IBM® WebSphere® Process Server V6.1 – Lab exercise

Business state machine: Account manager

What this exercise is about

The objective of this lab is to provide you with an understanding of the Business State Machine and how to develop an application using the state machine as the primary controller which drives other business processes.

Lab requirements

List of system and software required for the student to complete the lab.

- WebSphere Integration Developer V6 installed
- WebSphere Process Server V6 test environment installed
- Sample code in the directory c:\LabFiles602 (Windows®) or /tmp/LabFiles602 (Linux®)
- You should already be familiar with the Component Test Client.
- You should know how to wire up a module assembly, with Imports and Exports.

What you should be able to do

At the end of this lab you should be able to:

- Create and test a Business State Machine which collaborates with a BPEL business process using *Actions* on the transition
- Add *Conditions* to the Business State Machine to apply business restrictions to the overall business process flow
- Use the WebSphere Test Environment to move through the states of the Business State Machine.
- Programmatically send events/operations to the Business State Machine from a long running BPEL business process

Introduction

This application is for the life cycle management of a customer account.

The sales team for 'MyCompany' will identify a new prospect and review the account to see what business opportunities are available. If, as a result of the review, the sales team concludes that this is a good candidate, then the sales team will work with the customer to complete the application. Once the application for a new account is completed, it is verified for correctness and accuracy. Having been successfully verified, the account can then be activated.

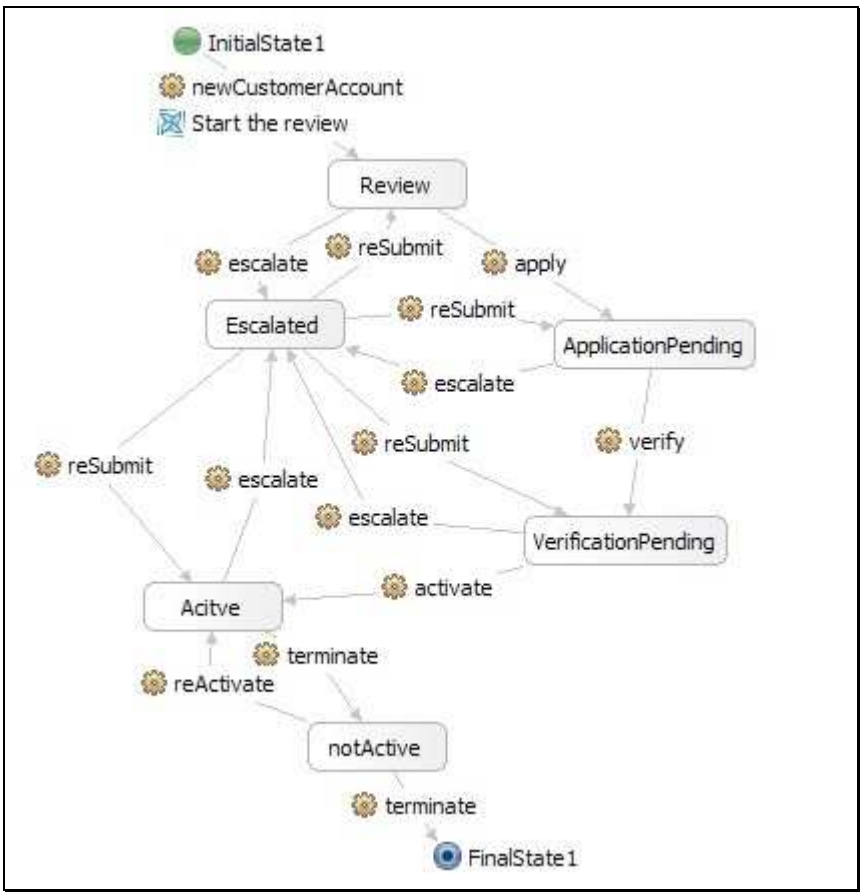
At any point in the business process the flow may be escalated to achieve greater focus and urgency or sent back to the previous step for rework.

The underlying business processes such as review, apply, verify and escalate are implemented as long running business processes that normally involve some kind of human interaction. (***You will not implement the human interactions at this time, since it is not the main focus of the scenario***.)

Additionally there will be Conditions/Guards on some of the transitions to insure that the criteria for moving from one step to the next have been met. You will implement the conditions using Java™ snippets but you can see how this would be a natural place to insert business rules.

The secondary, underlying business processes such as review, verify, and others, will send the events to the business state machine either upon completion or escalation of the process. In the diagram below, when the state machine is started with the ***newCustomer*** operation, the action labeled, *Start the Review*, invokes the Review business process and the state machine will remain in the *Review* state until either the *escalate* or the *apply* operations are invoked.

This diagram is not complete yet. To make this state machine behave properly, you must add additional *Actions* to invoke the other business processes and then some conditions. Notice that the Escalate state has many possible resubmit transitions. If you leave it like this the Business State Machine runtime will not be able to determine which transition to use and the editor will flag this as an error. This ambiguity is resolved by using conditions based on the input of the escalate operation.



