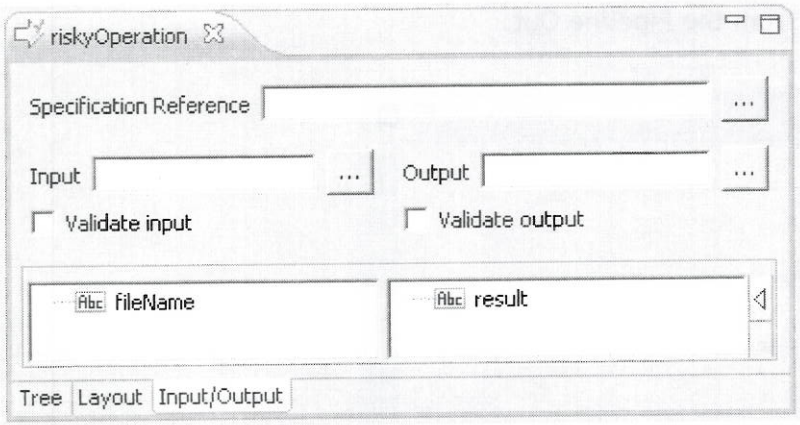
# Exercise 7:
# Building Flow Services – SEQUENCE

## Overview

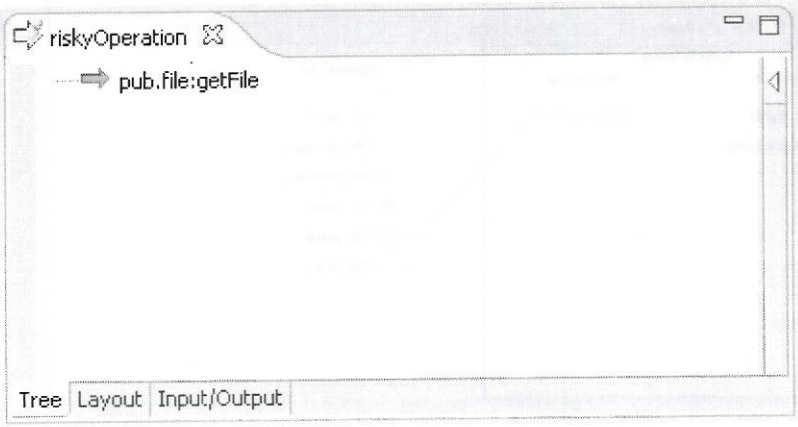
In this exercise, you will create business logic for a Try/Catch using sequence in a Flow service. We will create a service that will return an exception, and then embed that service in a Try/Catch sequence.

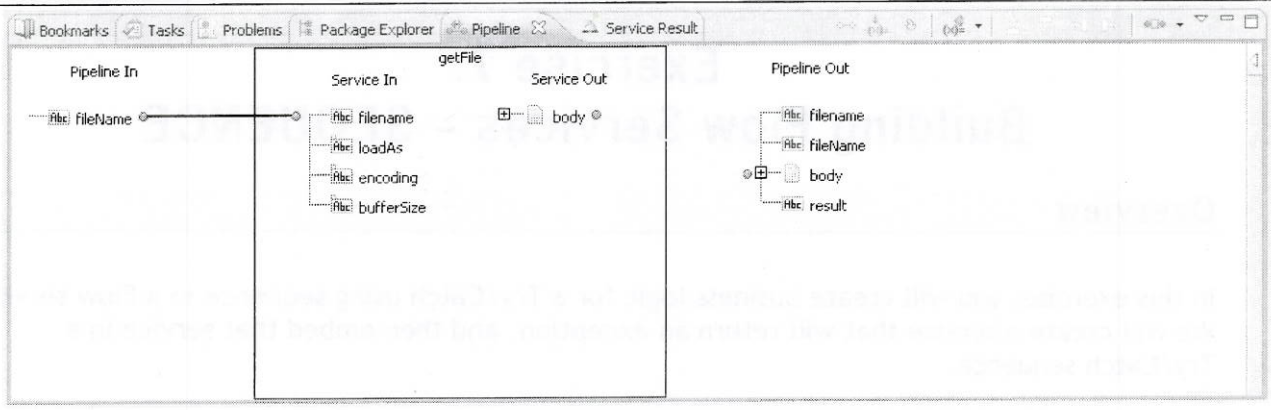
## Steps

1. In the **acme.PurchaseOrder.work** folder, create a new Flow service called **riskyOperation**. Set the input to be a String called **fileName** and the output to be a String called **result**.

