C Unit Testing

C.1 Introduction

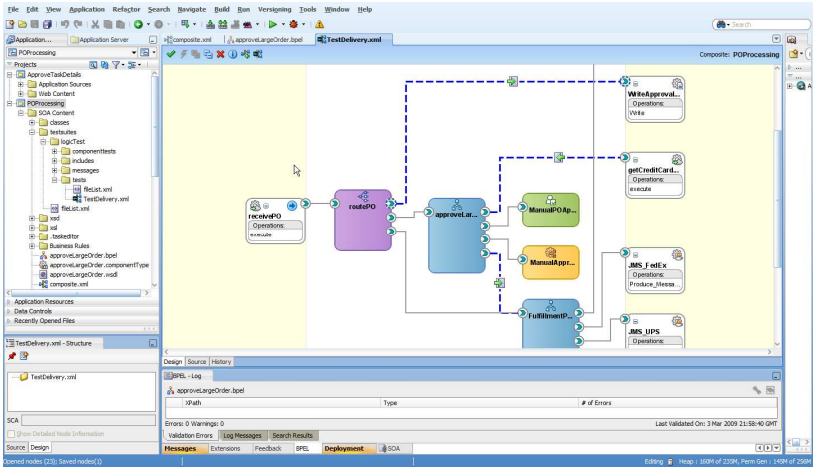
Note: The solution for this chapter can be found in `c:\po\solutions\apC`.
To run this solution, you must have completed labs through chapter 9. Alternatively you can do the setup in chapter 1 and use the solution from chapter 9 located at `c:\po\solutions\ch9`.

The test framework supports testing at the SOA composite level. In this type of testing, wires, binding component services, service components (such as BPEL processes and Oracle Mediator), and binding component references are tested.

In this chapter, you create a Unit Test for the POProcessing composite, including

1. An inbound message for receivePO.
2. A simulation of a callback message returned by a service
3. An assertion to verify the order status at completion
4. An assertion that will always fail for the input we provided in this testcase

When completed, the unit test looks like this in JDeveloper.



C.2 Designing the flow

You will modify POProcessing to add a Unit Test which sets the inbound message, the simulation message, and the two assertions. The inbound message will be a valid order (valid credit card) with order total between 1000 and 5000 so that it triggers the approveLargeOrder process but not the human task (for convenience).

The simulated message is the return from the credit validation service returning the correct value.

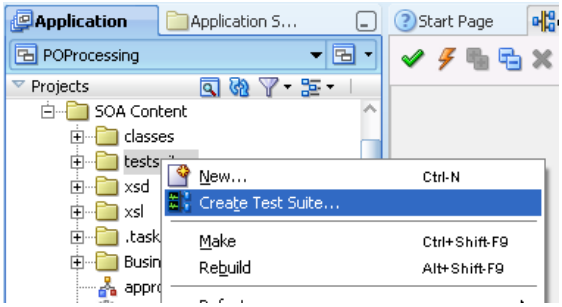
The first assertion is on the data being passed to the WriteFile service, checking that the value of status = 'approved' which is the expected value for this input data.

The second assertion is on the data being passed to FulfillmentProcess, checking that the customer id is 9999 – since the customer id is actually 1111, this test will always fail. This last assertion shows what happens when the data being checked is not the expected value.

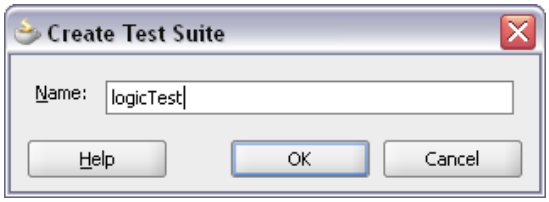
C.3 Create the Unit Test

1. Open the POProcessing project in JDeveloper. In the Application navigator expand the SOA Content folder and right-click on testsuites folder.