Managing a customer account can be a complex business process. The customer account has a life cycle that must me carefully managed. There are typically several phases, each of which can be defined as a BPEL business process. In this exercise you will see how the business state machine can be used to apply the overall top-level control structure to the various phases of the account management life cycle.

The BPEL business processes, which will be long running, will be invoked using the actions on the transitions of the business state machine (BSM). Additionally, conditions will be added to the transitions to apply business rules, either Conditions or real Business Rules.

A few words on the structure of the application:

There are two SCA Modules and one Library Module.

1. AccountManagerBSM
2. AccountManagementProcesses
3. AccountManLib

The Library Module contains the interfaces and the schema definitions used by the other two modules.

Exercise instructions

Some instructions in this lab might be specific for Windows platforms. If you run the lab on a platform other than Windows, you will need to run the appropriate commands, and use appropriate files (for example .sh in place of .bat) for your operating system. The directory locations are specified in the lab instructions using symbolic references as follows:

Reference variable	Windows location	AIX®/UNIX® location
<WPS_HOME>	C:\IBM\WID61\runtimes\bi_v61	
<WPS_PROFILE_HOME>	C:\IBM\WID61\pf\wps	
<LAB_FILES>	C:\LabFiles61	

Windows users: When directory locations are passed as parameters to a Java program such as EJBdeploy or wsadmin, you must replace the backslashes with forward slashes to follow the Java convention. For example, replace C:\LabFiles61\ with C:/LabFiles61

Instructions for using a remote testing environment, such as z/OS®, AIX or Solaris, can be found at the end of this document, in the section "**Task: Adding remote server to WebSphere Integration Developer test environment**".

Part 1: Getting started

____ 1. Follow the directions below to initialize the Workspace using the following values:

<WORKSPACE>

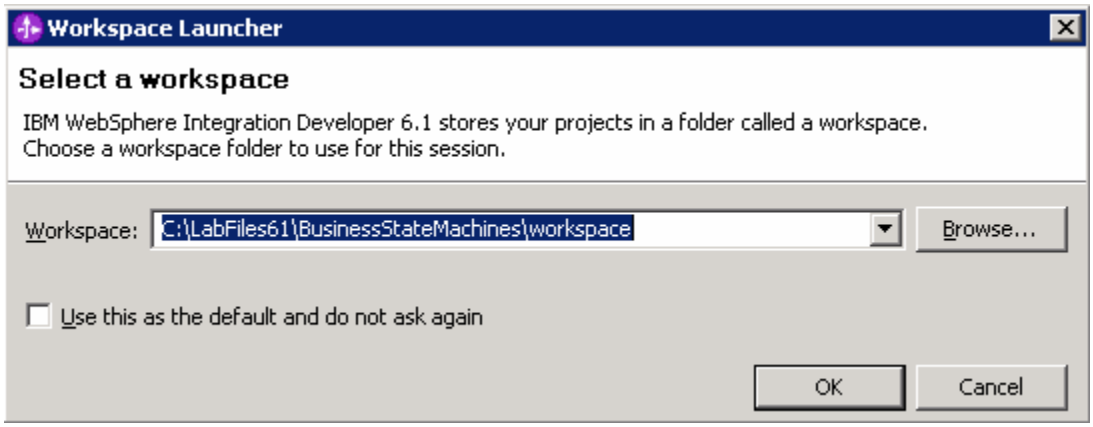
C:\LabFiles61\BusinessStateMachines\workspace

<PROJECT_INTERCHANGE>

C:\LabFiles61\BusinessStateMachines\Snippets\AccountManager.beta.pi.zip

____ 2. Start the WebSphere Integration Developer and setup the environment

__ a. When prompted for workspace name, enter the value provided by the **<WORKSPACE>** variable for this lab and click **OK**



