete?

Flat File Schema under Developer helpdoc.

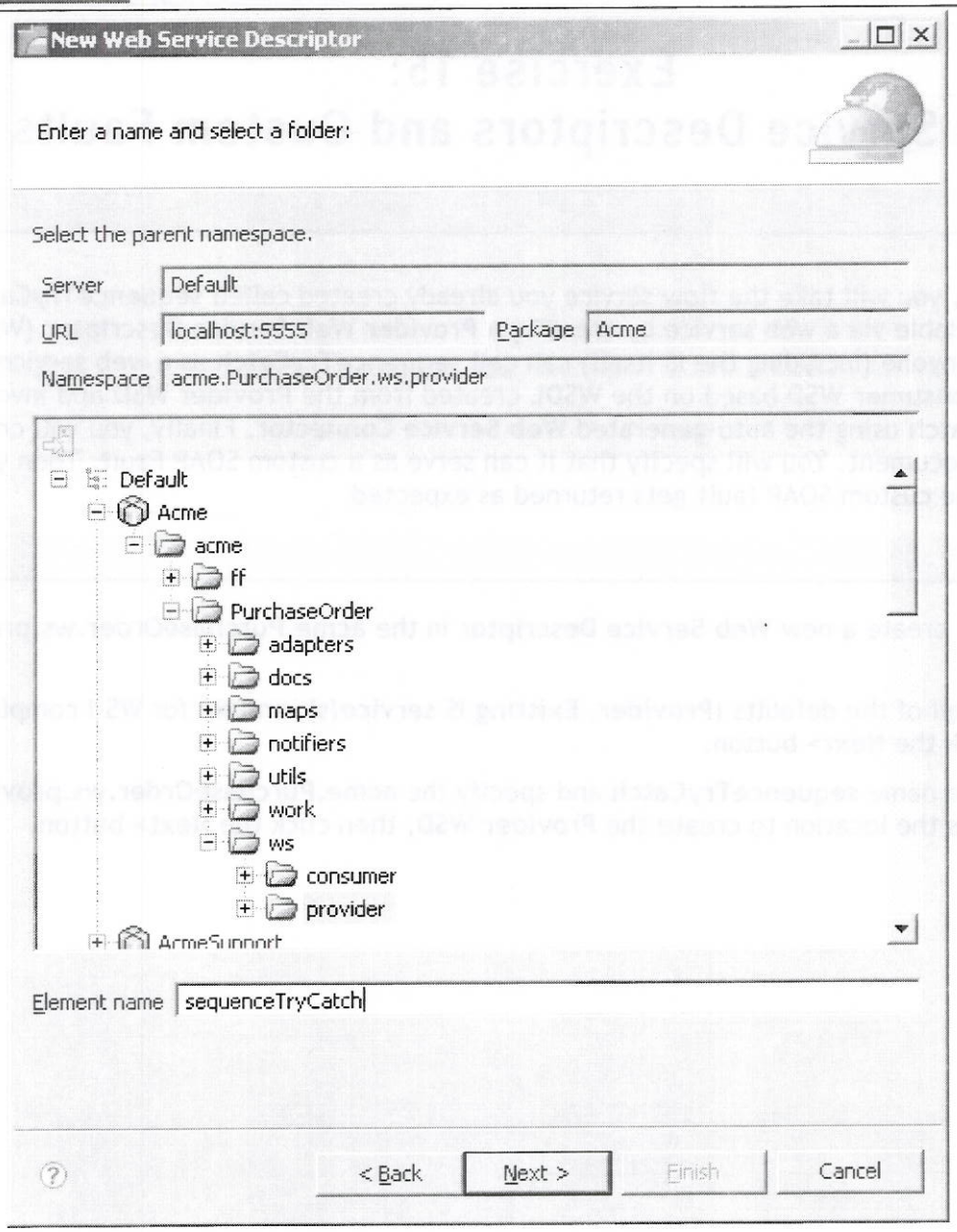
Exercise 15:
Web Service Descriptors and Custom Faults

Overview

In this exercise, you will take the flow service you already created called **sequenceTryCatch** and make it callable via a web service by creating a **Provider Web Service Descriptor (WSD)**. To prove that anyone (including the IS itself) can call **sequenceTryCatch** as a web service, you will create a **Consumer WSD** based on the **WSDL** created from the **Provider WSD** and invoke **sequenceTryCatch** using the auto-generated **Web Service Connector**. Finally, you will create a generic **Error** document. You will specify that it can serve as a custom SOAP Fault. Then you test to see if the custom SOAP fault gets returned as expected.

Steps

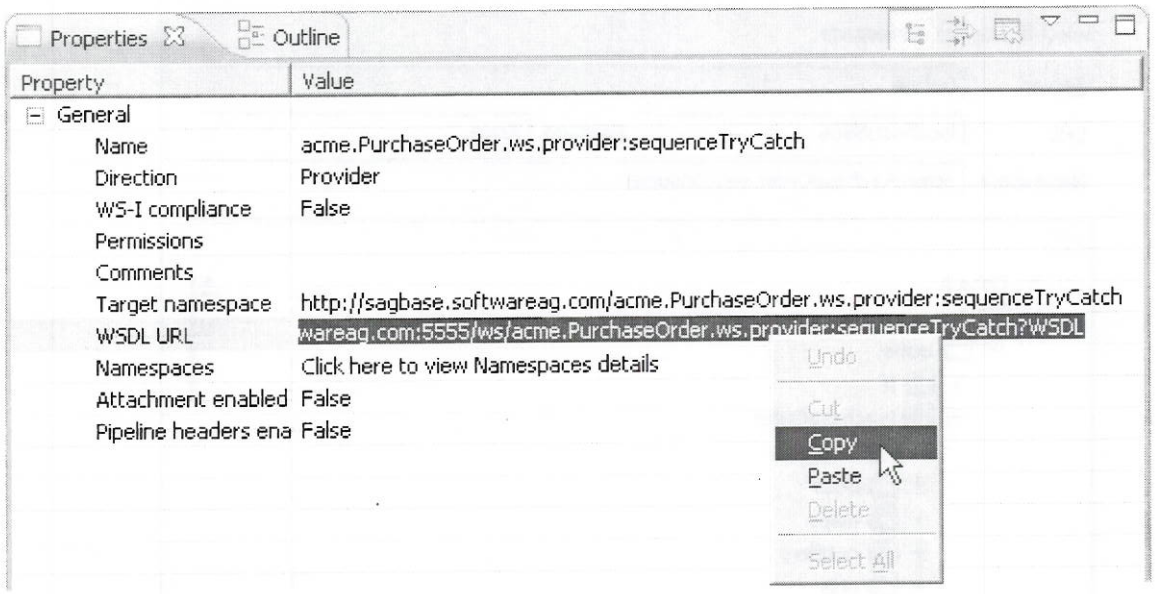
1. In Designer, create a new **Web Service Descriptor** in the **acme.PurchaseOrder.ws.provider** folder.
   a. Accept all of the defaults (**Provider**, **Existing IS service(s)**, and **No** for WS-I compliance) and click the **Next>** button.
   b. Type the name **sequenceTryCatch** and specify the **acme.PurchaseOrder.ws.provider** folder as the location to create the **Provider WSD**, then click the **Next>** button.



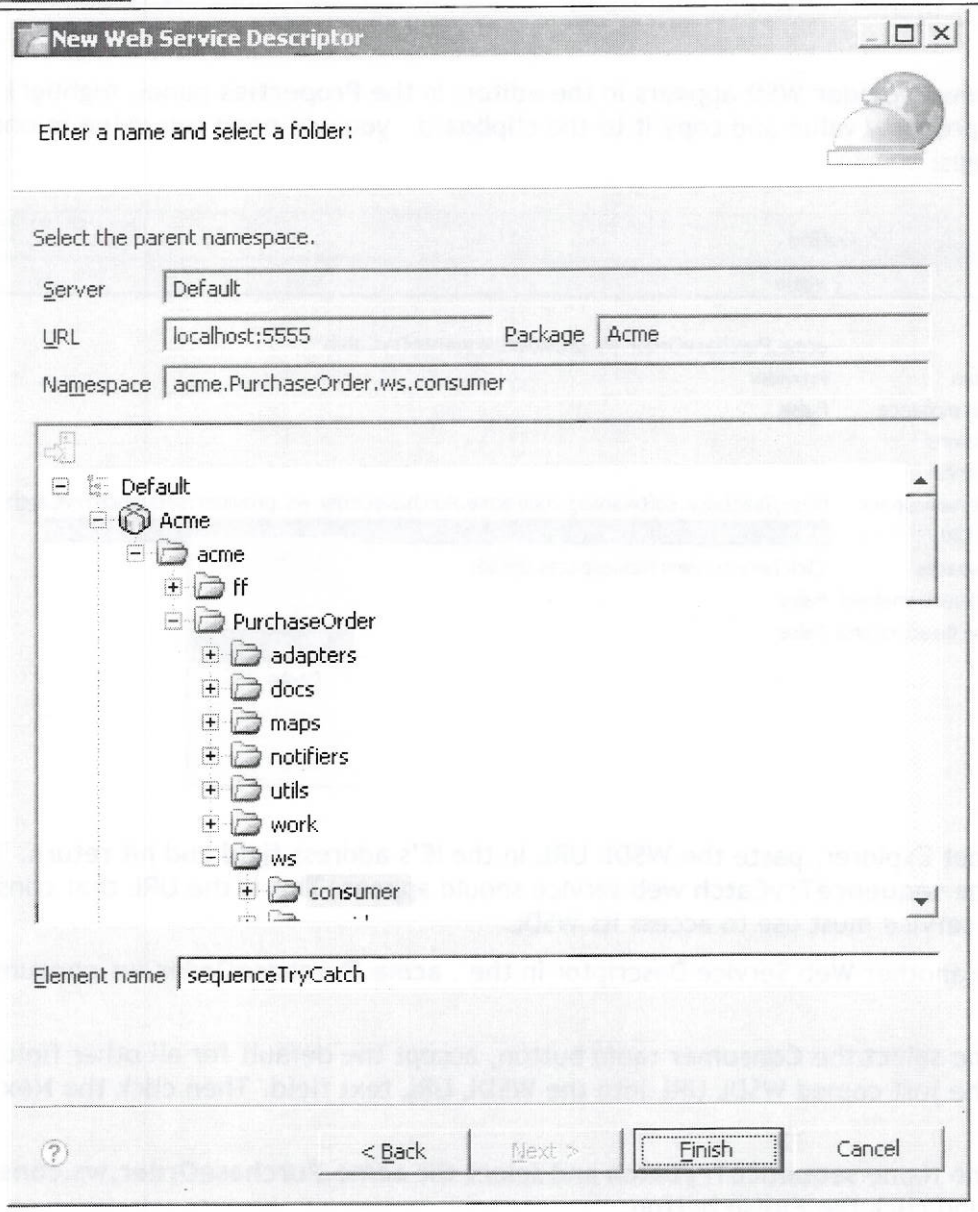
c. In the dialog that appears next, navigate to and select the **acme.PurchaseOrder.work:sequenceTryCatch** flow service, then click the **Next>** button.

d. In the next screen, leave all the defaults in place and click the **Finish** button.