2. Select **Create Test Suite**

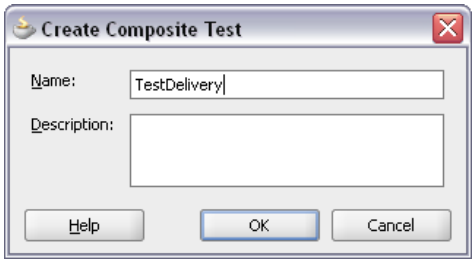


3. Name the test suite *logicTest*



4. Click **OK**

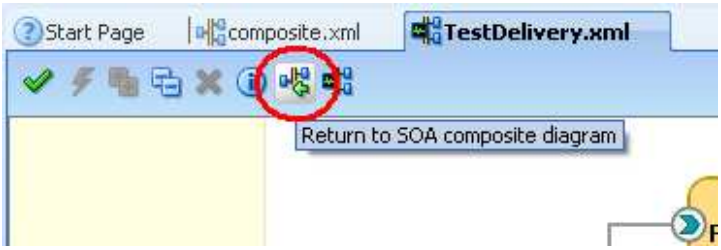
5. Name the test *TestDelivery*



6. Click **OK**

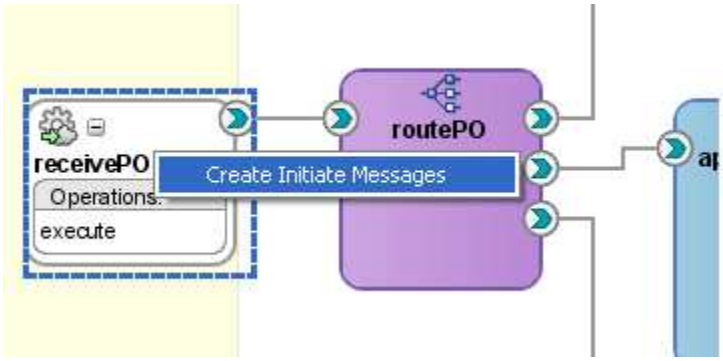
   Notice the composite view changes slightly to show you are in Unit Test creation mode now. The swim lanes on the left and right are yellow.

   Note: You can return to the normal composite editor by selecting the **Return to SOA composite** diagram button at the top of the window.



C.4 Set the inbound message

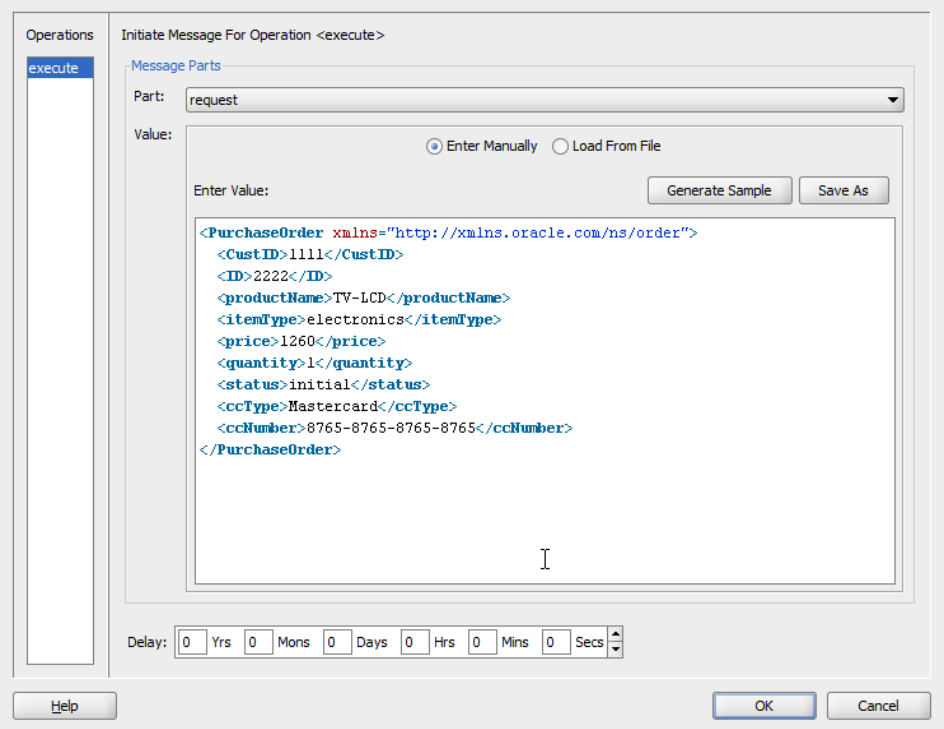
7. First create an inbound message for receivePO. Double-click the binding component receivePO or right-click and select **Create Initiate Messages**



8. In the **Initiate Messages** dialog that appears insert the following payload (copy from file `c:\po\input\po-unittest.txt` :