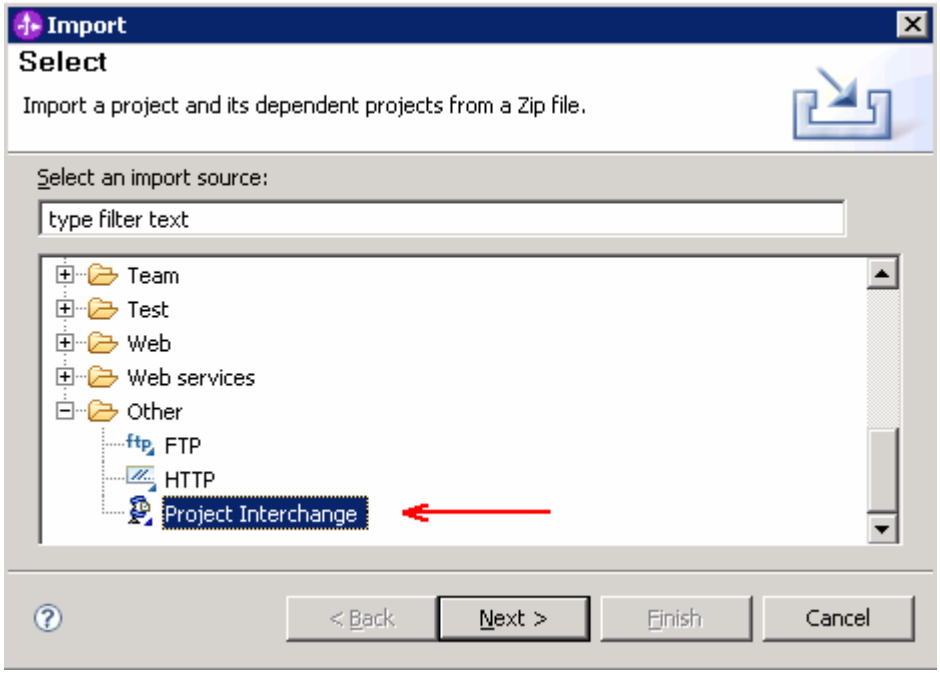
__ b. Close the Welcome tab

__ c. Ensure you are in the **Business Integration** perspective.

____ 3. If this lab requires you to import a project interchange file, setup the required libraries and modules for this lab by importing the project interchange file **<PROJECT_INTERCHANGE>**

__ a. From the file menu, select **File → Import…**

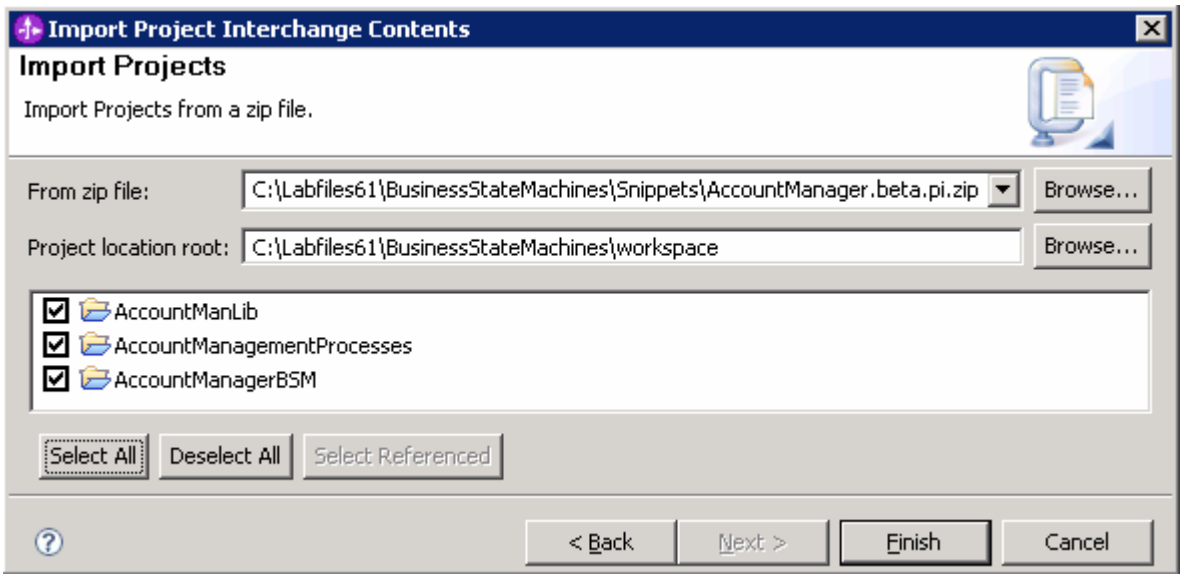
__ b. In the Import dialog, scroll down, expand **Other** and select **Project Interchange**



__ c. Click **Next**

__ d. In the '**Import Projects**' panel, click the **Browse** button and select the project interchange file located at **<PROJECT_INTERCHANGE>**

__ e. Click the '**Select All** button

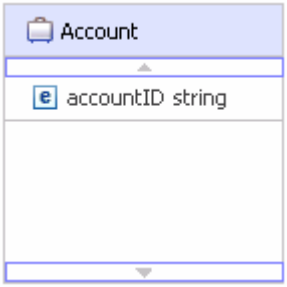


__ f. Click **Finish**

____ 4. Inspect the data objects in the '**AccountManLib**' library

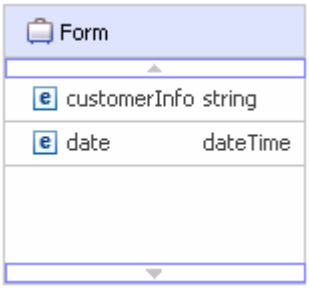
__ a. In the project explorer, expand to '**AccountManLib → Data Types**'