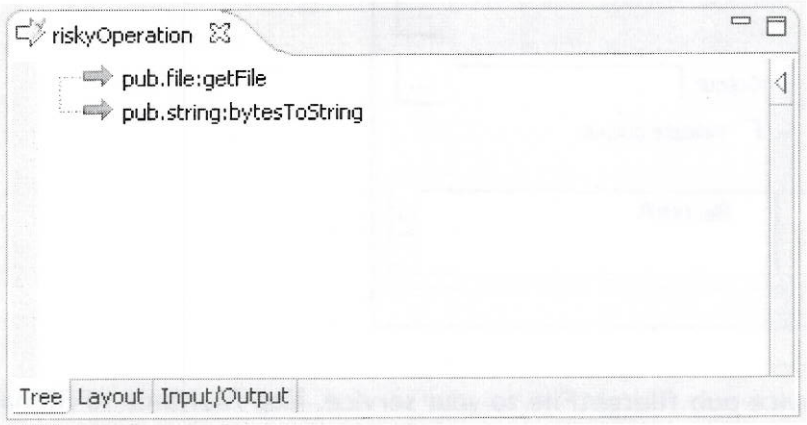


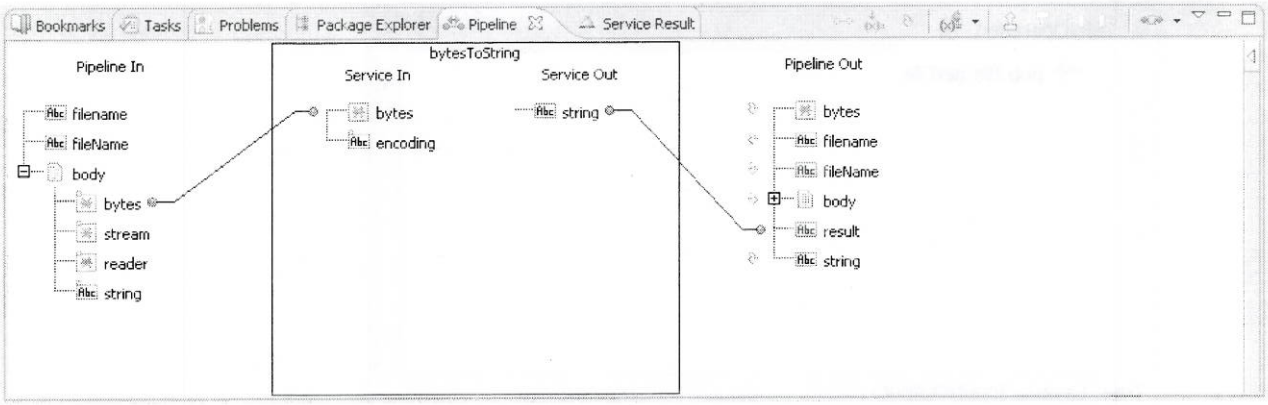

2. Add an invocation of the service **pub.file:getFile** to your service. Map **fileName** to **filename**.



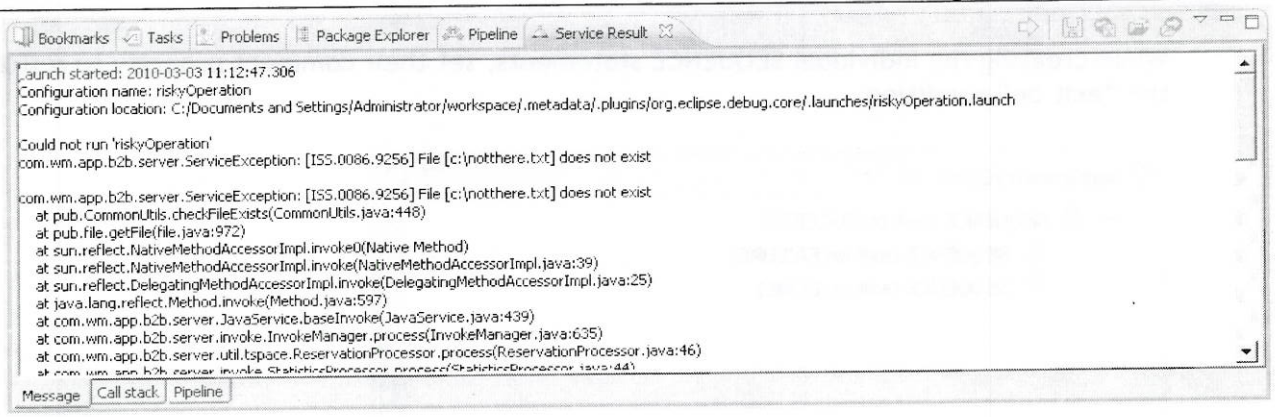


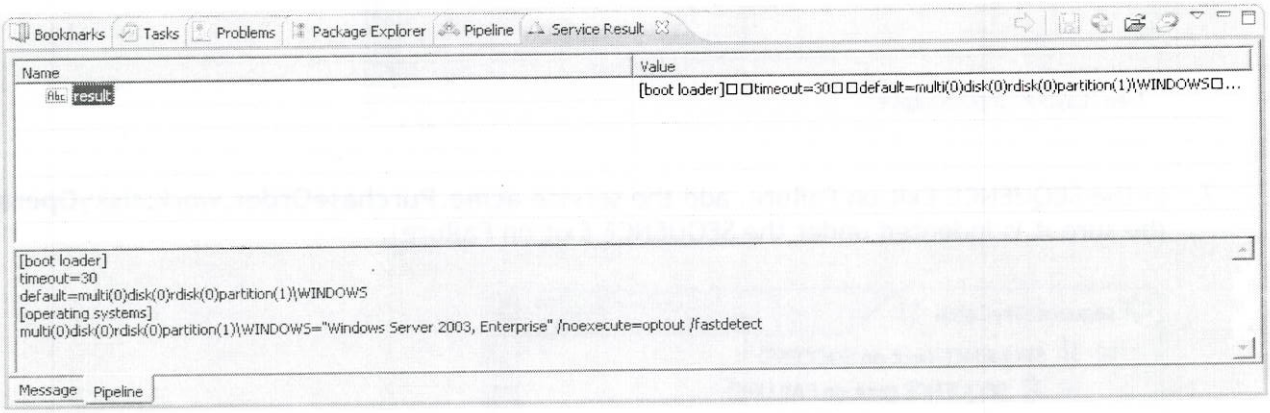

3. Add an invocation of the service **pub.string:bytesToString** to your service.
   a. Map **body/bytes** to **bytes** and **string** to **result**.
   b. Drop all other variables from the Pipeline Out.



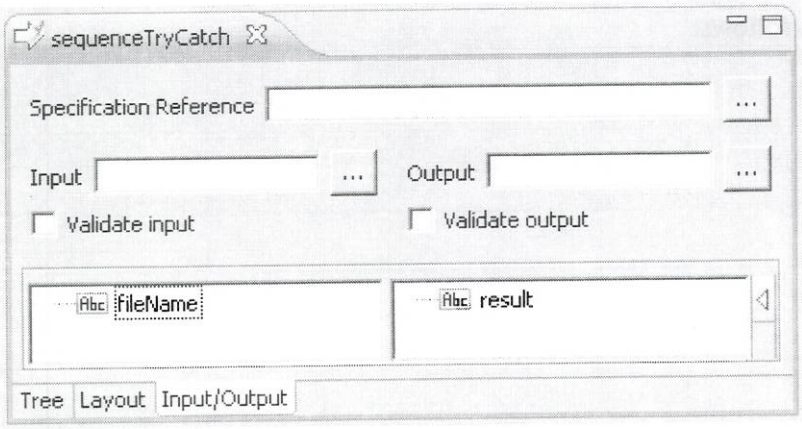




4. Save and run the service. Provide it with a **fileName** of **c:\notthere.txt** (or any other file that does not exist!) What result do you receive? Also try the service with an existing file like **c:\boot.ini**. What result do you receive then?



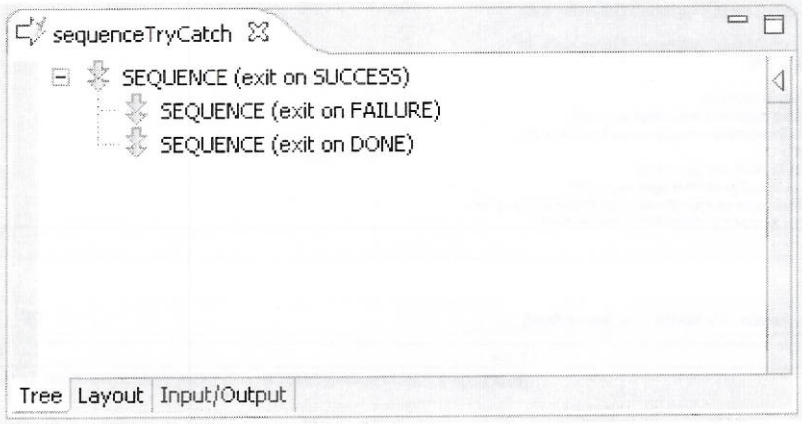




5. Now create another service in the **acme.PurchaseOrder.work** folder called **sequenceTryCatch.** We will use this service to allow us to catch the exception that **riskyOperation** raises if the file name is not correct. Define a single input String for the service called **fileName** and a single output String called called **result.**

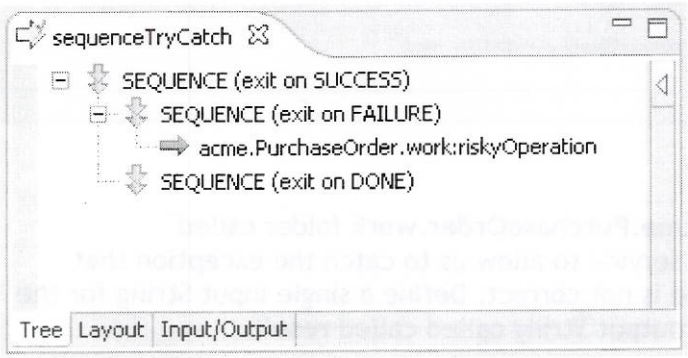



6. In the sequenceTryCatch service, add three **SEQUENCE** steps to create a Flow try/catch block, as follows:
   a. SEQUENCE set to Exit on Success
      i. SEQUENCE set to Exit on Failure (be sure it is indented under the first SEQUENCE)
      ii. SEQUENCE set to Exit on Done (be sure it is indented under the first SEQUENCE)

While creating the individual SEQUENCE statements, set their comment property to reflect the "exit on" condition.