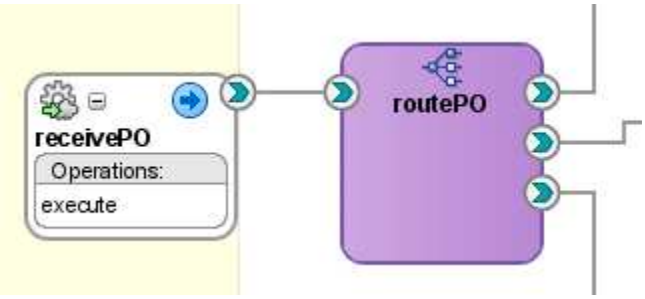
```
<PurchaseOrder xmlns="http://xmlns.oracle.com/ns/order">
  <CustID>1111</CustID>
  <ID>2222</ID>
  <productName>TV-LCD</productName>
  <itemType>electronics</itemType>
  <price>1260</price>
  <quantity>1</quantity>
  <status>initial</status>
  <ccType>Mastercard</ccType>
  <ccNumber>8765-8765-8765-8765</ccNumber>
</PurchaseOrder>
```

   This creates an order for Customer with ID=1111 and a $1260 purchase.



9. Click **OK**.

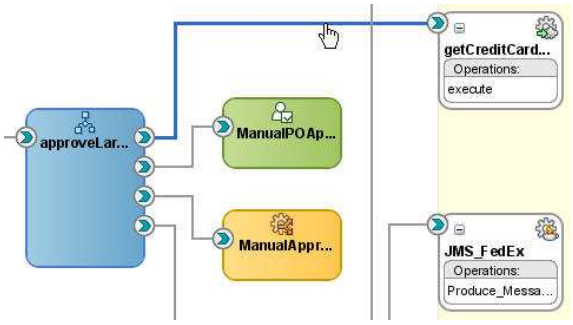
   After closing the dialog you will see a blue arrow on the inbound component indicating there is a message set for that service.



C.5 Set the simulated message

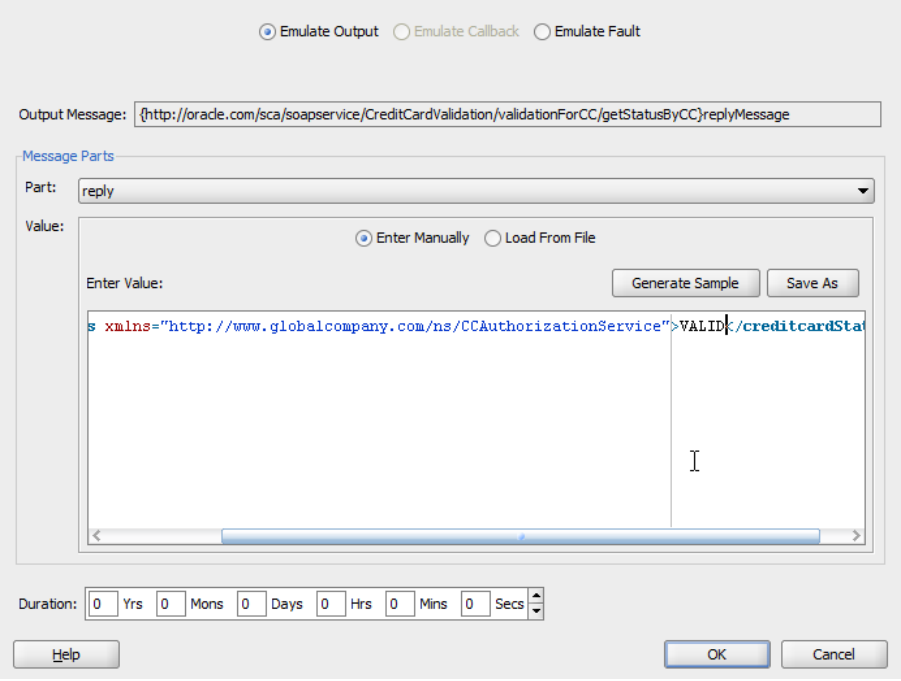
Now simulate a callback message returned from an synchronous web service.