__ b. Right click on '**Account**' and select **Open** to inspect the Business Object. The variable '**accountID**' is used as the correlation ID throughout the application



__ c. Repeat step '**b**' to inspect '**Customer**' business object

__ d. Repeat step '**b**' to inspect '**Form**' business object



1) In the **Form** business object, '**customerInfo**' is used to store any extra information about the customer and '**date**' is used to control execution logic added in '**Part 3: Add Condition**'

____ 5. Inspect the Interfaces in the '**AccountManLib**' library

__ a. In the project explorer, expand to '**AccountManLib → Interfaces**'

__ b. The '**AccountManagerInterface**' defines the operations on the business state machine. To cause a transition from one state to another, invoke one of these methods on an instance of the business state machine. You will use the component test environment to do this for most of the work you'll be doing

1) Note that all the operations will need to correlate. The variable '**accountID**' in the **Account** business object is used throughout this exercise for this purpose

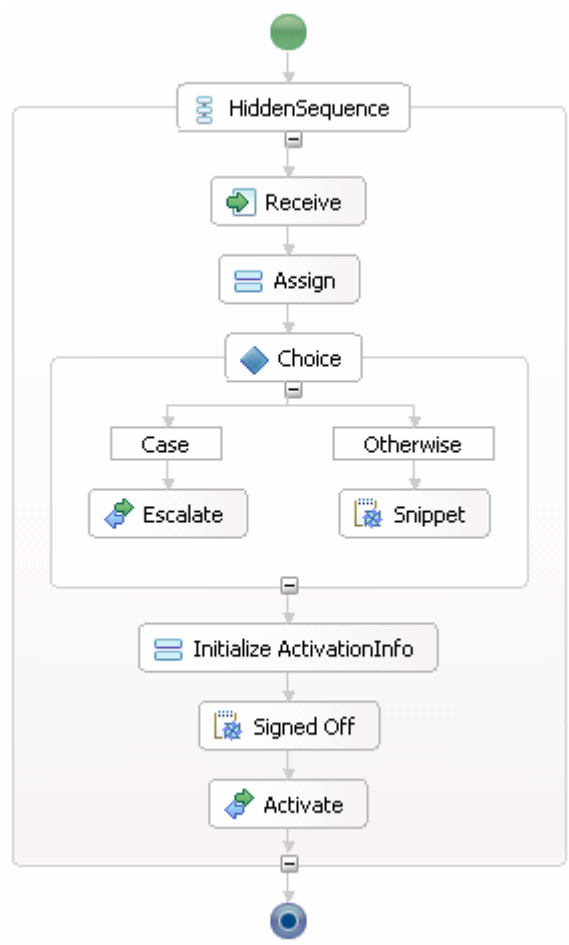
__ c. All of the interfaces with Process in the name are used for the BPEL business processes which can be invoked by the *Actions* of the Account Management business state machine. Their interfaces are all the same for this exercise

- ActivateProcessInterface
- ApplicationProcessInterface
- ReviewProcesssInterface
- VerificationProcessInterface

____ 6. Inspect the business process in the '**AccountManagementProcess**' module.

__ a. In the project explorer, expand to '**AccountManagementProcess → Business Logic → Processes**'

__ b. Double click on '**ReviewProcess**' to open it in the Business Process Editor



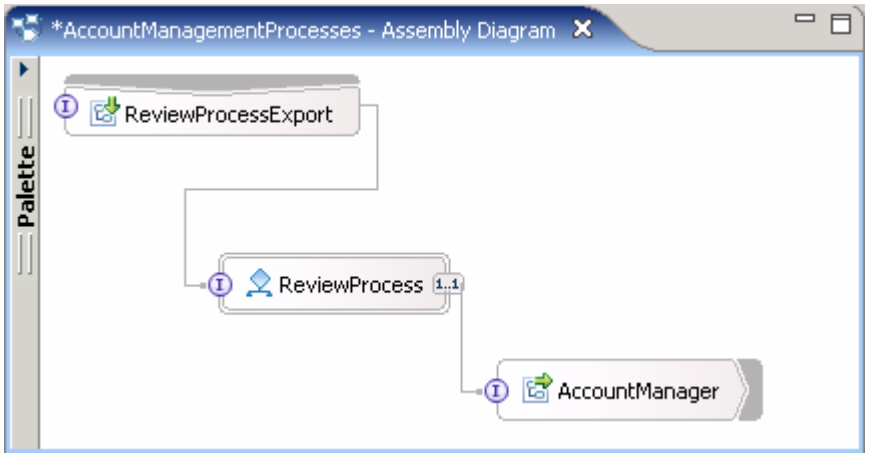
1) ReviewProcess is a simple and incomplete business process at this time. There will eventually be one BPEL business process per state. It is not a requirement to implement an Action as a business process; these instructions use a long running business process to illustrate what can be done and to demonstrate how the different technologies can be used together