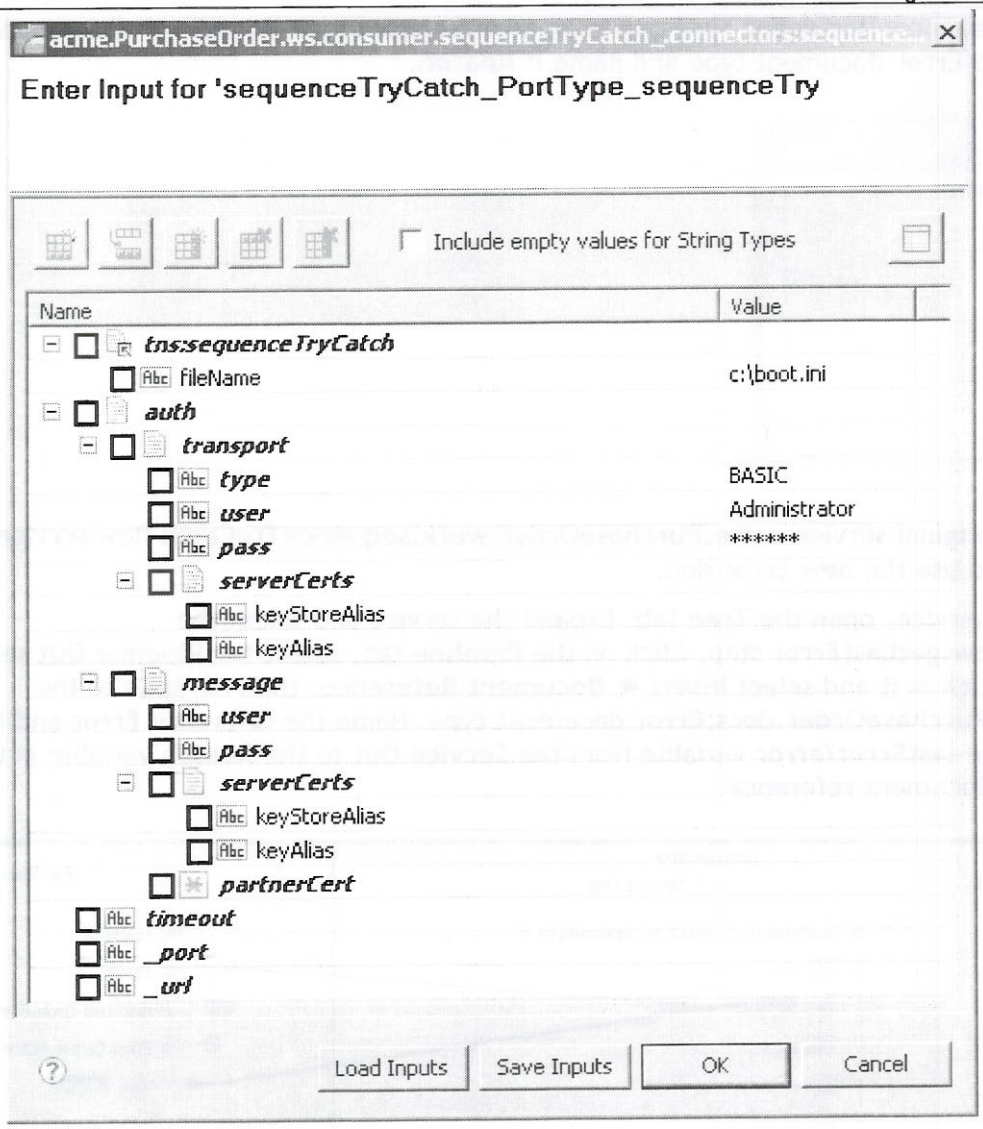
2. When the new **Provider WSD** appears in the editor, in the **Properties** panel, highlight the **WSDL URL** property value and copy it to the clipboard - you will need this value in one of the next steps.



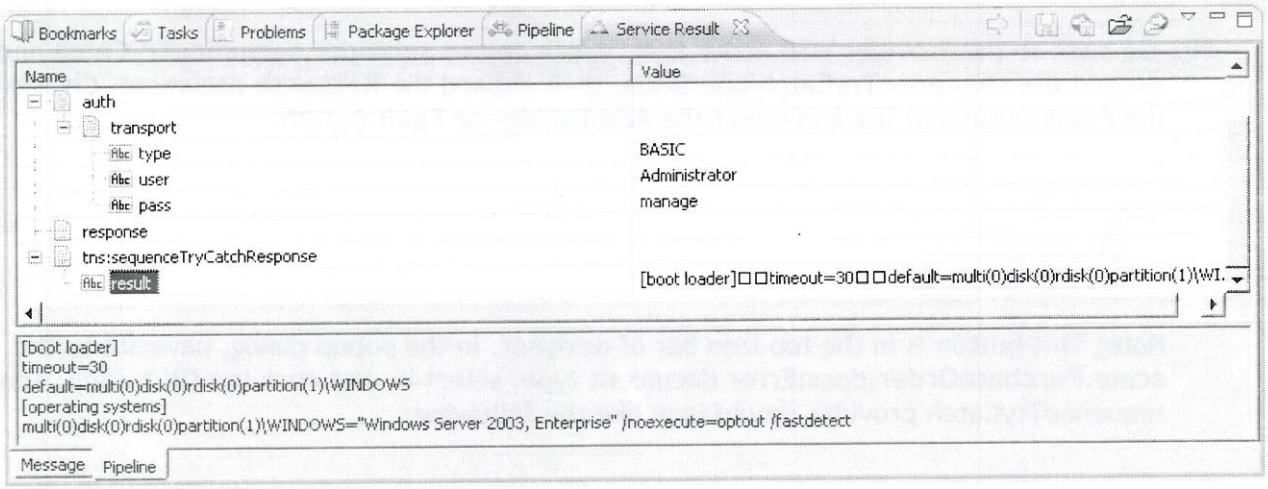
3. Open Internet Explorer, paste the **WSDL URL** in the IE's address field and hit return. The WSDL for the **sequenceTryCatch** web service should appear. This is the URL that consumers of the web service must use to access its WSDL.
4. Now create another Web Service Descriptor in the . **acme.PurchaseOrder.ws.consumer** folder.
   a. This time select the **Consumer** radio button, accept the default for all other fields and paste the just copied WSDL URL into the **WSDL URL** text field. Then click the **Next>** button.
   b. Enter the Name **sequenceTryCatch** and select the **acme.PurchaseOrder.ws.consumer** folder and click the **Finish** button.



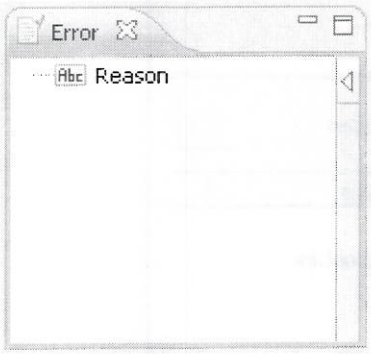
5. After Designer finishes generating the **sequenceTryCatch Consumer WSC**, navigate to and open the **Web Service Connector acme.PurchaseOrder.ws.consumer.sequenceTryCatch_.connectors:sequenceTryCatch_PortType_sequenceTryCatch.**
6. Right-Click to run the **sequenceTryCatch_PortType_SequenceTryCatch** and select "Run As" ➡ "1 Run Flow Service". provide the following values as input:
   a. fileName = c:\boot.ini
   b. auth/transport/type = BASIC
   c. auth/transport/user = Administrator
   d. auth/transport/pass = manage



7. Review the results in Service Result view.

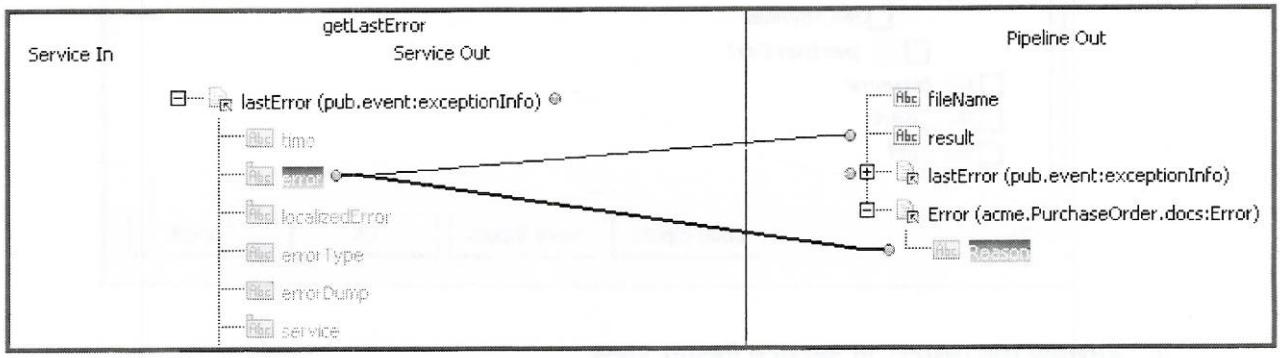


8. Now create a new Document Type called **acme.PurchaseOrder.docs:Error**. Add one string field to the **Error** document type and name it **Reason.**

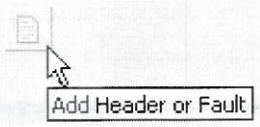


9. Open the original service **acme.PurchaseOrder.work.sequenceTryCatch** flow service and modify it to use the new **Error** doc.