10. Double-click the wire between BPEL process *approveLargeOrder* and the *getCreditCardStatus* Web service.



11. Go to Emulates tab and click on the plus sign

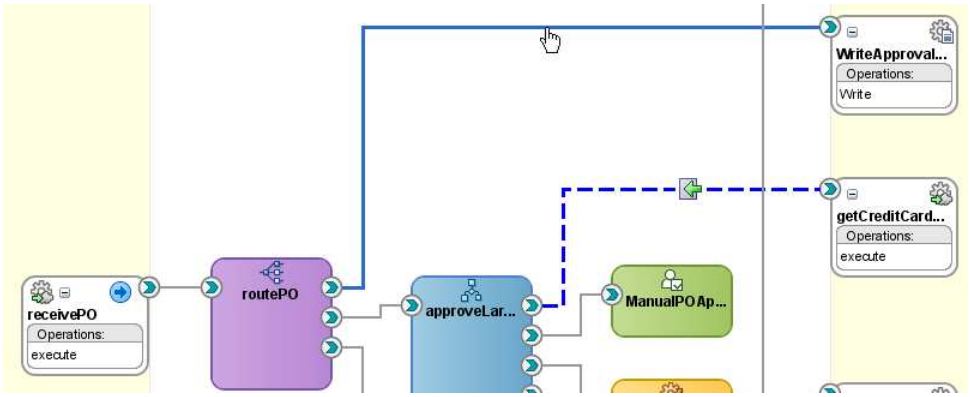
12. Click on Generate Sample to have a sample response automatically created for you and Edit the XML fragment and change the response value to VALID, as if the Web service would return response that the credit card number is valid



13. Click **OK** and click **OK** again.

C.6 Set the assertion for success

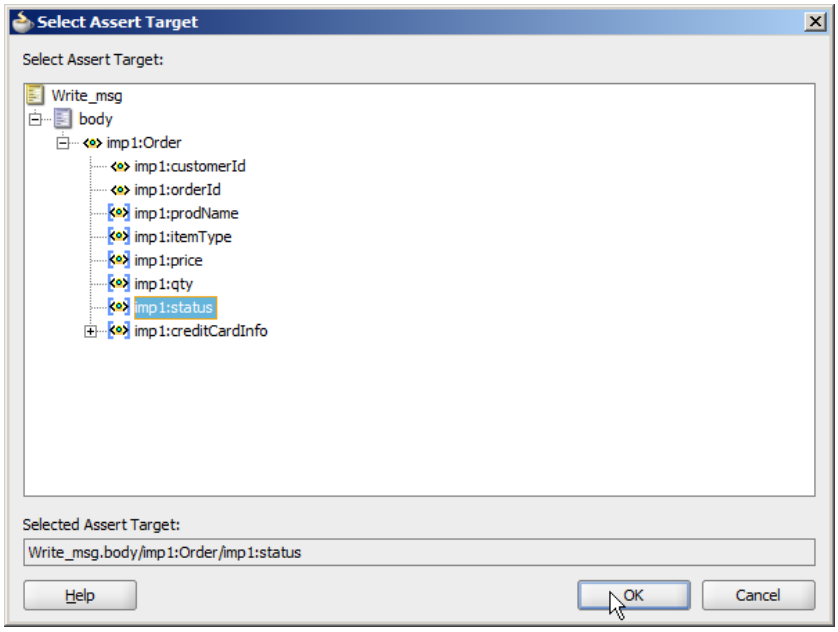
1. Now perform an assertion to verify what order status. Double-click on the wire between routePO and WriteApprovalResults



2. Add an assertion by selecting the green plus sign.

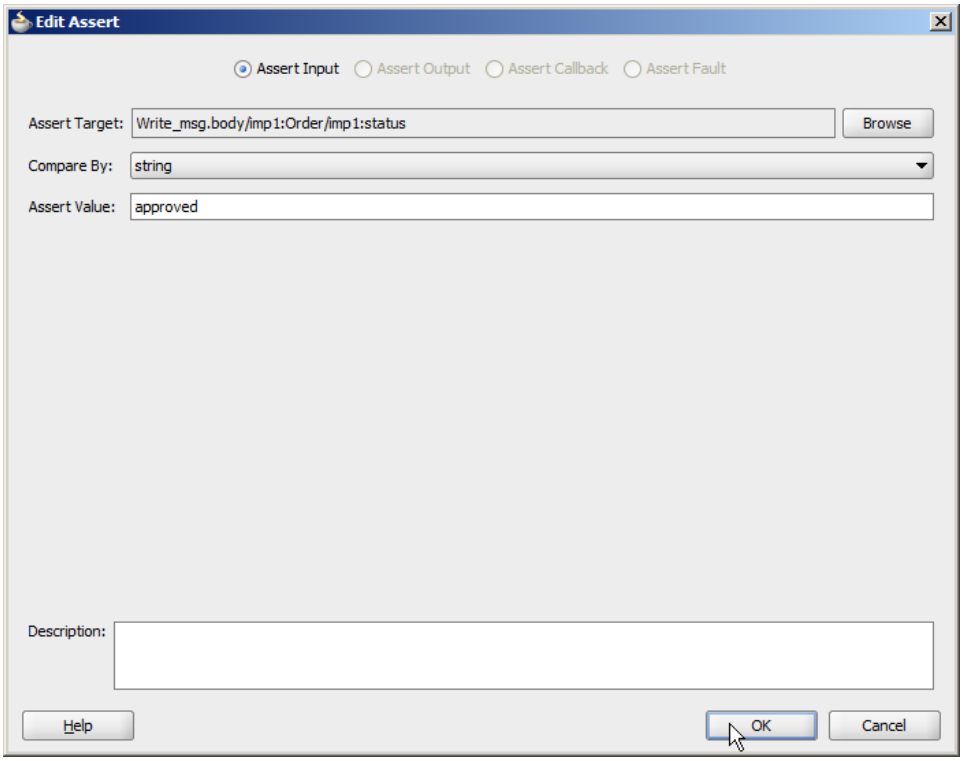
3. To check only part of the message click on Browse button to select only one field of the XML structure:

4. Select only the imp1:status



5. Click **OK**

6. For assert value enter *approved* in order to check the purchase order status

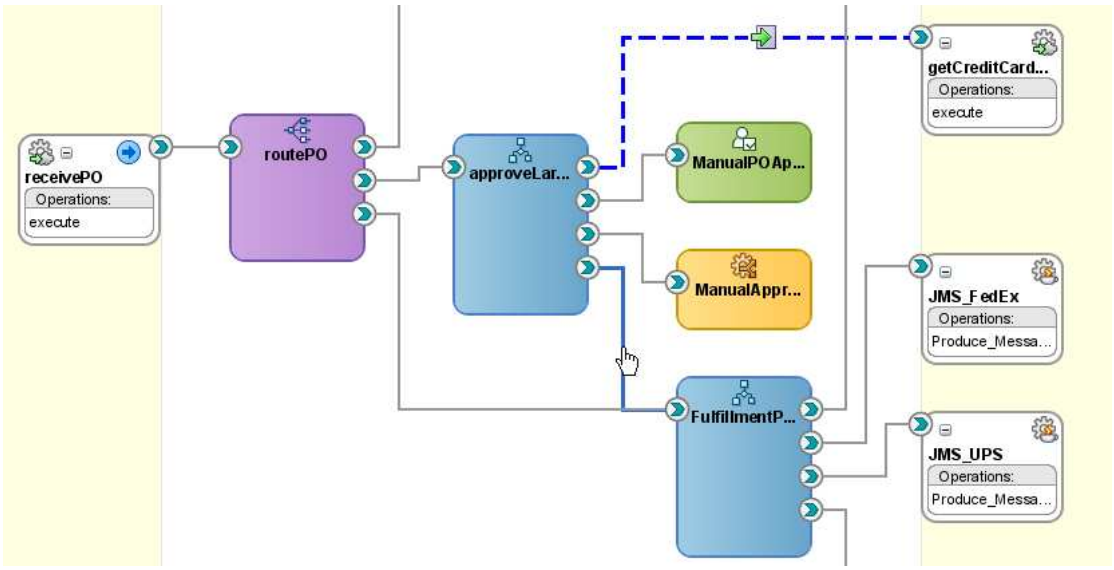


7. Click **OK** and click **OK** again.

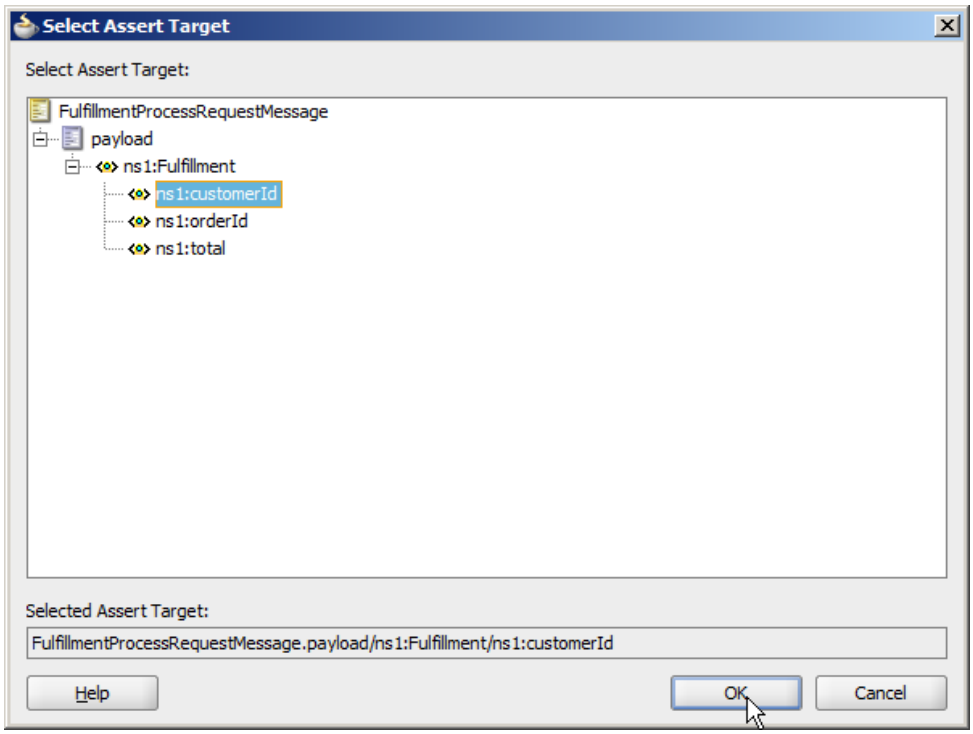