2) The ReviewProcess (BPEL) will check the name of the customer and if the name is on the '***preferred customer'*** list the business process will be escalated to ensure that it receives the proper attention

____ 7. Inspect the module assemblies for '**AccountManagementProcess**' and '**AccountManagerBSM**'

__ a. In the project explorer, expand **AccountManagementProcess**

__ b. Double-click on Assembly Diagram (Assembly Diagram) to open the Assembly Diagram

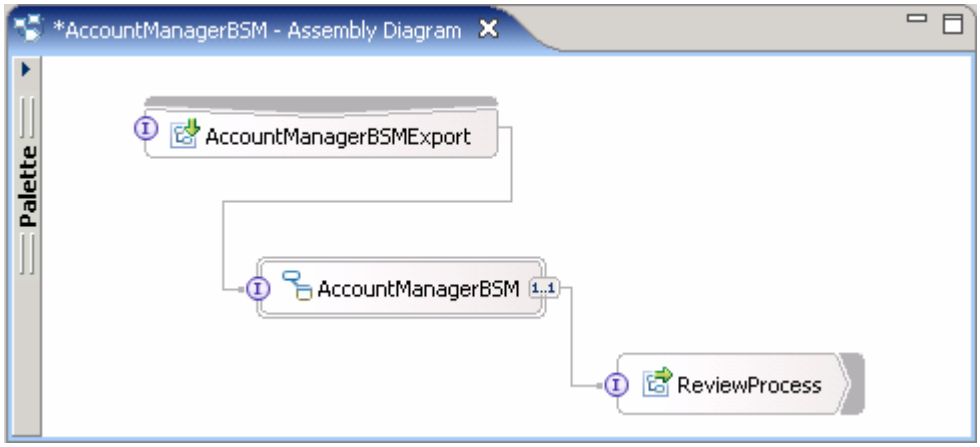


1) There is nothing special except the simplicity

2) AccountManagementProcess uses SCA bindings

__ c. In the project explorer, expand '**AccountManagerBSM**'

__ d. Double-click on Assembly Diagram (Assembly Diagram) to open the Assembly Diagram



Part 2: Make a test run

At this point there is enough of the state machine constructed so you can learn how to run it manually using the Component Tester. This is what you will do in this section.

The Component Test Client will allow you to invoke any of the operations that have been exported from the '**AccountManagerBSM**' module. You must have a separate invoke for each one. For this test run you will setup three invokes initially. This will make it easier to step through the operations. As you move from state to state and invoke BPEL business processes, there will be messages in the console to help you see the transitions and actions.

The first operation you will invoke will be the '***newCustomer***' operation, which will move the business state machine from the initial state to the *Review* state, and invoke the '***ReviewProcess***' as an action.

The '***activate***' operation will cause the transition to the *Active* state if the *WaitFor* Condition has been met. The *WaitFor* condition is a simple check against a timestamp to allow for a cooling off period before actually activating the account. (Initially this is an automatic transition that will be called by the ReviewProcess business process.) Of course this will be too soon for the condition so the state machine will remain in the *Review* state until it times out or you invoke the *activate* operation again after the designated cool off period.

The next operation will be '***terminate***', which will cause the transition to the '***nonActive***' state. The state machine will remain in this state until the account has been '***reActivated***' or the '***TermialTimeout***' has been reached.

The next operation will again be '***terminate***', but this time the current state is *notActive* so the transition will be to terminate the state machine, '***End***'.

Because of the way the timeouts are setup, if you do nothing after the initial invocation of newCustomer the business state machine will terminate in about 5 to 10 minutes.

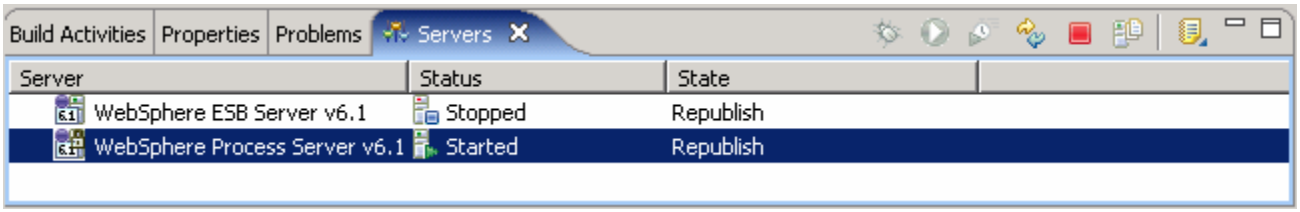
If you are interested in the '***Escalate***' state, take a look at the '***ReviewProcess***'

____ 1. Start the test server

__ a. Change to the Servers view

Note: If using a remote testing environment, follow the directions provided in **Task: Adding remote server to test environment** at the end of this document to add a server to the test environment and start it. This is especially true for z/OS, AIX®, Solaris remote test environment, where the WebSphere Integration Developer will be remote to the test environment.