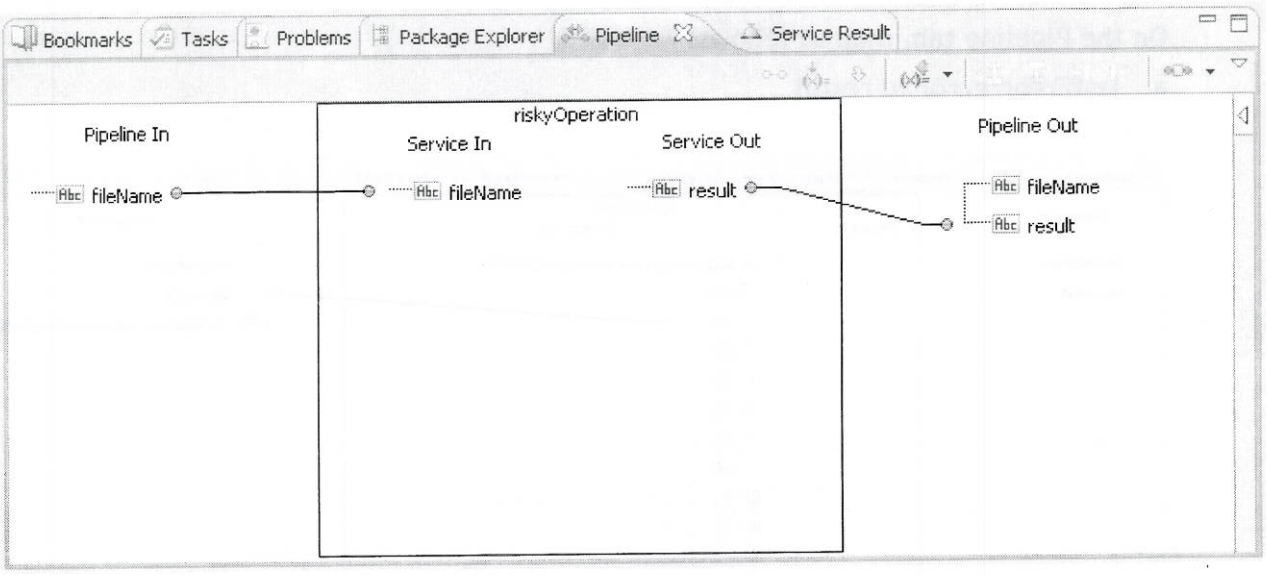

7. In the SEQUENCE Exit on Failure, add the service **acme.PurchaseOrder.work:riskyOperation** (be sure it is indented under the SEQUENCE Exit on Failure).



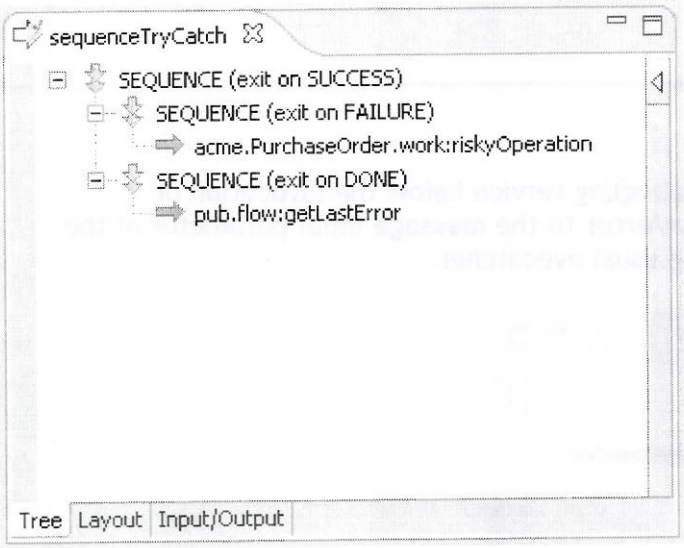

On the Pipeline tab, map as follows:

a. **fileName** to **fileName**

b. **result** to **result**

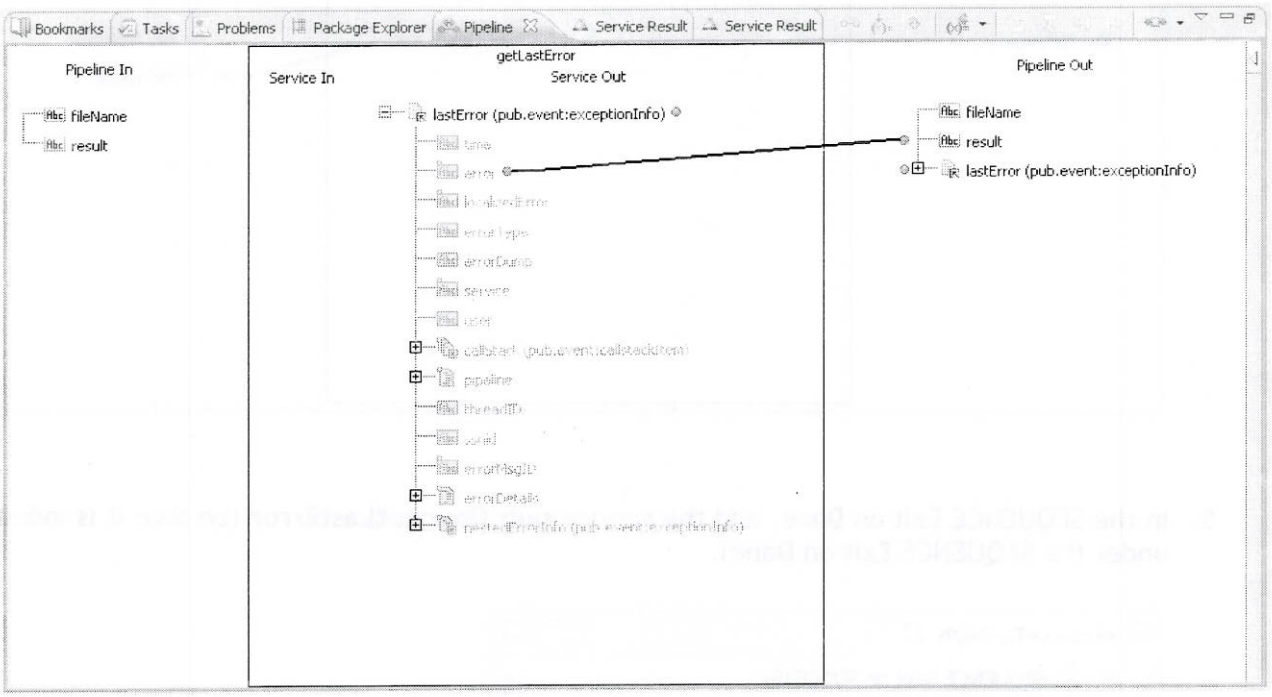



8. In the SEQUENCE Exit on Done, add the service **pub.flow:getLastError** (be sure it is indented under the SEQUENCE Exit on Done).