C.7 Set the assertion for failure

Now, you want to create an assertion that will always fail for the input we provided in this testcase.

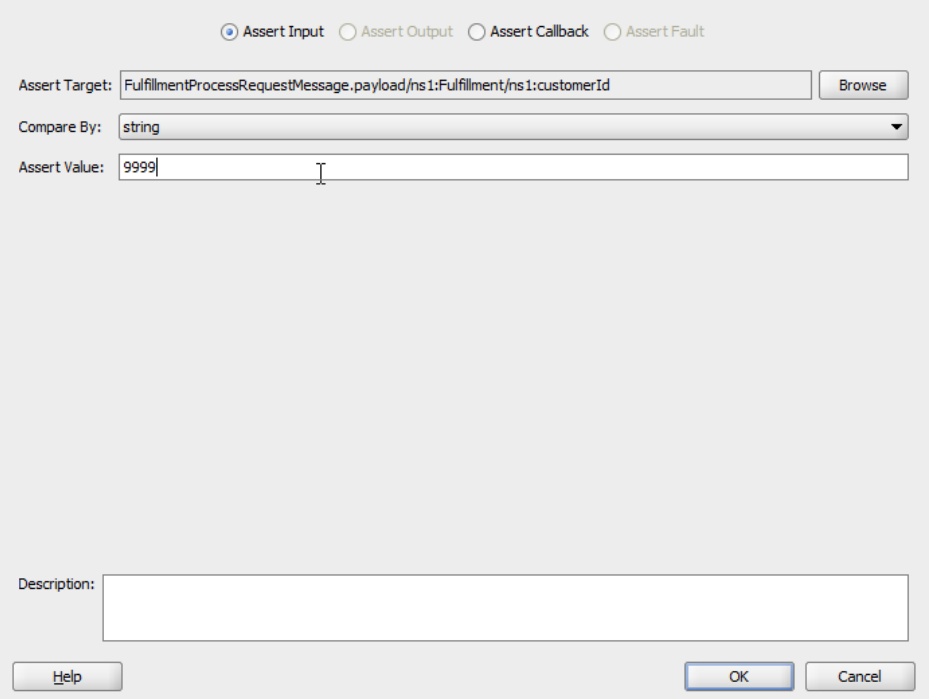
8. Double-click the wire between *approveLargeOrder* and *FulfillementProcess*