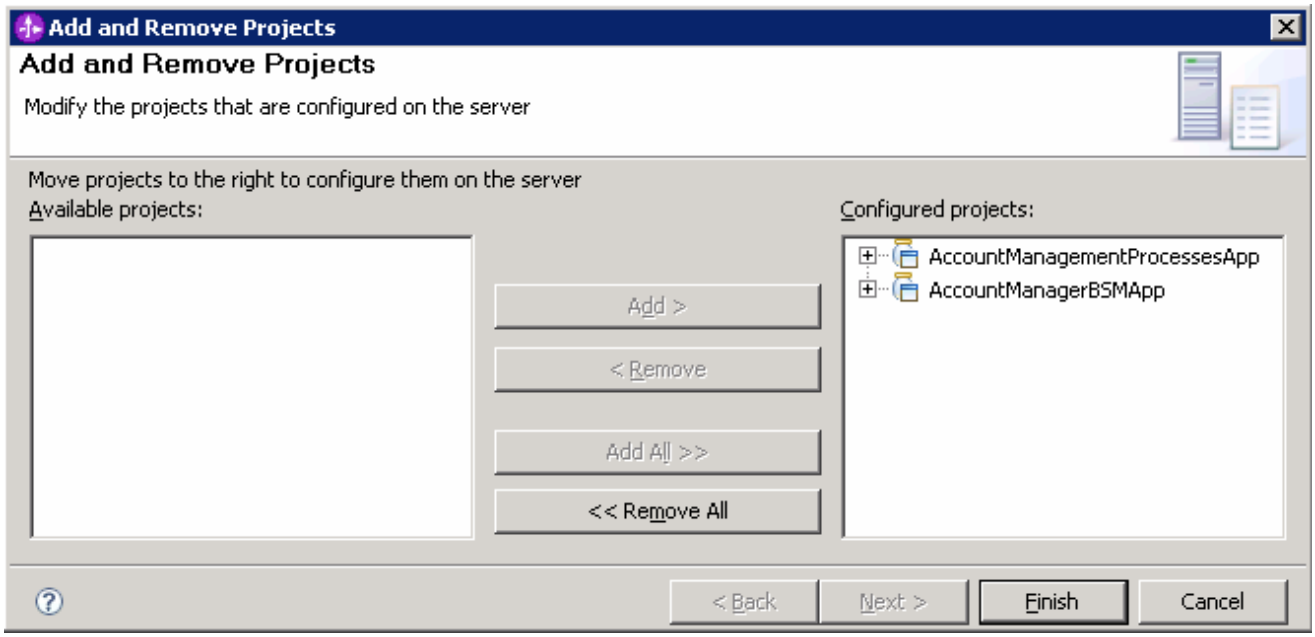
If using a local testing environment, right-click on **WebSphere Process Server v6.1** and select **Start** from the context menu



____ 2. Add both applications to the server when the server has started

__ a. Right-click on the server and select '**Add and Remove projects**' from the pop-up menu

__ b. Click the '**Add All >>**' button to add both applications to the server



__ c. Click **Finish**

____ 3. Start the Component Tester on the '**AccountManagerBSM**' module

__ a. In the project explorer, right click on '**AccountManagerBSM**' module and select '**Test → Test Module**' from the pop-up menu

---

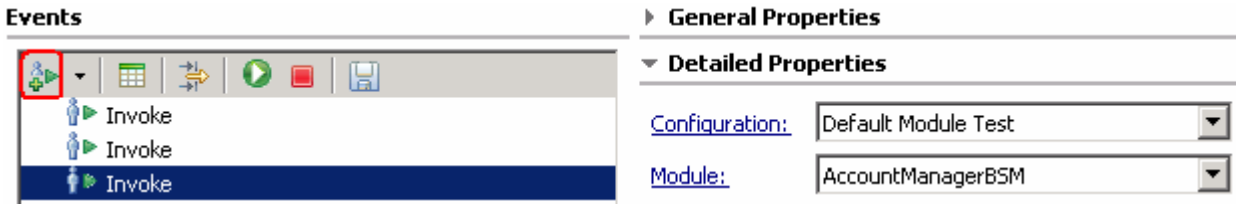
Note: If using a remote testing environment, select **Project → Properties → Integration Test Client**. Unselect the check box for '**Always use the default target in the test client**' if selected and click **OK**. You'll be prompted for your deploy location now when testing.

____ 4. Run the tests

Note: Remember that the '**accountID**', **act1234**, is used to correlate the entire system. Enter **act1234** for accountID.