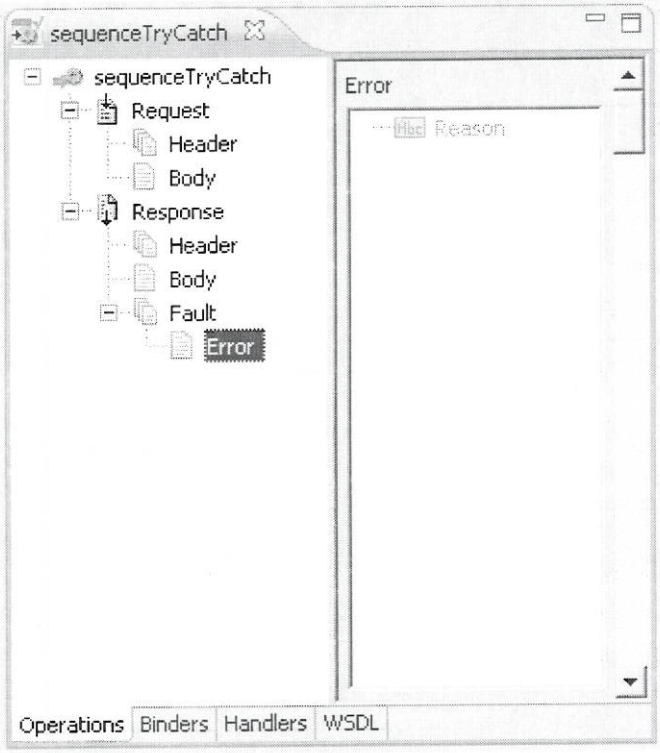
   a. In the service, open the **Tree** tab. Expand the service and select the **pub.flow:getLastError** step. Click on the **Pipeline** tab, select the **Pipeline Out** section, right click in it and select **Insert ➡ Document Reference**, then navigate to the **acme.PurchaseOrder.docs:Error** document type. Name the reference **Error** and then map the **lastError/error** variable from the **Service Out** to the **Reason** variable in the **Error** document reference.



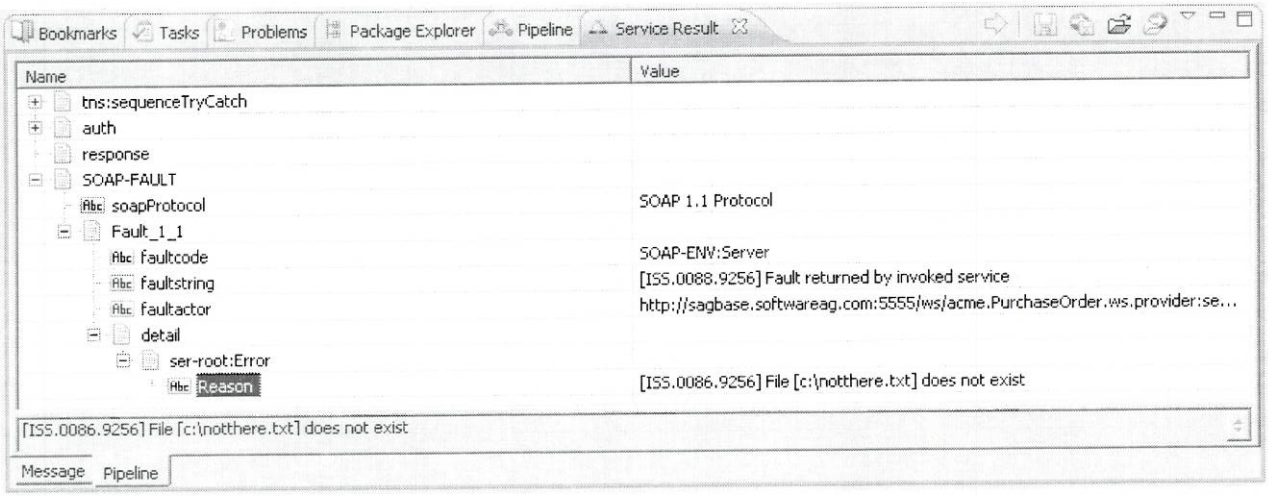
10. Go back to the Provider WSD **acme.PurchaseOrder.ws.provider.sequenceTryCatch** and expand the **sequenceTryCatch** operation, then expand the **Response** document. Click on the **Fault** document list and select the **Add Header or Fault** button.



Note: This button is in the top icon bar of designer. In the popup dialog, navigate to the **acme.PurchaseOrder.docs:Error** document type, select it, and click the **OK** button. Your sequenceTryCatch provider should look like the following:



11. Now you must regenerate the consumer **WSD** connector Flow services so that they capture the change to the Provider WSD. Right-click on the Consumer WSD and select "Refresh Web Service Connectors".
12. Follow steps 5 - 7 above to run the recreated consumer **WSD**, but enter **c:\notthere.txt** for **fileName**. Note: you should see your custom fault document in the Service Result view.



Check Your Understanding

1. When would you create a Provider WSD when a Consumer WSD?
2. How and when are WSC's created? After you have the Provider WSD.
3. Can you have more than one custom SOAP Fault Document? Yes, because the fault object is a document list

This page intentionally left blank

Exercise 16:
Broker Pub/Sub

Overview

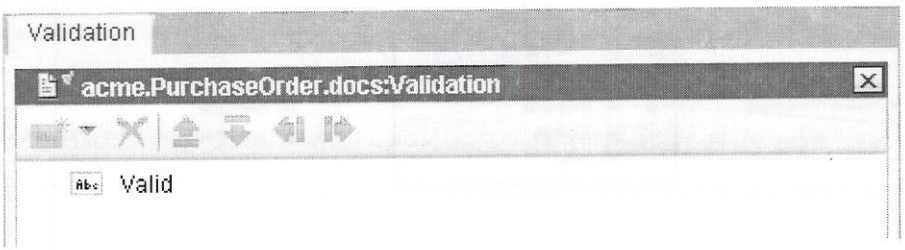
In this exercise, you will create a document type, a handling service and a subscribing Broker trigger. Then you create a service to publish the document.

Since Broker triggers are not yet implemented in Designer, you will use Developer for this exercise. Even so, you could do some of the work in designer; we use developer throughout this exercise so we don't have to swap IDE's continuously.

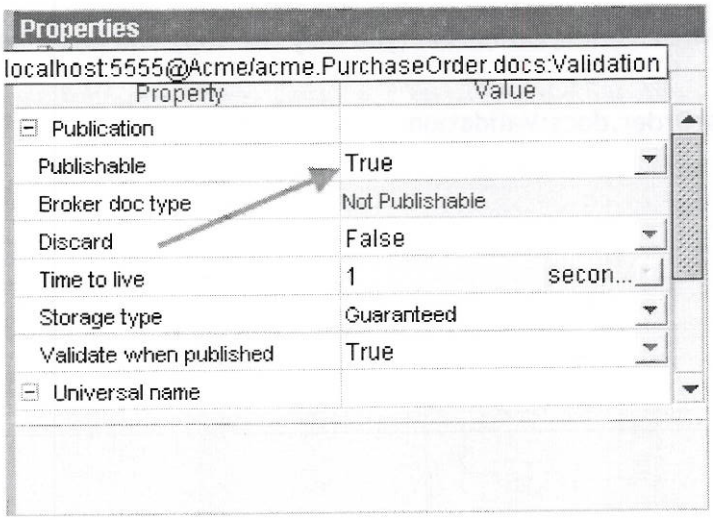
Steps

First, create the subscribing components: document type, handling service, and subscriber.

1. In the Acme package, create a **acme.PurchaseOrder.docs:Validation** document type containing one **String** field named **Valid**.

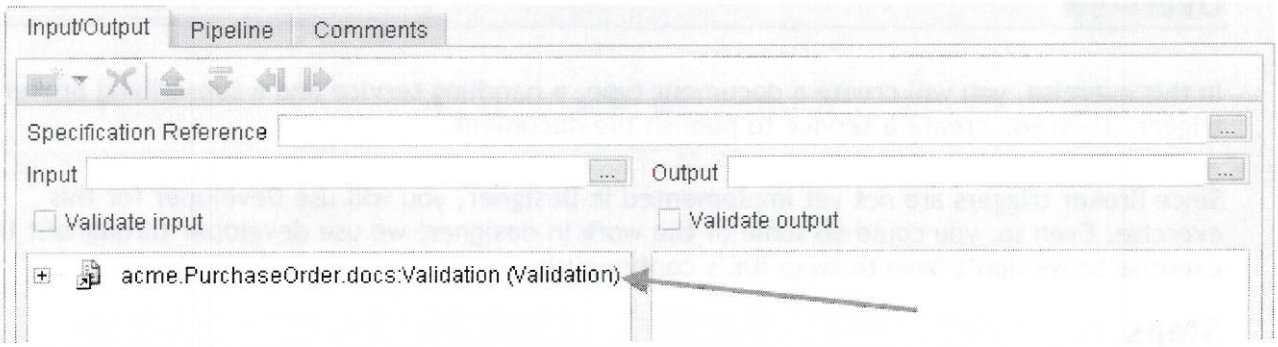


2. Make this document **publishable** to the Broker by setting the **Publishable** property for the document to **true**.

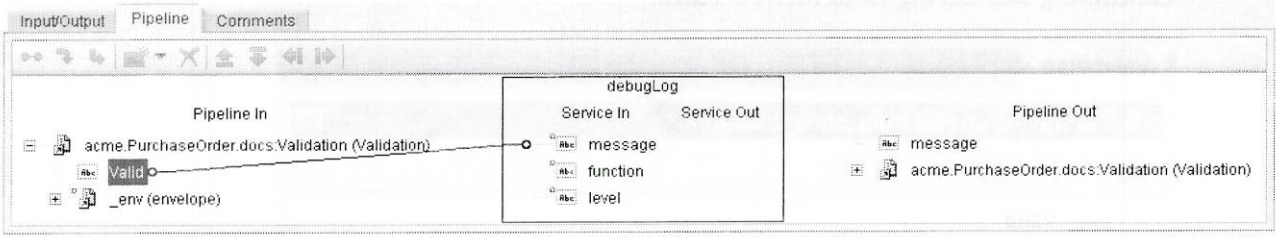


Note: if you do not see the publishable property, make sure you are viewing the properties of the Validation document itself.Open the Documemt Type in the editor and then double-click the editor's title bar of the document type to see its properties.