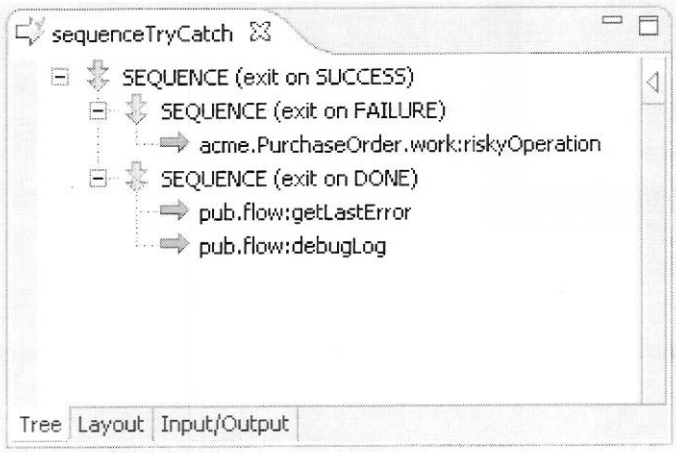


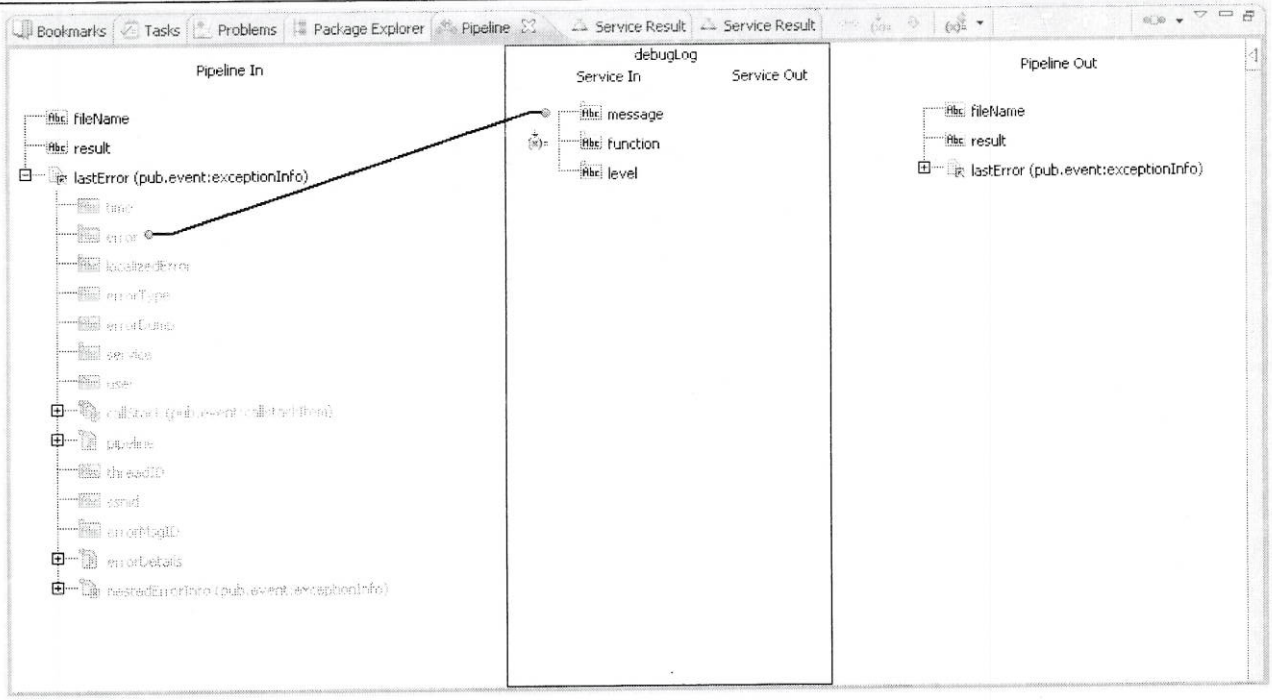
On the Pipeline tab, map as follows:

a. **lastError/error** to **result**




9. Add an invocation of the **pub.flow:debugLog** service below the invocation of **pub.flow:getLastError**. Map **lastError/error** to the **message** input parameter of the **debugLog** service. Set function to the usual eyecatcher.






10. Save and run your service. Check the Results tab and the server log.
    a. To fail, provide **c:\notthere.txt** as the file. Verify you see an error message in the server log.
    b. To succeed, provide **c:\boot.ini.** Successful execution will show the file contents in the Results tab and no error message in the server log.

    Try using the debugger in both cases to see the decision path.

## Check Your Understanding

1. Rather than using a service you know will fail, how can you throw an Exception in Flow?
2. What happens if the riskyOperation service works (doesn't fail?)

This page intentionally left blank

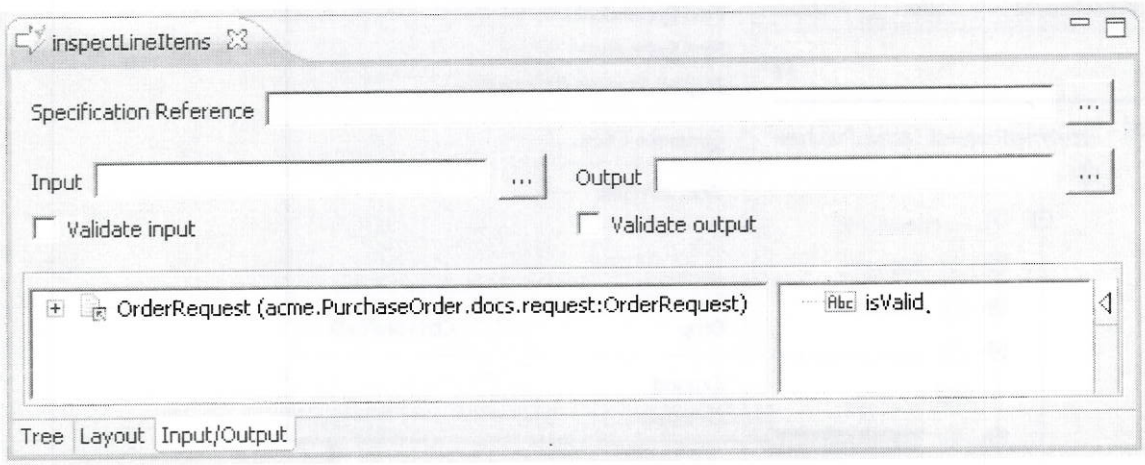
# Exercise 8:
# Validation Service

## Overview

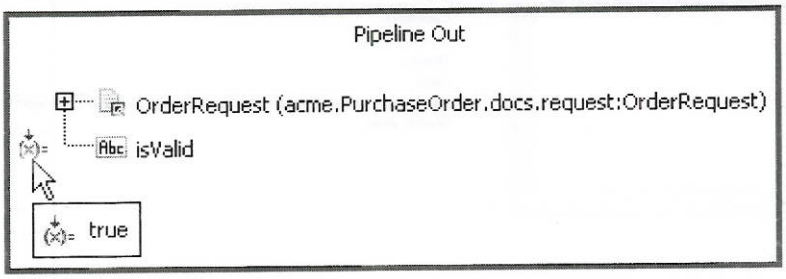
In this exercise, you will create business logic to validate an inbound Purchase Order. For now, this means to verify that the line items in the Purchase Order have a valid quantity. If this is not the case, you will flag the order as invalid for follow up by a customer service representative.