9. Add a new assertion and click again on Browse to select only part of the message, and select only **CustomerID**

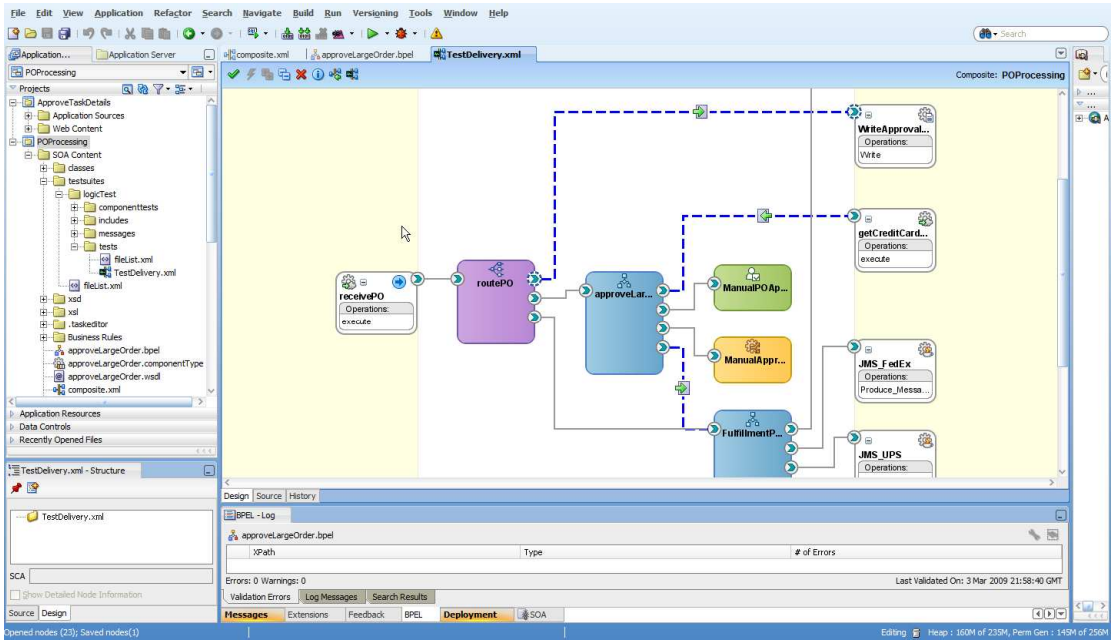


10. For Assert Value enter *9999*. This assertion will fail because the initial payload that we supply was with customerId=1111.



11. Click **OK** and click **OK** again.

The whole Test should look like this :



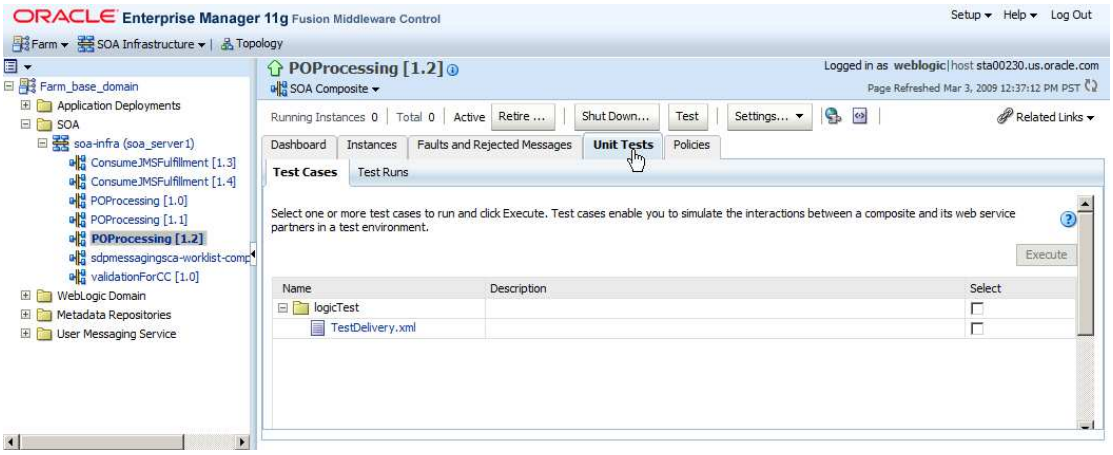
C.8 Deploying the application

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

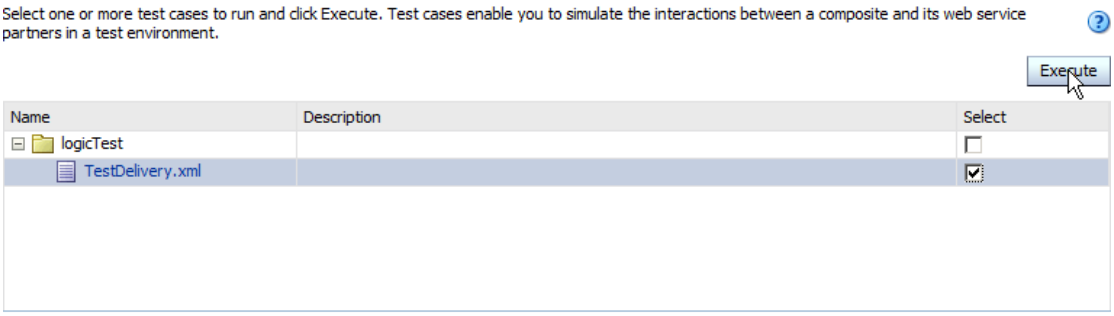
Since you already deployed POProcessing once before, you must either choose a new version number or select **Overwrite any existing composite** when deploying. If you do overwrite the previous composite, the existing instances for the version become stale and can no longer be viewed.