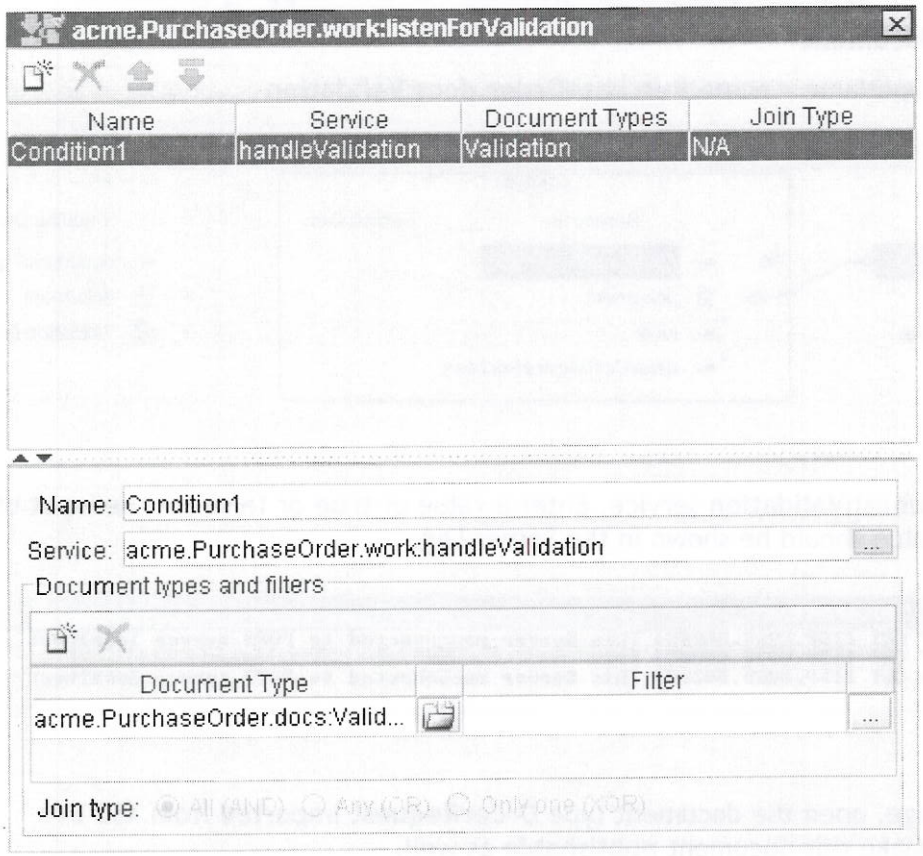
3. Create a new Flow service **acme.PurchaseOrder.work:handleValidation**. Set the input of this service to be a document reference to your document **acme.PurchaseOrder.docs:Validation**. Set the name of the document reference to be the fully-qualified name of the document type: **acme.PurchaseOrder.docs:Validation.** (Copy and paste the document type name rather than type it!)



4. In the service, add a **pub.flow:debugLog** step and map **Valid** to **message.**

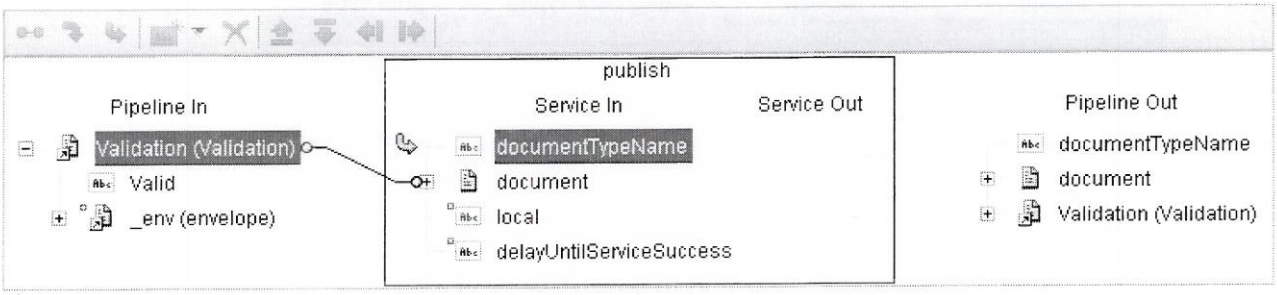


5. In the **acme.PurchaseOrder.work** folder, create a new Broker/Local Trigger, name it **listenForValidation**. Configure the trigger as follows:
   a. Name = Condition1
   b. Service = acme.PurchaseOrder.work:handleValidation (you may use copy and paste here as well)
   c. Document type = acme.PurchaseOrder.docs:Validation
   d. Filter = *leave this empty*

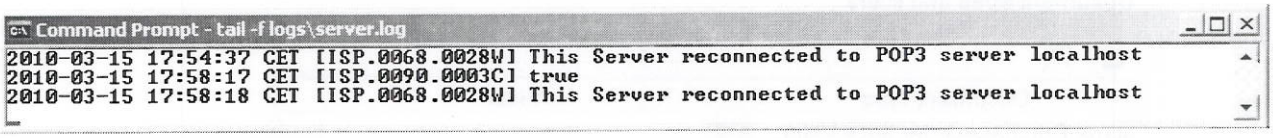


6. Save the trigger. If you want to test you work so far, double-click the **acme.PurchaseOrder.docs:Validation** document in the Navigation view and click the run button. Enter some message int the **Valid** field, leave all other fields unchanged and click next. Then select "Publish to the Broker" and click finish. Your message must appear in the Integration Server log.
7. Next, create the publishing service to publish the publishable document and test your subscription components. Create an **acme.PurchaseOrder.work:publishValidation** Flow service. In the Input of this service add a document reference to your **Acme** package's **acme.PurchaseOrder.docs:Validation.** Name it **Validation.**

8. In the service, add a step to call **pub.publish:publish**. Perform mapping as follows:
   a. **Validation** to **document**
   b. Set documentTypeName = **acme.PurchaseOrder.docs:Validation**



9. Save and run the **publishValidation** service. Enter a value of **true** or **false** for the input of this service. This value should be shown in the Server Log.



10. In your Acme package, open the document type **OrderRequest** imported from XSD in a previous exercise. Make this document **publishable** as well.

Check Your Understanding

1. What happens when a document is made publishable? The icons changes from regular document to a document with a envelope and document to envelope is used
2. What would be the appropriate production settings for publishable properties **Discard** and **Time to Live** if the Storage type = Guaranteed?
3. What two objects are required for publishing?
4. What three objects are required for subscribing?
5. Why were you required to use the full document type name as argument name in the handleValidation Service?

Exercise 17:
JMS Pub/Sub

Overview

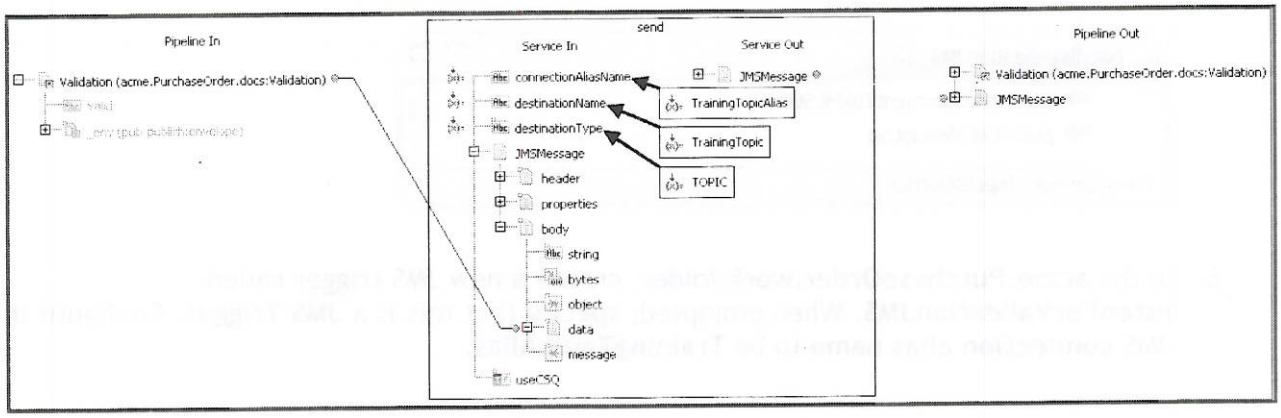
In this exercise, you will create a service to publish an existing document via an existing JMS Topic. Then you create a handling service and a subscribing JMS trigger to receive the document instance.

We will use the existing **acme.PurchaseOrder.docs:Validation** document type and publish it by using a JMS send service. Nothing at the Document needs to be changed in order to use it.

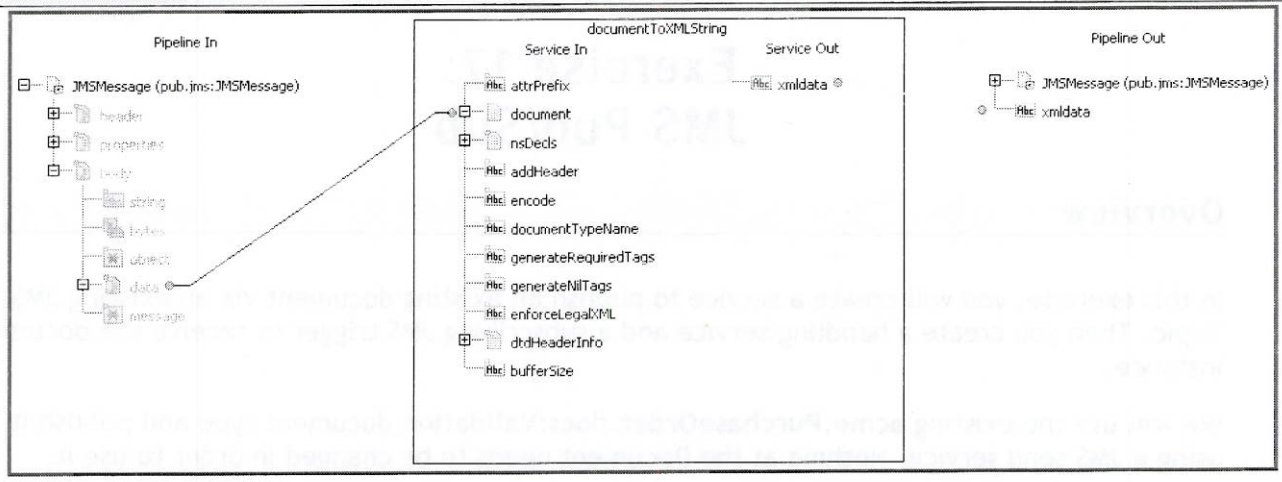
Steps

1. Create a Flow service called **acme.PurchaseOrder.work:publishValidationJMS**. This is used to publish an already created document type. In the Input of this service add a document reference to your **Acme** package's **acme.PurchaseOrder.docs:Validation.** Name it **Validation.**
2. In the new service, add a step to call **pub.jms:send**. Perform mapping as follows:
   a. **Set** connectionAliasName = TrainingTopicAlias
   b. **Set** destinationName = TrainingTopic
   c. **Set** destinationType = TOPIC
   d. **Map** Validation to JMSMessage/body/data

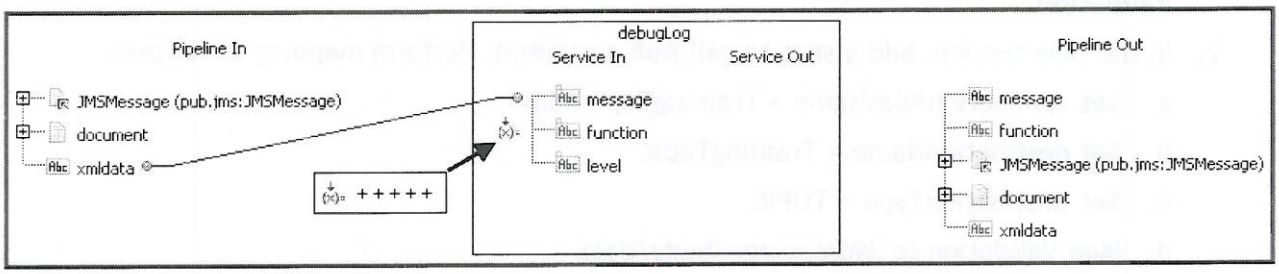
   When you are done editing your service, save the **publishValidationJMS** service.