## Steps

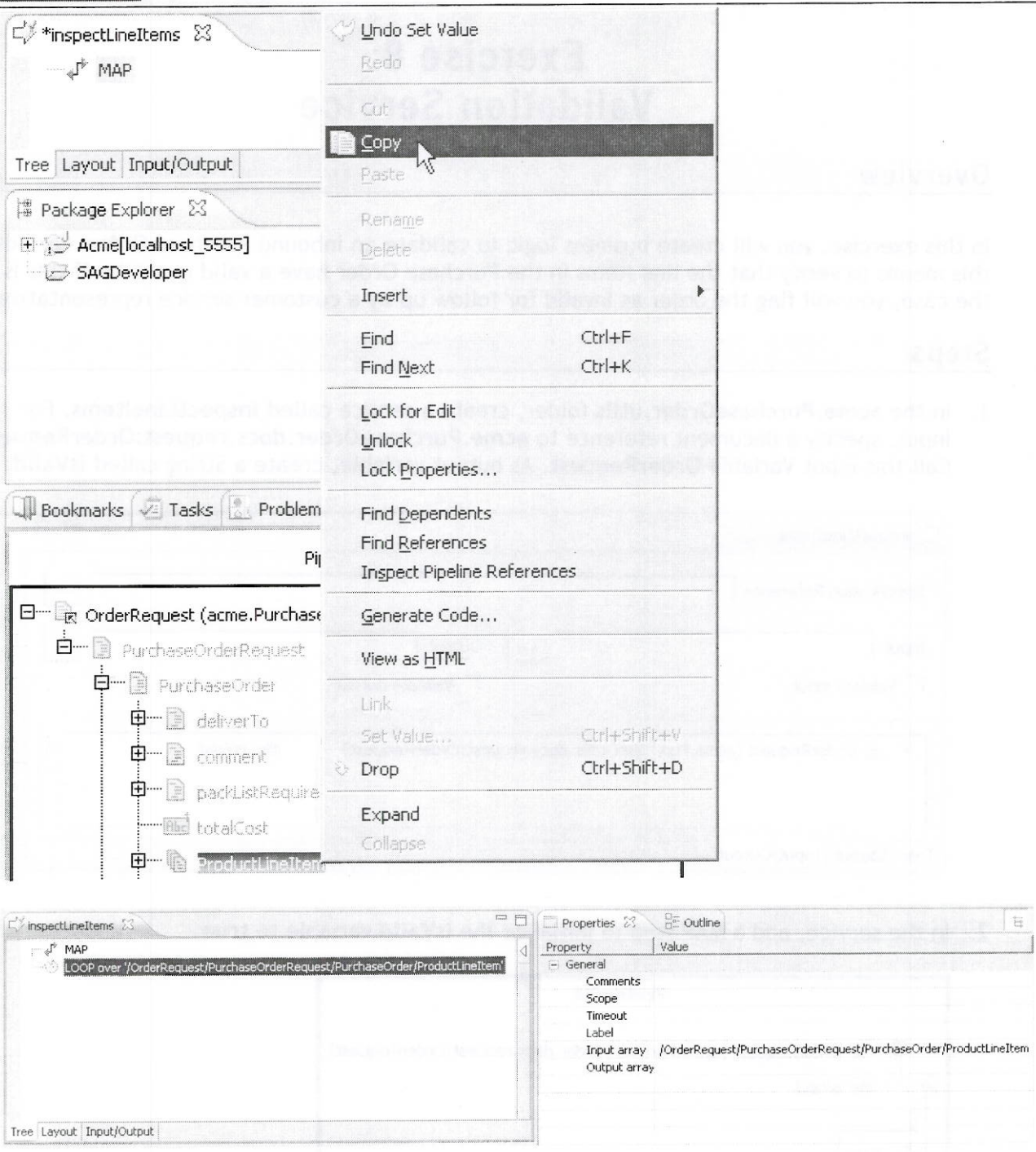
1. In the **acme.PurchaseOrder.utils** folder, create a service called **inspectLineItems.** For input, specify a document reference to **acme.PurchaseOrder.docs.request:OrderRequest.** Call this input Variable **OrderRequest.** As output variable, create a String called **isValid.**




2. In the service, add a MAP step to initialize the **isValid** variable to **true.**




3. In the Pipeline tab, locate and copy the **OrderRequest/PurchaseOrderRequest/ PurchaseOrder/ProductLineItem** field. Then add a **LOOP** step to your service and set the **Input array** property by pasting the copied value.







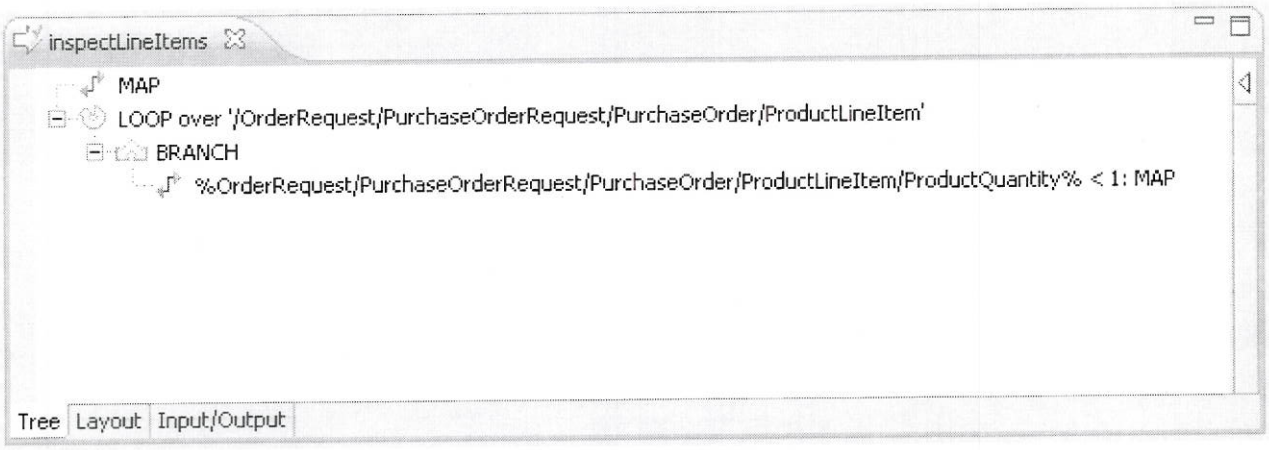
4. Under the LOOP step in your service, add a **BRANCH** step (be sure it is indented under the LOOP). Leave the **Switch** property empty and set **Evaluate labels** = **True**.

5. Now you will add code under the BRANCH. Add a MAP step below the BRANCH step (be sure it is indented under the BRANCH). In the Pipeline tab, locate and copy the **/OrderRequest/PurchaseOrderRequest/PurchaseOrder/ProductLineItem/ProductQuantity** field.

   Then type: **"%% < 1"** (six characters, no quotes, around the smaller sign are spaces) into the Label property of the MAP step.

Then click the mouse between the two percent signs and paste the copied value. You should end up with %OrderRequest/PurchaseOrderRequest/PurchaseOrder/ProductLineItem/ProductQuantity% < 1

6. In the Pipeline of the MAP step, set **isValid** = **false**. Your service should look like this:




7. Save and run the service. For input, load the file **...\IntegrationServer\packages\AcmeSupport\pub\order_request_input.txt.**

   When using this input file, **isValid** should be **true** in the Results.

   Run again and change one of the **ProductQuantity** values to 0 (in the **ProductLineItem** array). **isValid** should be **false** in the results.

8. **For extra credit**, stop processing after the first invalid **ProductLineItem**

## Check Your Understanding

1. Why did we set isValid to true at the very beginning?
2. Is there another way we could have validated this particular value with writing Flow or Java?