__ a. Add two invokes

1) In the '**Events**' section, click the **Invoke** button two times; there should now be three invokes listed as shown below:



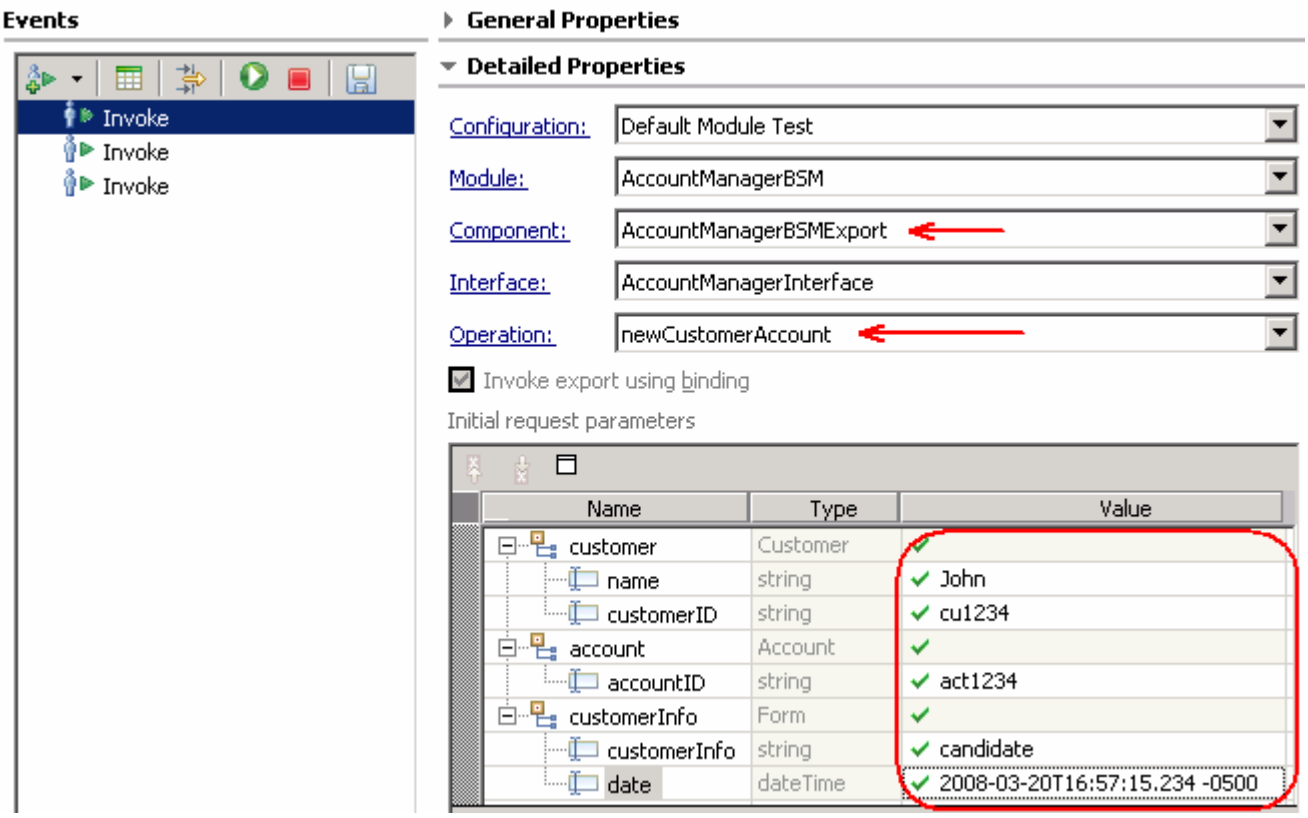
__ b. Highlight the Invoke 1 and select '**AccountManagerBSMExport**' for **Component** under the 'Detailed Properties' section