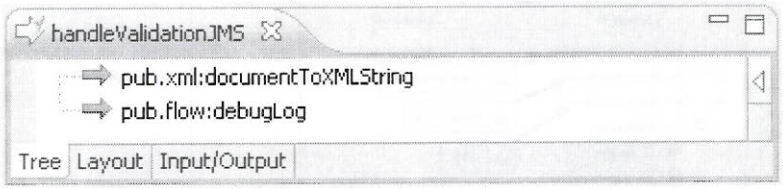
3. Create a new Flow service **acme.PurchaseOrder.work:handleValidationJMS**. This service does the handling service and trigger for subscription. On the Input/Output tab, click the ... button to the right of the **Specification Reference** field. Browse for and select **pub.jms:triggerSpec** in the WmPublic package.
4. In the service, add a **pub.xml:documentToXMLString** step and map **JMSMessage/body/data** to **document**.



5. Next add an invocation of **pub.flow:debugLog** to the service and map **xmldata** to **message**. As our eyecatcher set the value of **function** to "+ + + + + ". Of course, you do not enter the Quotes.

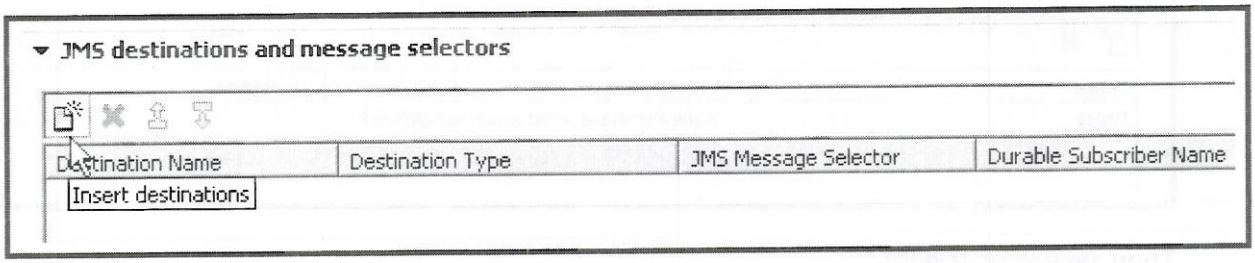


Your finished service should look like this:



6. In the **acme.PurchaseOrder.work** folder, create a new JMS trigger called **listenForValidationJMS.** When prompted, specify that this is a **JMS Trigger**. Configure the **JMS connection alias name** to be **TrainingTopicAlias.**

7. Configure the triggers **JMS destinations and message selectors** by clicking the **Insert destinations** button.

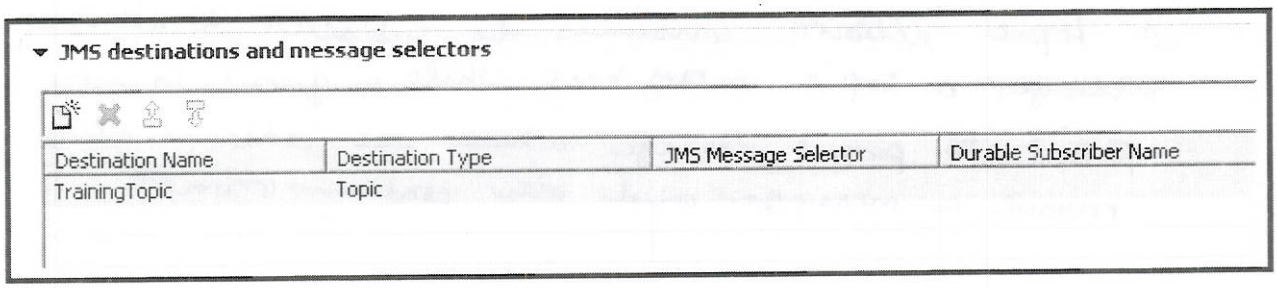


Then enter the following:

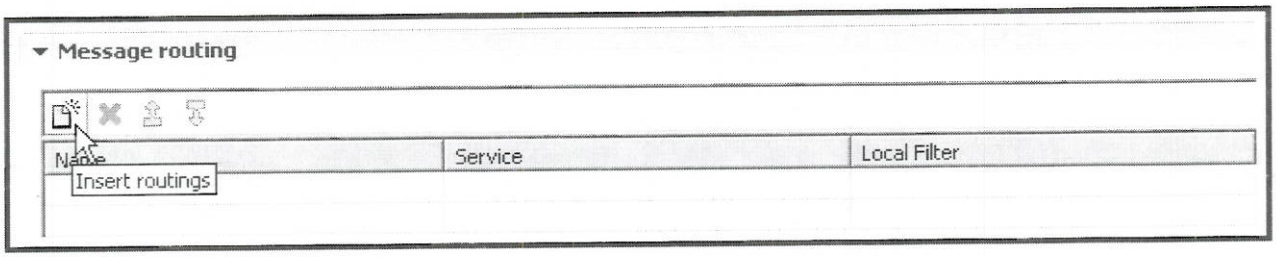
a. Destination name = TrainingTopic. If this is not in dropdown, you have to create it by selecting the "create new destination" button.

b. Destination Type = Topic

Leave all other fields empty



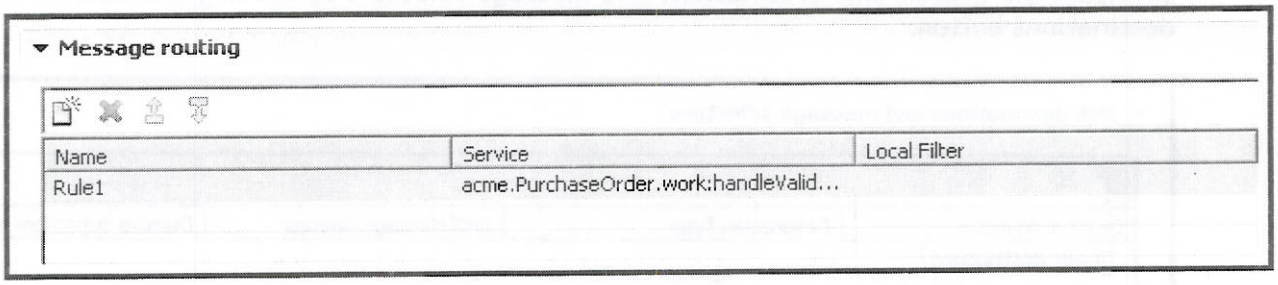
8. Configure the trigger **Message routing** by clicking the **Insert routings** button.



Then enter the following:

a. Name = Rule1

b. Service = acme.PurchaseOrder.work:handleValidationJMS



Then save your trigger.

9. Run the **publishValidationJMS** service. Enter a value of **true** or **false** for the input of this service. This value should be shown, embedded in an XML document, in the Server Log.

Check Your Understanding

1. What is a topic versus a queue?

a topic allows consumers to subscribe to all the messages a topic may have while a queue is more point to point message where one consumer may receive a message and other consumers can not.

Exercise 18:
Create Adapter Services

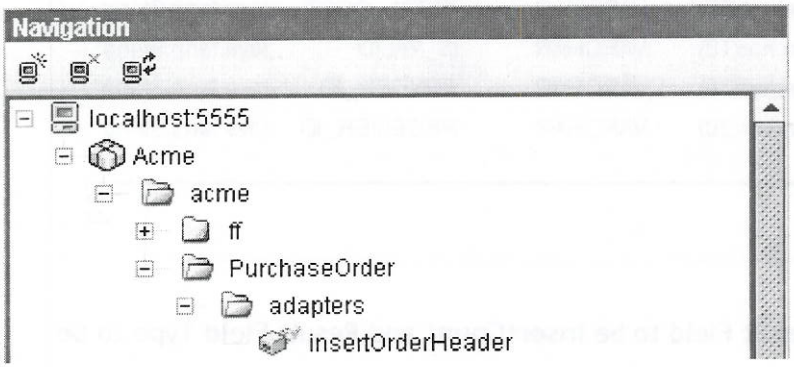
Overview

In this exercise, you will create insert and select adapter services. You combine these with a parent flow service. Take these together and you can easily work with data in a database.

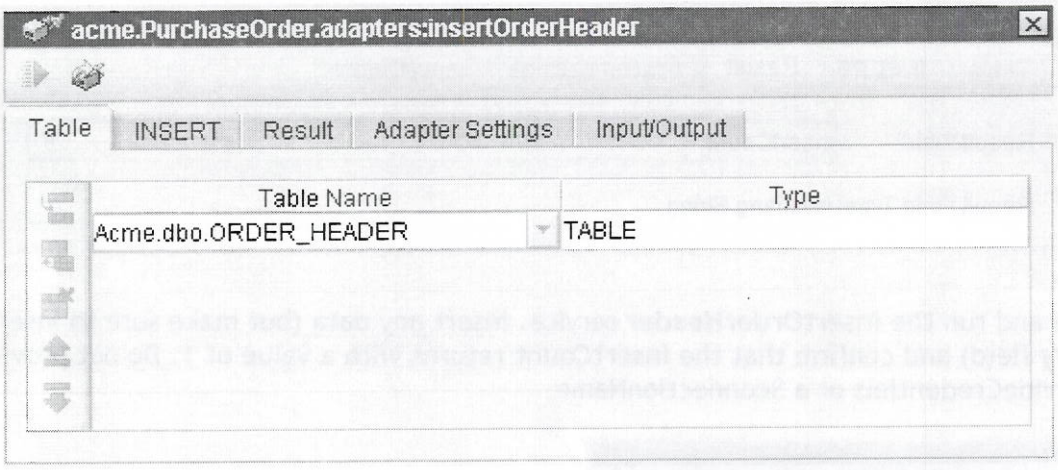
This exercise will be done with Developer.