This page intentionally left blank

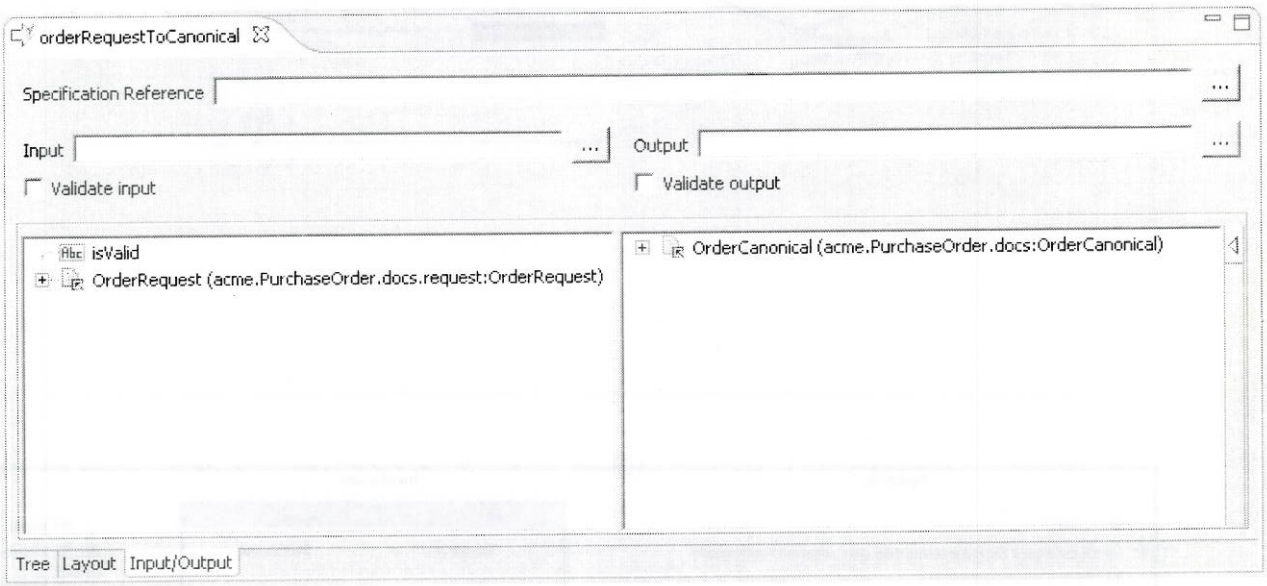
# Exercise 9:
# Mapping Service

## Overview

In this exercise, you will create a service that maps from one data format to another using the document types you created in a previous exercise.

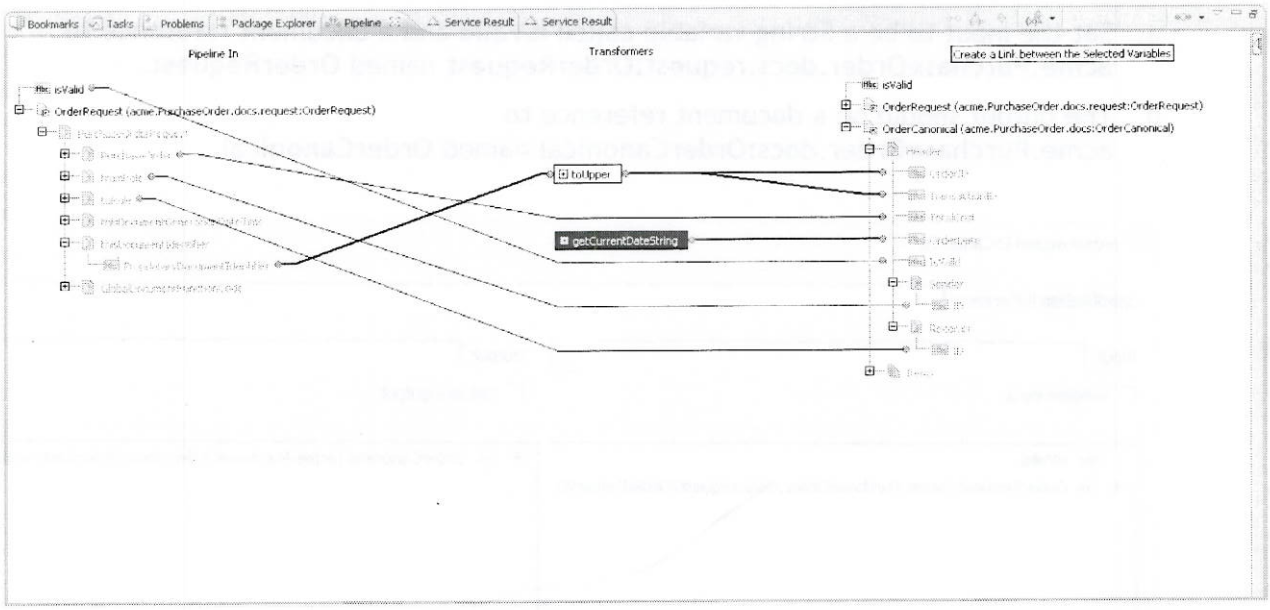
## Steps

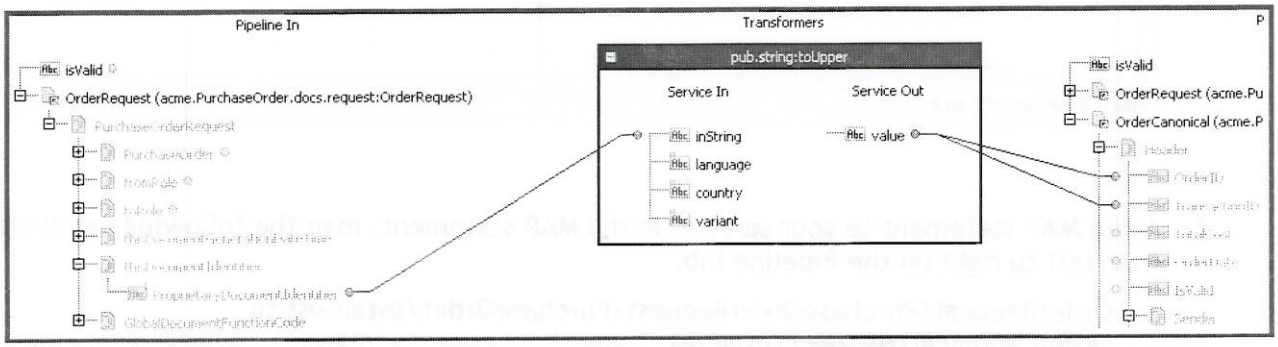
1. In the **acme.PurchaseOrder.maps** folder, create a Flow service called **orderRequestToCanonical**.
   a. Set the input to be a **String** variable called **isValid** and a document reference to **acme.PurchaseOrder.docs.request:OrderRequest** named **OrderRequest.**
   b. The output should be a document reference to **acme.PurchaseOrder.docs:OrderCanonical** named **OrderCanonical.**