C.9 Testing the application

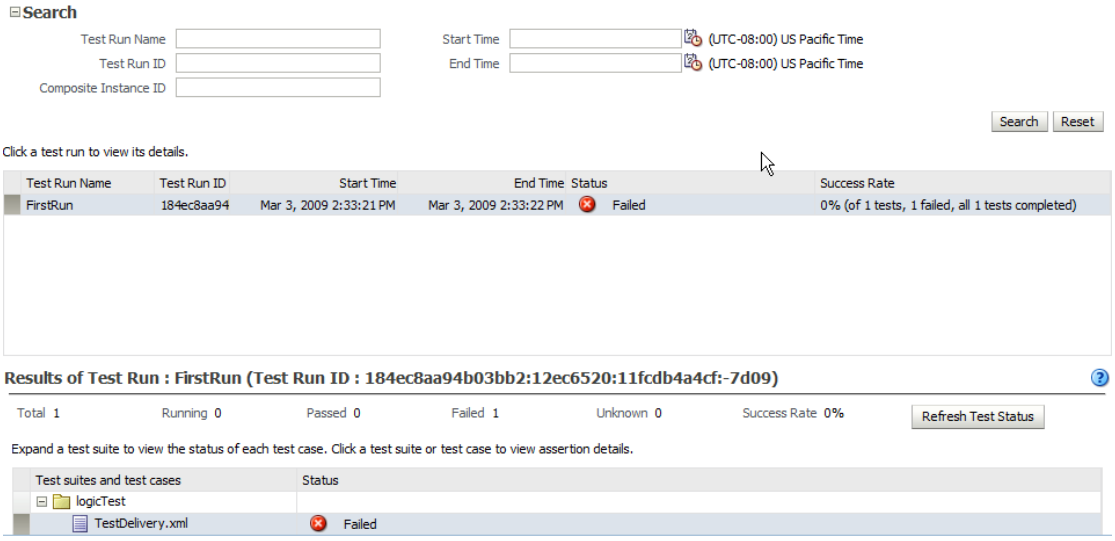
1. After deploying, in the EM console, click on the **POProcessing** application and then open the Unit Test tab.
2. You see your test suite listed.



3. Select TestDelivery.xml and click Execute

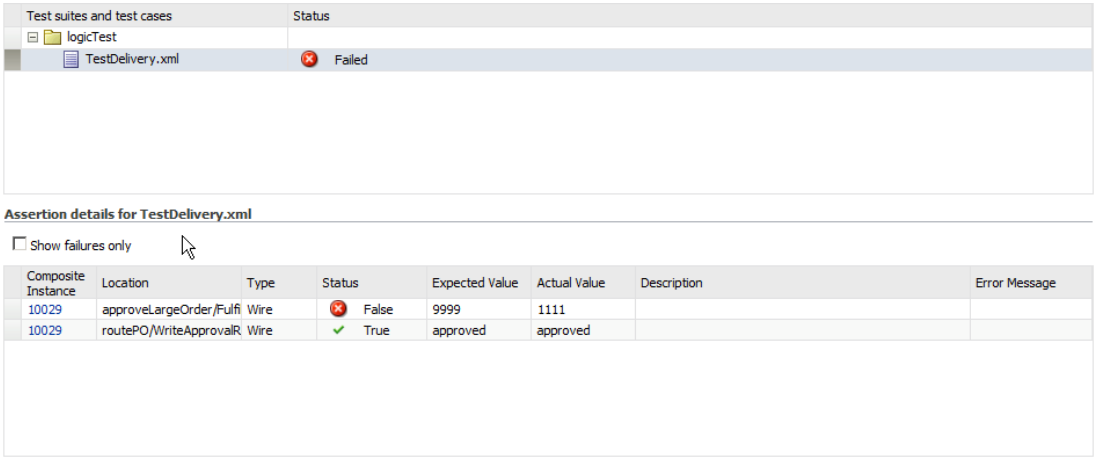


4. After the test has completed you see that the overall status is marked as Failed:



5. Scroll a little down to see the details about the assertions we inserted
6. Note that the first one is Failed as expected because the CustomerID is *1111* and not *9999*

7. The second one is True as the status passed to "WriteApprovalResult" is *approved*



8. Click the Composite Instance ID in the last screen. You see the complete flow of the process. Note that the "getCreditCardStatus" was never called, because of the response Emulation you inserted.