Steps

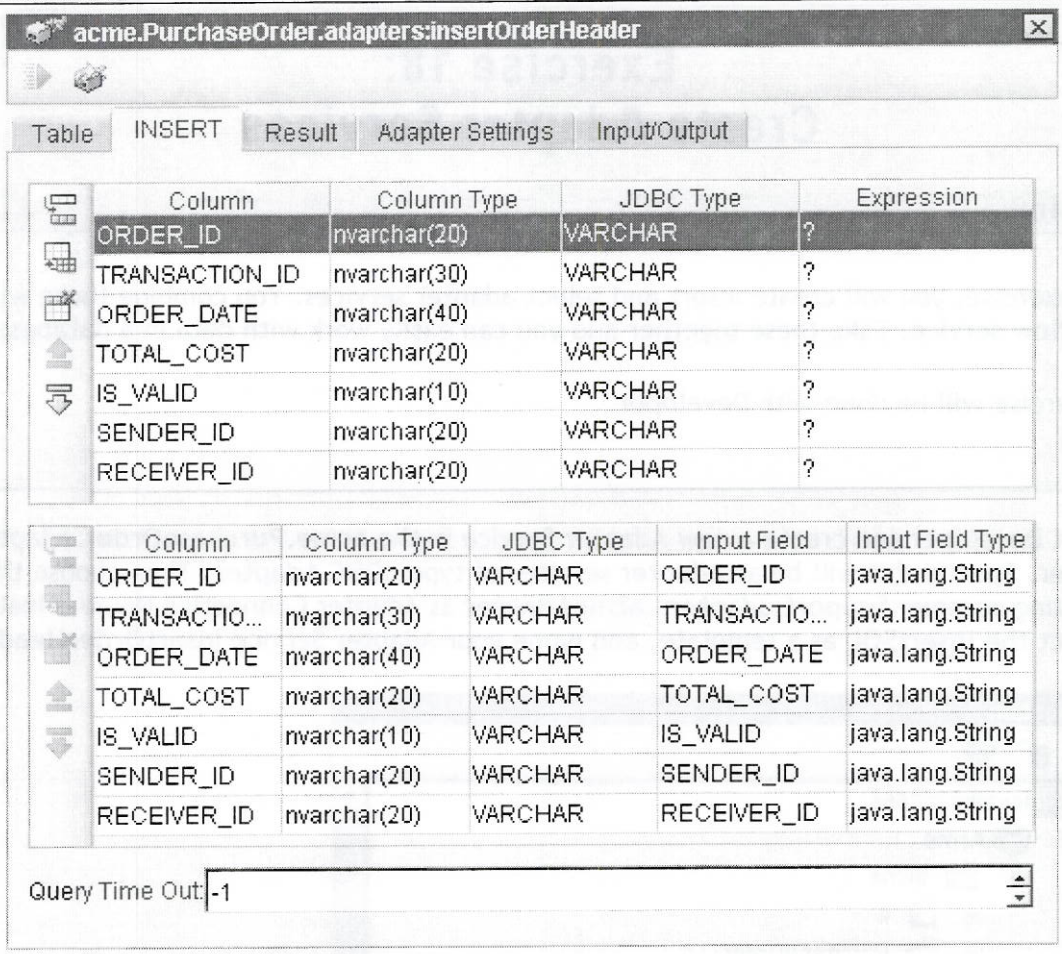
1. Start Developer and create a new Adapter Service in the **acme.PurchaseOrder.adapters** folder. Specify this will be an adapter service, of type **JDBC Adapter**. Then choose the existing **commonSupport.adapters.acmeAdapter** as Adapter Connection Name. Finally select the **InsertSQL** as a template, and name your Adapter Service **insertOrderHeader**.



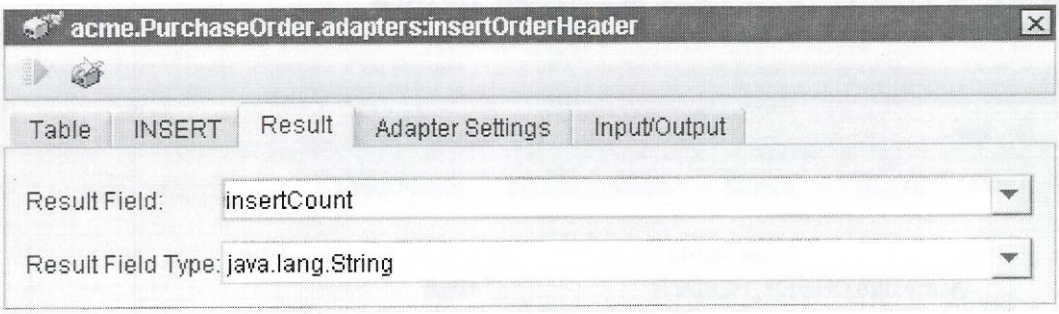
   a. On the Table tab, select **Acme.dbo.ORDER_HEADER**.



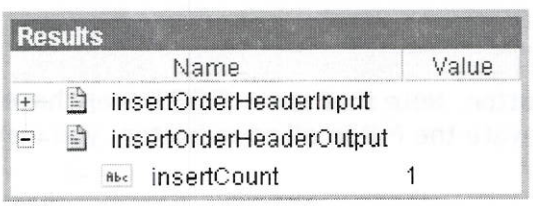
   b. On the Insert tab, click the **Fill in all rows** button. Note that some JDBC Drivers have Problems with this operation. In order to activate the **Fill in all rows** button, you may have to press the **Insert row** button once.



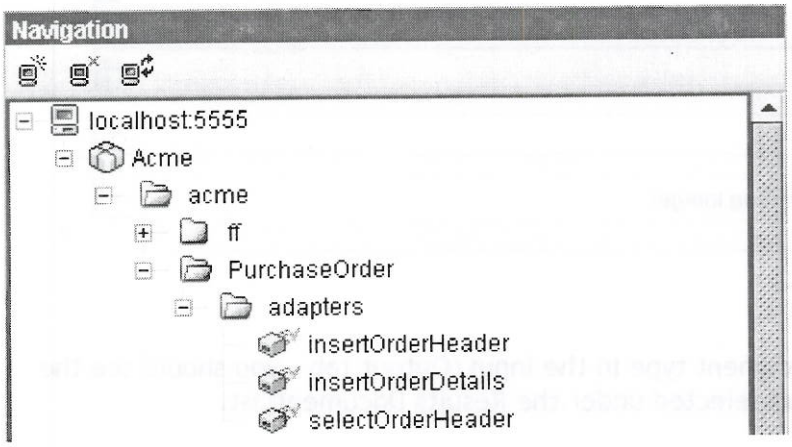
c. On the Result tab, set Result Field to be **insertCount** and Result Field Type to be **java.lang.String**.



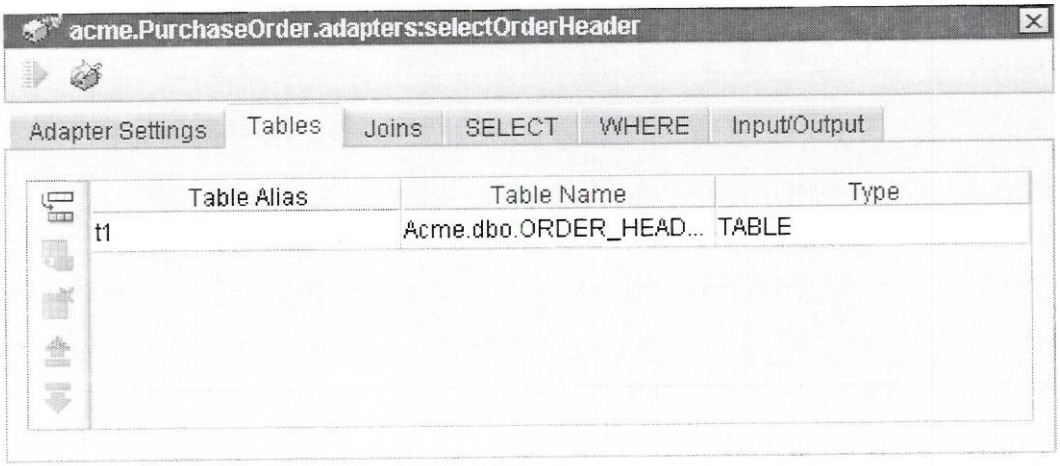
2. Save and run the **insertOrderHeader** service. Insert any data (but make sure to insert for every field) and confirm that the **insertCount** returns with a value of 1. Do not provide overrideCredentials or a $connectionName.



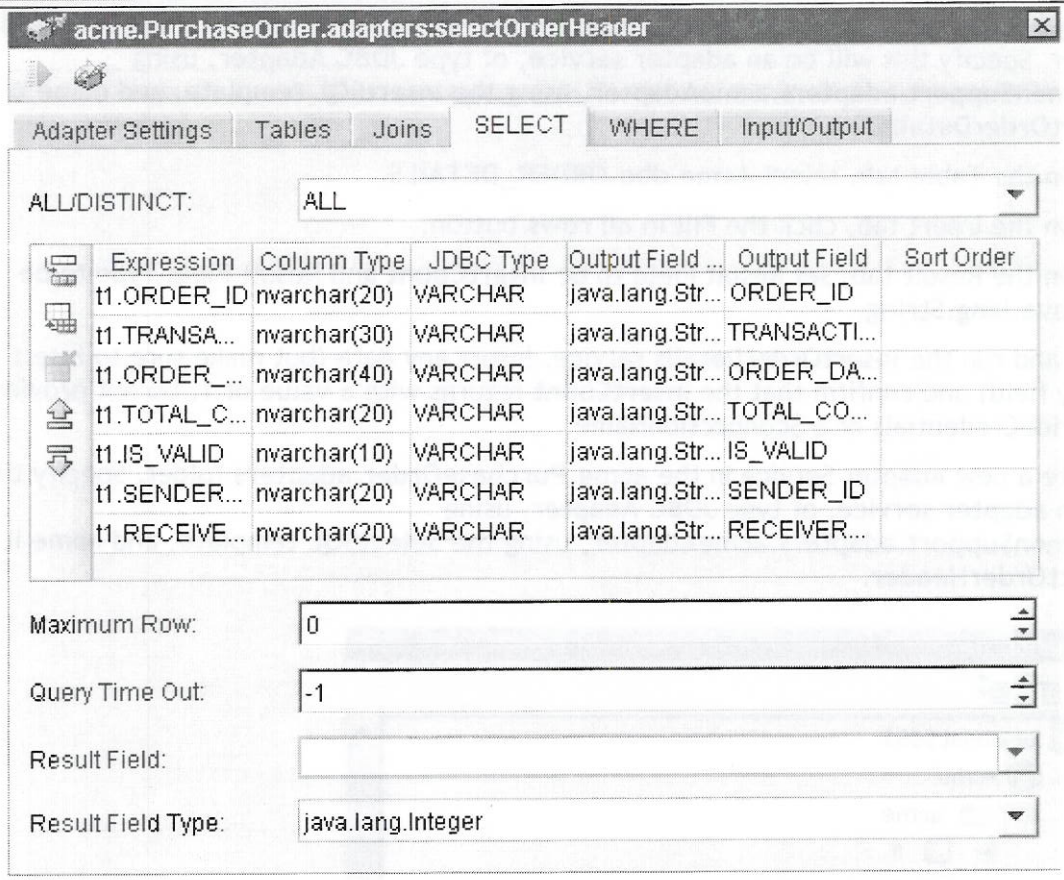
3. Like you did in step 1, create another Adapter Service in the **acme.PurchaseOrder.adapters** folder. Specify this will be an **adapter service**, of type **JDBC Adapter**, using **commonSupport.adapters.acmeAdapter**, using the **InsertSQL** template, and name it **insertOrderDetails**.
   a. On the Table tab, select **Acme.dbo.ORDER_DETAILS**.
   b. On the Insert tab, click the **Fill in all rows** button.
   c. On the Result tab, set Result Field to be **insertCount** and Result Field Type to be **java.lang.String**.
4. Save and run the **insertOrderDetails** service. Insert any data (but make sure to insert for every field) and confirm that the **insertCount** returns with a value of 1. Do not provide overrideCredentials or a $connectionName.
5. Create a new Adapter Service in the **acme.PurchaseOrder.adapters** folder. Specify this will be an **adapter service**, of type **JDBC Adapter**, using **commonSupport.adapters.acmeAdapter**, using the **SelectSQL** template, and name it **selectOrderHeader**.