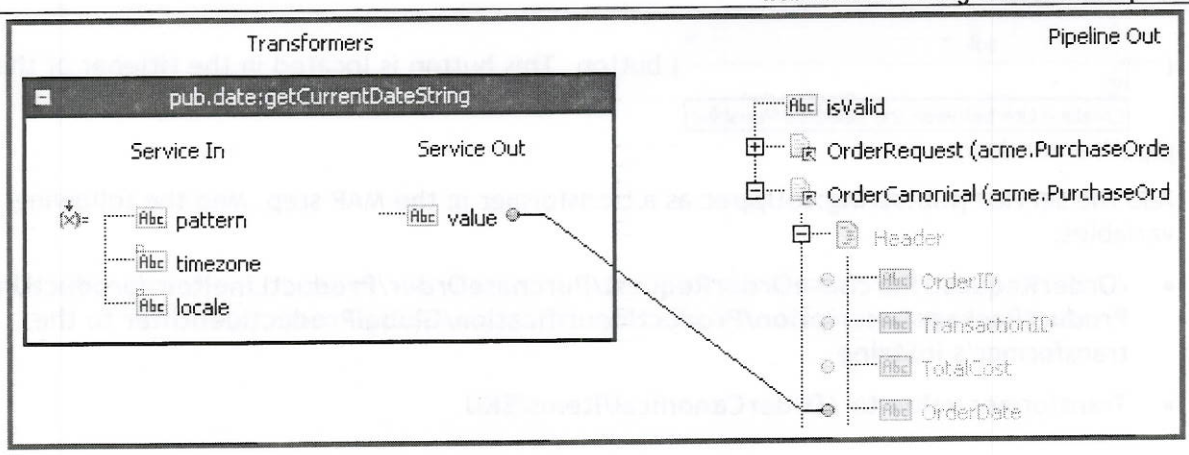



2. Add a **MAP** statement to your service. In the MAP statement, map the following variables from left to right on the Pipeline tab.
   a. OrderRequest/PurchaseOrderRequest/PurchaseOrder/totalCost to OrderCanonical/Header/TotalCost
   b. isValid to OrderCanonical/Header/IsValid
   c. OrderRequest/PurchaseOrderRequest/fromRole/PartnerRoleDescription/DUNS to OrderCanonical/Header/Sender/ID
   d. OrderRequest/PurchaseOrderRequest/toRole/PartnerRoleDescription/DUNS to OrderCanonical/Header/Receiver/ID

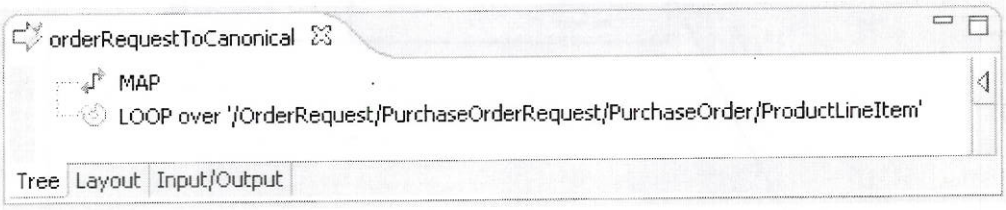
3. Add the service **pub.string:toUpper** as a transformer in your MAP step. Map the following variables from OrderRequest to the transformer and from the transformer to OrderCanonical:
   a. **OrderRequest/PurchaseOrderRequest/thisDocumentIdentifier/ ProprietaryDocumentIdentifer** to the transformer **inString**
   b. The transformer value to **OrderCanonical/Header/OrderID**
   c. The transformer value to **OrderCanonical/Header/TransactionID**
4. Add the service **pub.date:getCurrentDateString** as a transformer in your MAP step. Map the following variables in the transformer and the OrderCanonical document.
   a. Set the transformer **pattern** to **MMMM dd, yyyy**
   b. Map the transformer **value** to **OrderCanonical/Header/OrderDate**

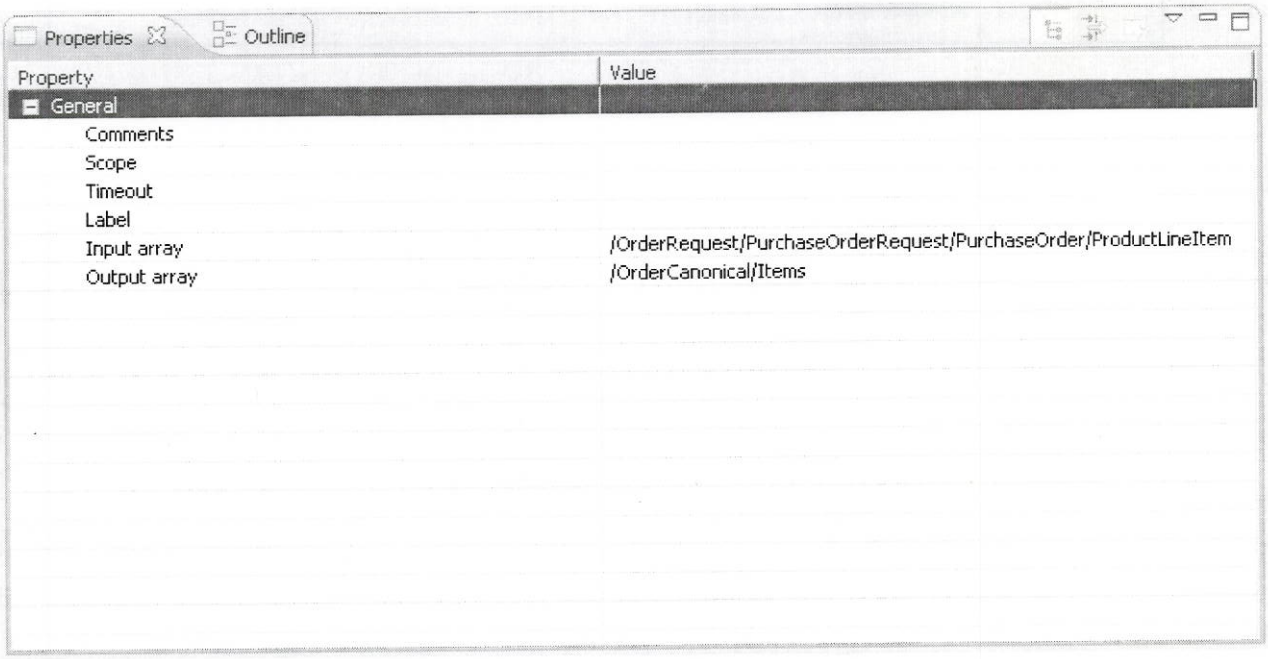


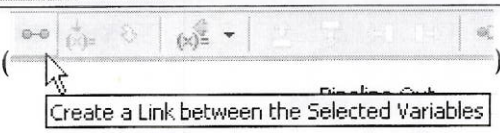



5. Add a **LOOP** step to the service. Set the input array to be **/OrderRequest/PurchaseOrderRequest/PurchaseOrder/ProductLineItem**. Set the output array to be **/OrderCanonical/Items.**



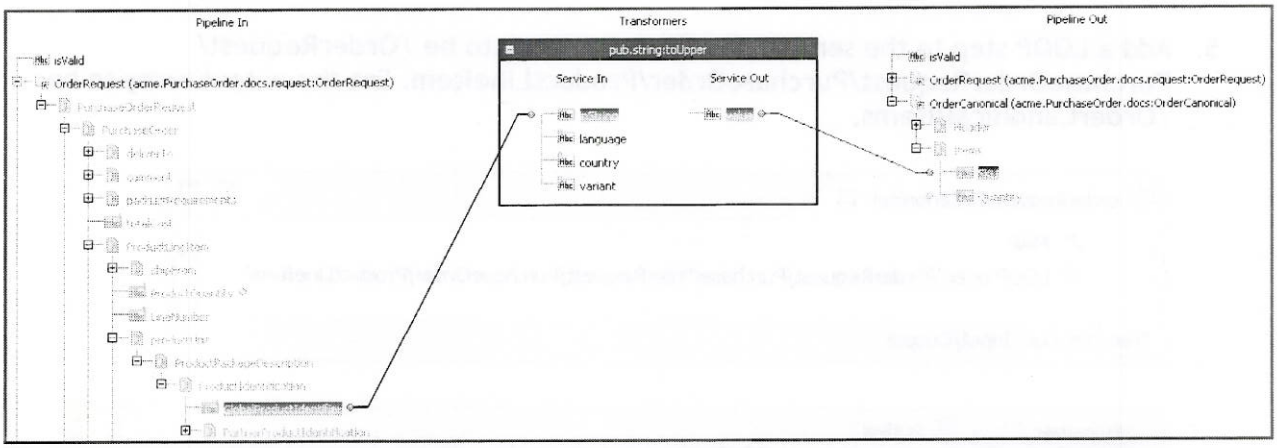




6. Add a **MAP** step in the LOOP (be sure it is indented under the LOOP statement). Map **/OrderRequest/PurchaseOrderRequest/PurchaseOrder/ProductLineItem/ProductQuantity** from the **OrderRequest** variable to the **OrderCanonical**'s variable **/OrderCanonical/Items/Quantity** member.
Hint: When mapping such deeply nested structures, it is often easier to create a map line in two steps. First select the **from** and the **to** fields by clicking them with the mouse. In the second step connect the two selected fields by clicking on the "Create a Link..."

( 

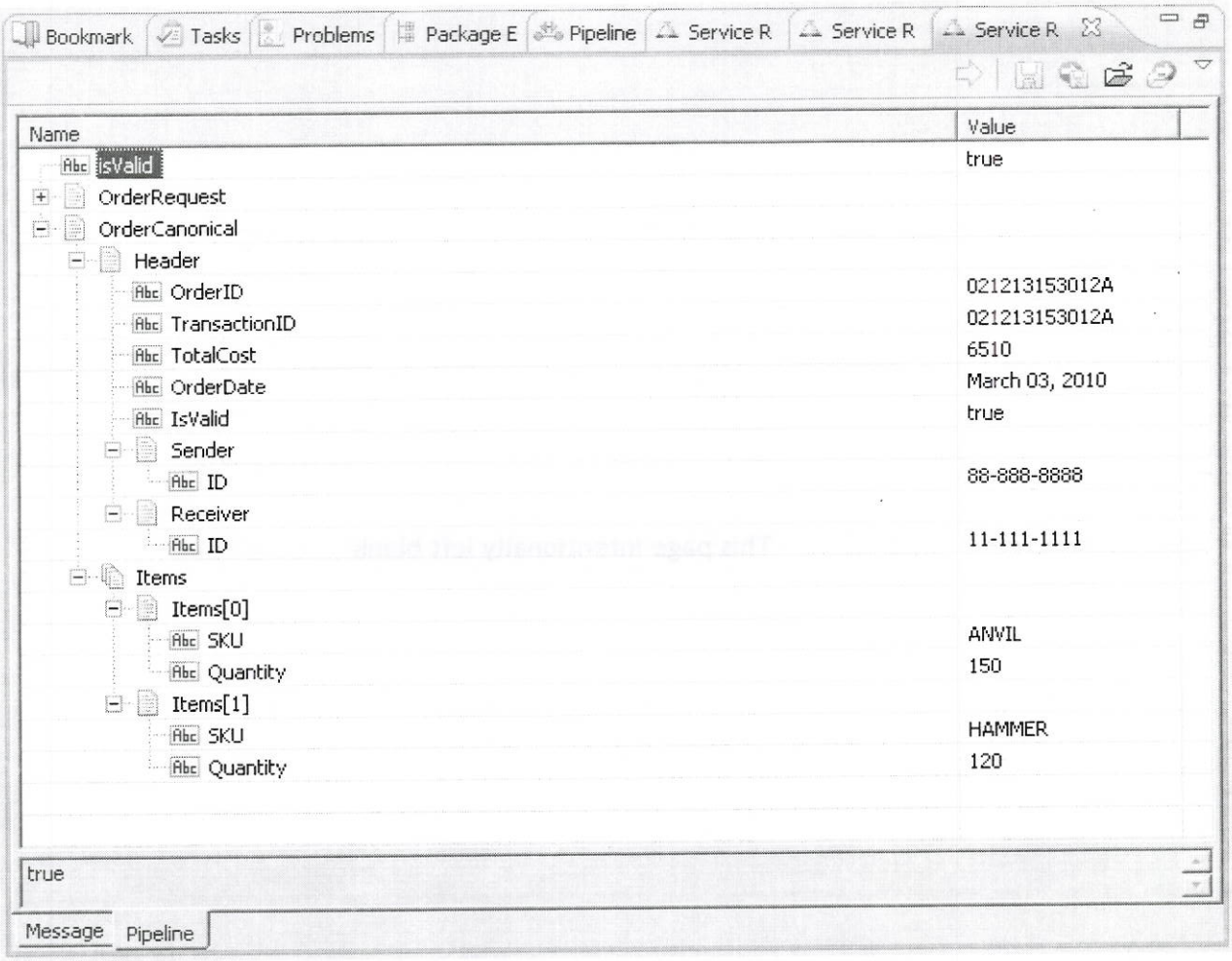
) button. This button is located in the titlebar of the pipeline window.

7. Add the service **pub.string:toUpper** as a transformer in the MAP step. Map the following variables:
   - **/OrderRequest/PurchaseOrderRequest/PurchaseOrder/ProductLineItem/productUnit/ProductPackageDescription/ProductIdentification/GlobalProductIdentifier** to the transformer's **inString**
   - Transformer **value** to **/OrderCanonical/Items/SKU**



8. Save the service and run. Use the Load button and the input file **...\IntegrationServer\packages\AcmeSupport\pub\order_request_input.txt** (Do not forget to set **isValid** to true or false when you test!).

   Check the **Results** panel. Collapse **OrderRequest** and look at **OrderCanonical.** This variable must be completely populated. Especially check the date and the uppercase **OrderID**, **TransactionID**, and **SKU** values.




## Check Your Understanding

1. How is a transformer different from a normal service?
2. What if the transformer you want to use is not in the transformer drop-down list?
3. Why did we need to LOOP over ProductLineItems? Why not just map from ProductLineItems to Items?

This page intentionally left blank

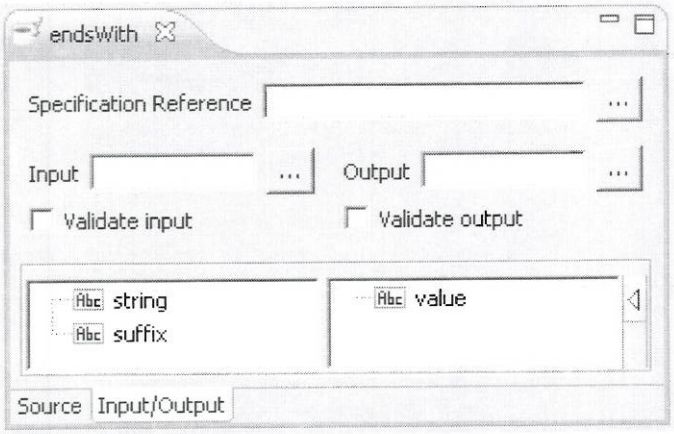
# Exercise 10:
# Create a Java Service

## Overview

In this exercise, you will create, compile, and run a Java Service using Designer. Imagine that you require a special service that tests if a string ends in a second string. There is no such service in the pub.string folder, but you want to use the String.endsWith() method in the Java runtime environment.

## Steps

1. Create a new Java service called **endsWith** in the **acme.PurchaseOrder.work** folder. This service has two string inputs, called **string** and **suffix**.




2. Enter the following code for your service:

```
IDataCursor cursor = pipeline.getCursor();

String string = IDataUtil.getString(cursor, "string");
String suffix = IDataUtil.getString(cursor, "suffix");

String value = string.endsWith(suffix) ? "true" : "false";

IDataUtil.put(cursor, "value", value);

cursor.destroy();
```

   Note: All Java development features, like code completion, that you are used to from the Eclipse IDE are available.

3. Run your service with some sample input values and verify that the returned values are correct.

## Check Your Understanding

1. What exactly is each line of the Java code doing in the endsWith service?
2. Is the service thread safe? What would you have to do if not?
3. How could the cursor handling be improved?