   a. On the Tables tab, in the **"Table Name"** column select **Acme.dbo.ORDER_HEADER**

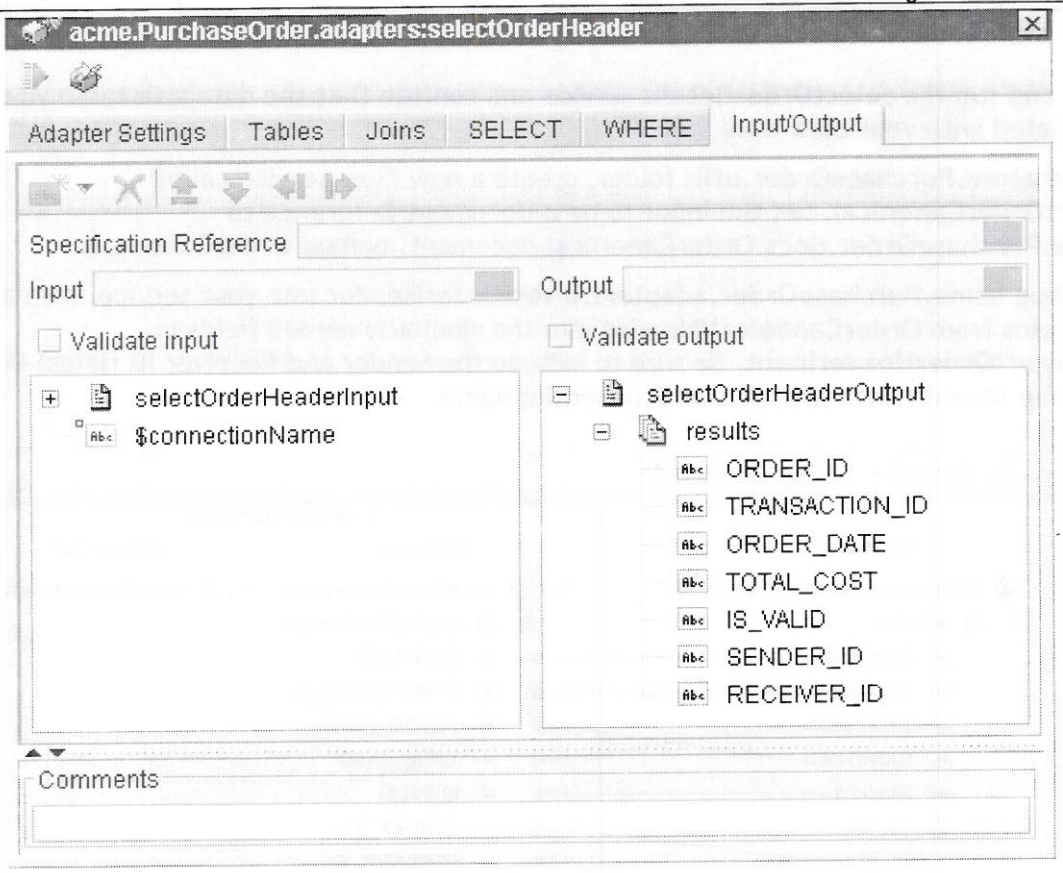


   b. Skip the Joins tab
   c. On the Select tab, specify **ALL**, and click **Fill in all rows**

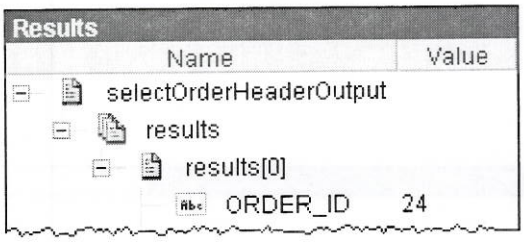


d. Skip the Where tab

e. Review the generated Document type in the Input/Output tab - you should see the database columns that you selected under the Results DocumentList.

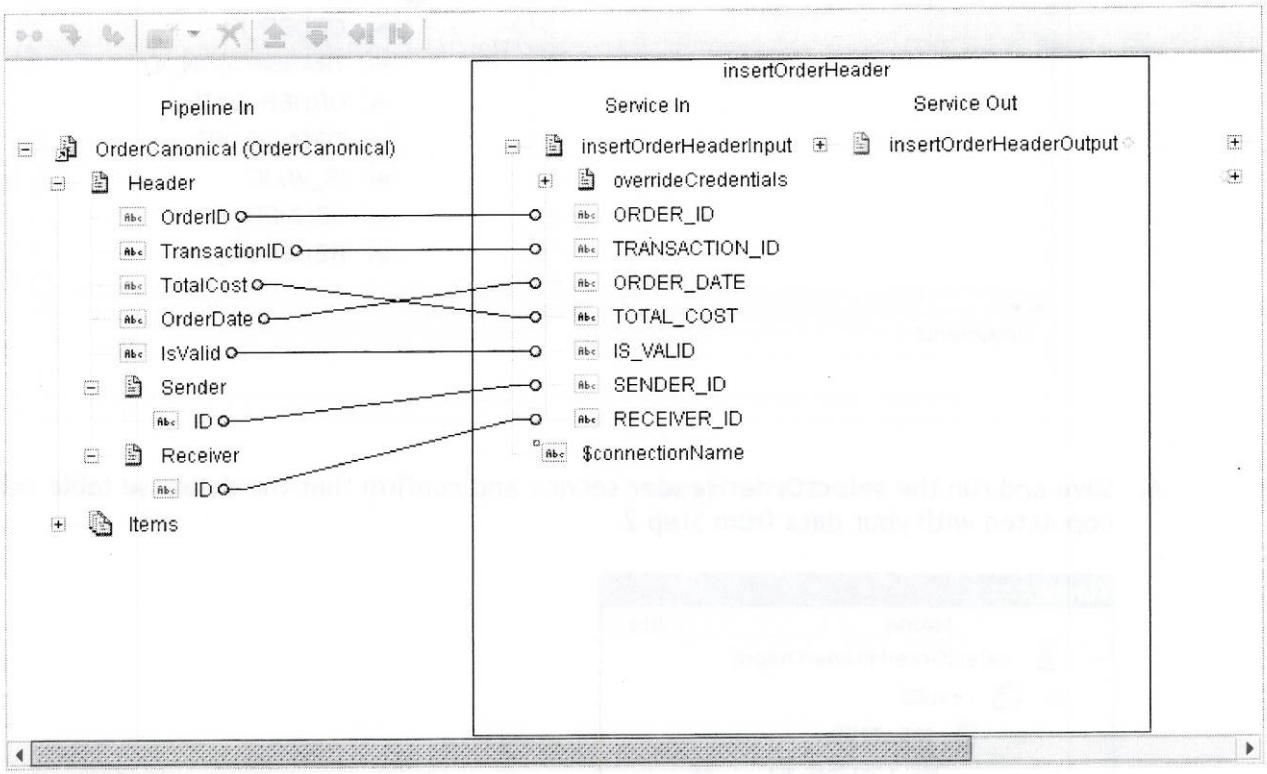


6. Save and run the **selectOrderHeader** service and confirm that the database table was populated with your data from Step 2.



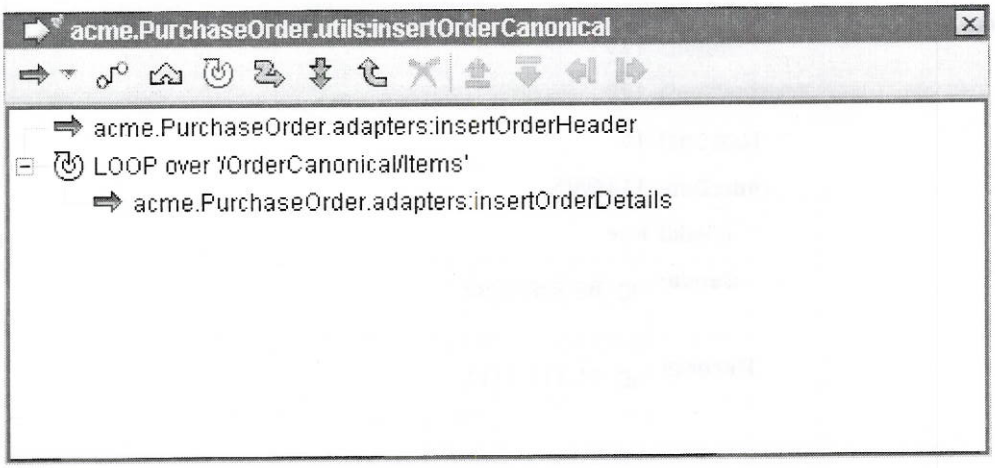
7. Create another Adapter Service in the **acme.PurchaseOrder.adapters** folder. Specify this will be an **adapter service** which is of type **JDBC Adapter** and is using **commonSupport.adapters.acmeAdapter.** Then choose the **SelectSQL** template and name the Adapter Service **selectOrderDetails.**
   a. On the Table tab, select **Acme.dbo.ORDER_DETAILS**
   b. Skip the Joins tab
   c. On the Select tab, specify **ALL**, and click **Fill in all rows**
   d. Skip the Where tab
   e. Review the generated Document type in the Input/Output tab - you should see the database columns that you selected under the Results DocumentList

8. Save and run the **selectOrderDetails** service and confirm that the database table was populated with your data from Step 4.
9. In the **acme.PurchaseOrder.utils** folder, create a new Flow service called **insertOrderCanonical.** Set the input to be a Document Reference to **acme.PurchaseOrder.docs.OrderCanonical** document, named **OrderCanonical.**
   a. Drag **acme.PurchaseOrder.adapters.insertOrderHeader** into your service. Map all the fields from **OrderCanonical/Header** into the similiarly named fields in **insertOrderHeaderInput.** Be sure to include the Sender and Receiver ID fields! Do not map to overrideCredentials or $connectionName.



   b. Add a **LOOP** step and set the Input array property to **OrderCanonical/Items.**

c. Drag **acme.PurchaseOrder.adapters.insertOrderDetails** into your service and indent it under the LOOP step.



In the invocation of the insertOrderDetails make sure that OrderCanonical/Items is no longer an array.

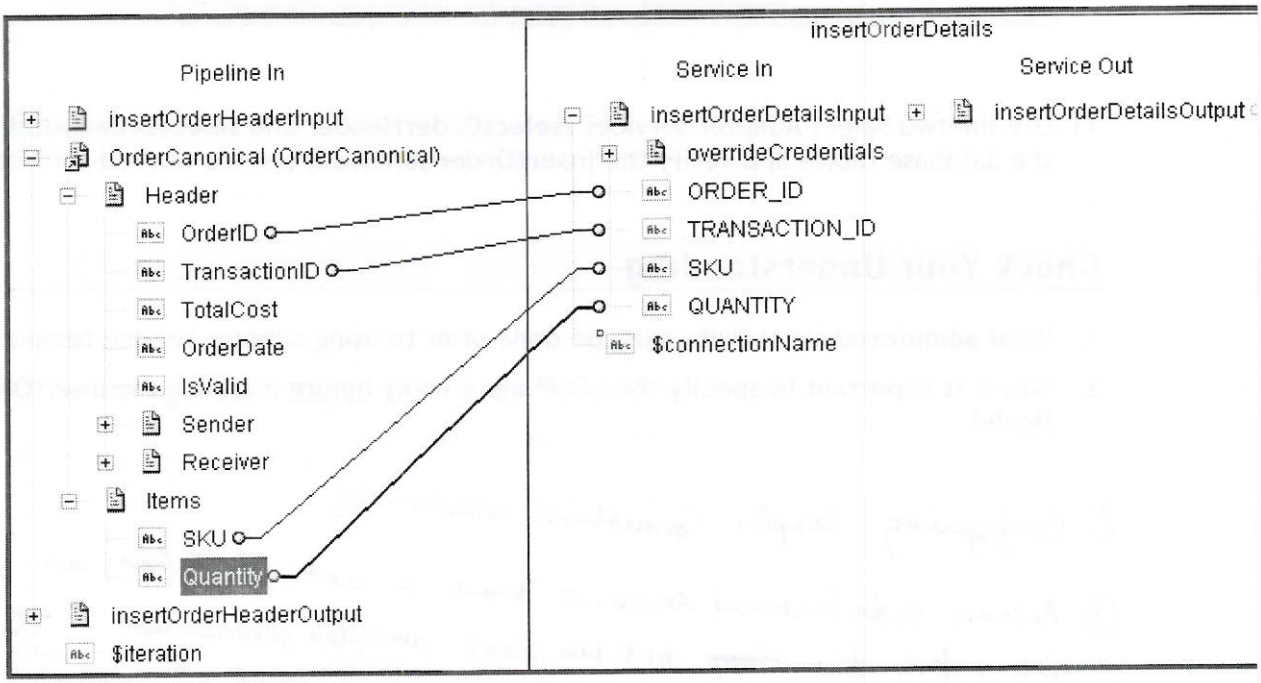
Then map as follows:

i. OrderCanonical/Header/OrderID to insertOrderDetailsInput/ORDER_ID

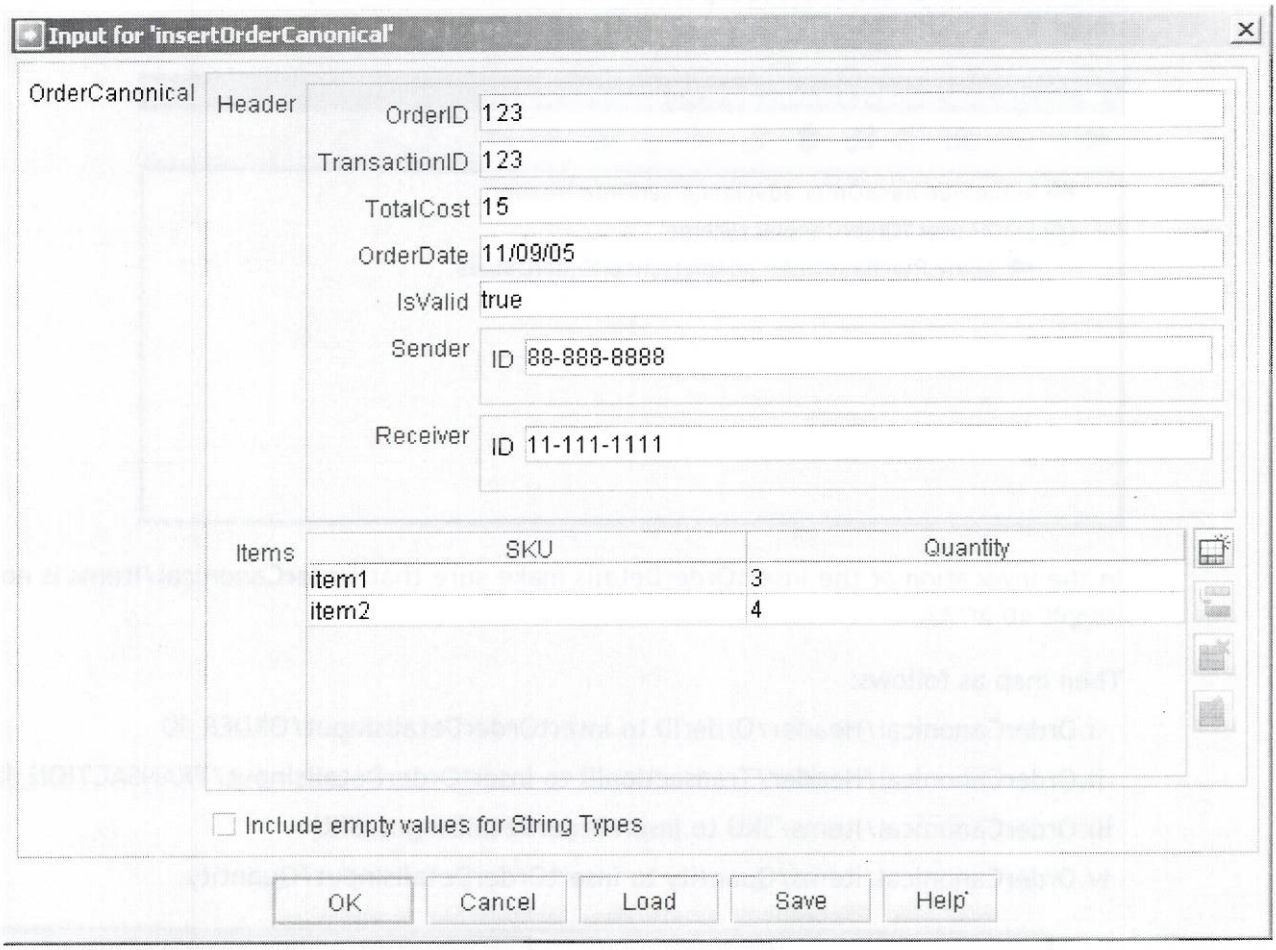
ii. OrderCanonical/Header/TransactionID to insertOrderDetailsInput/TRANSACTION_ID

iii. OrderCanonical/Items/SKU to insertOrderDetailsInput/SKU

iv. OrderCanonical/Items/Quantity to insertOrderDetailsInput/Quantity



10. Save and run the **acme.PurchaseOrder.utils:insertOrderCanonical** service by loading the data from the file **...\IntegrationServer\packages\AcmeSupport\pub\order_canonical_input.txt.**



11. Use the two Select Adapter services (**selectOrderHeader** and **selectOrderDetails**) to check the database tables and verify the **insertOrderCanonical** service worked correctly.

Check Your Understanding

1. What administrative activity must be done prior to using adapter service templates?
2. Why is it important to specify the LOOP input array before mapping the insertOrderDetails fields?

① Configuring Adapter Connection within IS

② Because OrderCanonical document had a set of Items so we wanted it as input ~~to loop~~ but the insert adapter service that it one document instead of ~~a~~ an array of document. By placing the label on the Loop, it makes the mapping seems like one document instead of a Document List in which ~~is~~ adapter is not looping for.

Exercise 19:
Adapter Notifications

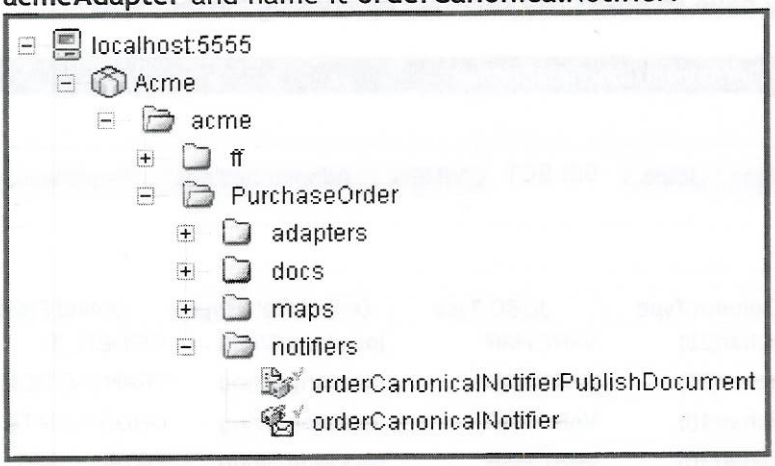
Overview

In this exercise, you will create a notification service to watch for database inserts. As new orders are entered into the database from the ordering interface, a notification document should be published so that interested systems and services can become aware of the insertion.

This exercise will also be done using Developer.

Steps

1. Create a new Adapter Notification in the **acme.PurchaseOrder.notifiers** folder. Specify this will be of type **JDBC Adapter**, an **InsertNotification**, using **commonSupport.adapters. acmeAdapter** and name it **orderCanonicalNotifier**.



   a. On the Notification Configure tab, specify **ORDER** as the **base name.**