__ c. Repeat the step '**b**' for Invoke 2 and Invoke 3

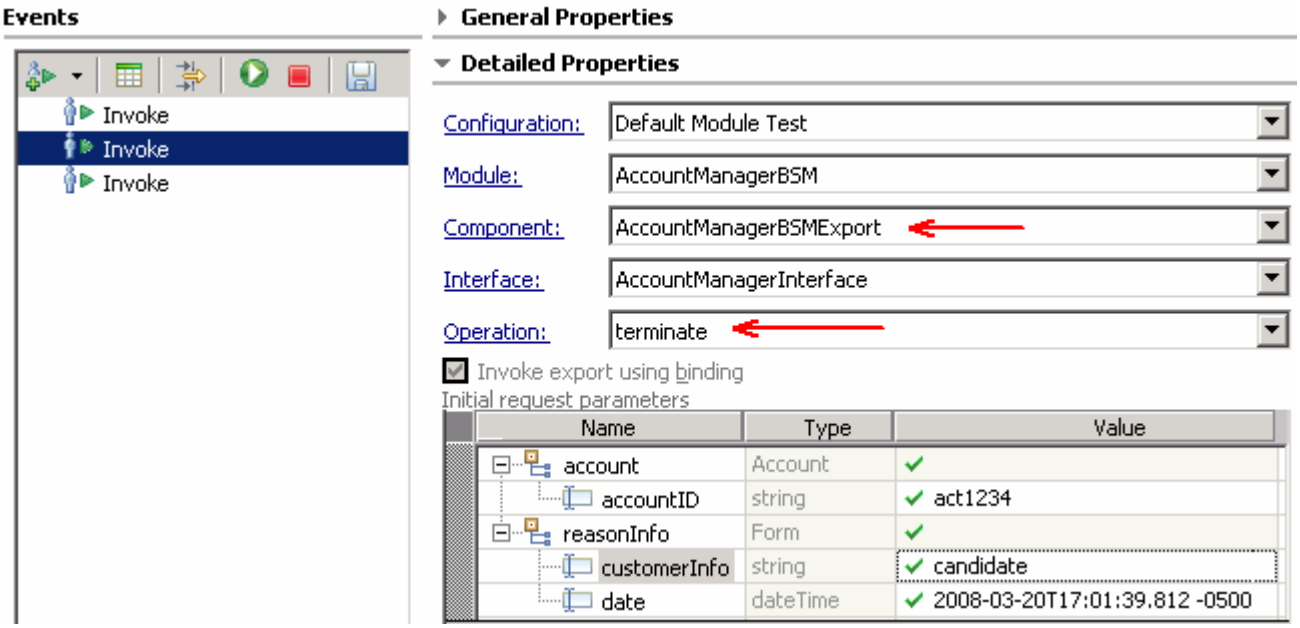
__ d. Set the input variables

1) For Invoke 1, set the **Operation** to '**newCustomerAccount**'

a) Enter your own values or use the values below for each variable



2) For the second invoke set the **Operation** to '**terminate**' and set the **accountID** to match that of the '**newCustomerAccountInvoke**'



__ e. Select the first invoke and press **Continue** ( ) to invoke the operation

__ f. Select '**WebSphere Process Server V6.1**' from the '**Deployment Location**' dialog and click '**Finish**'

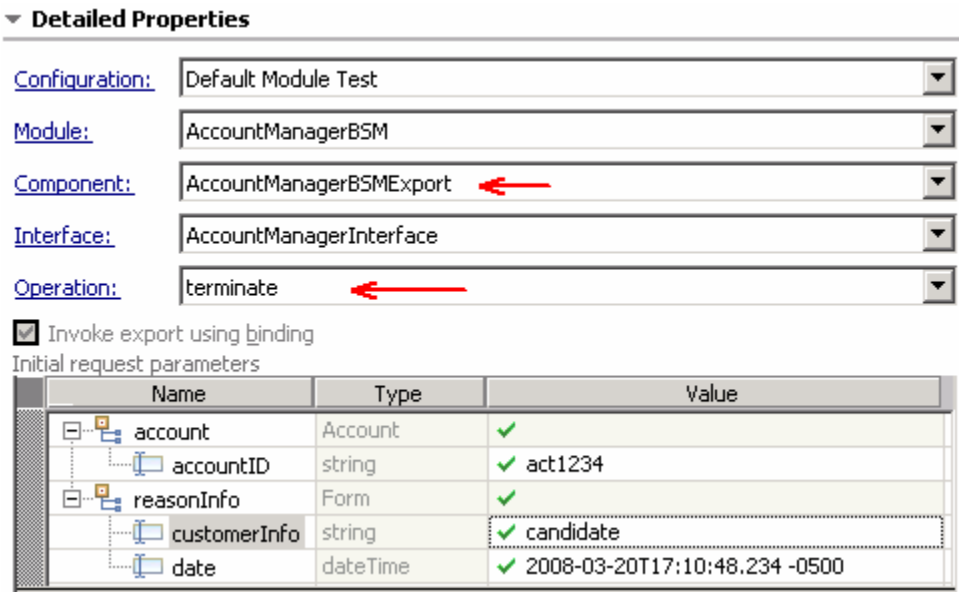
Note: Looking at the console messages you will see that the **ReviewProcess** business process was kicked off and at the end it sends the *activate* event to the business state machine, which moves it to the ***Review*** state.

__ g. Step through the state machine using the other two invokes

1) Select second invoke and press **Continue** ( ) to invoke a '**terminate**' operation

The next state will be the **notActive** state by way of the '**terminate**' event or operation

2) For the final invoke, set the operation to '**terminate**' and select **Continue** ( )



To leave the **notActive** state, you can '**terminate**' or '**reActivate**' using the remaining invoke or wait for it to time out

3) When finished, close the test window without saving changes

____ 5. Remove all applications from the server

__ a. Change to the servers view, right-click on '**WebSphere Process Server v6.1**' and select '**Add and Remove projects**' from the context menu

__ b. Click the '**<< Remove All**' button to remove both projects from the server

__ c. Click **Finish**

Part 3: Add a condition

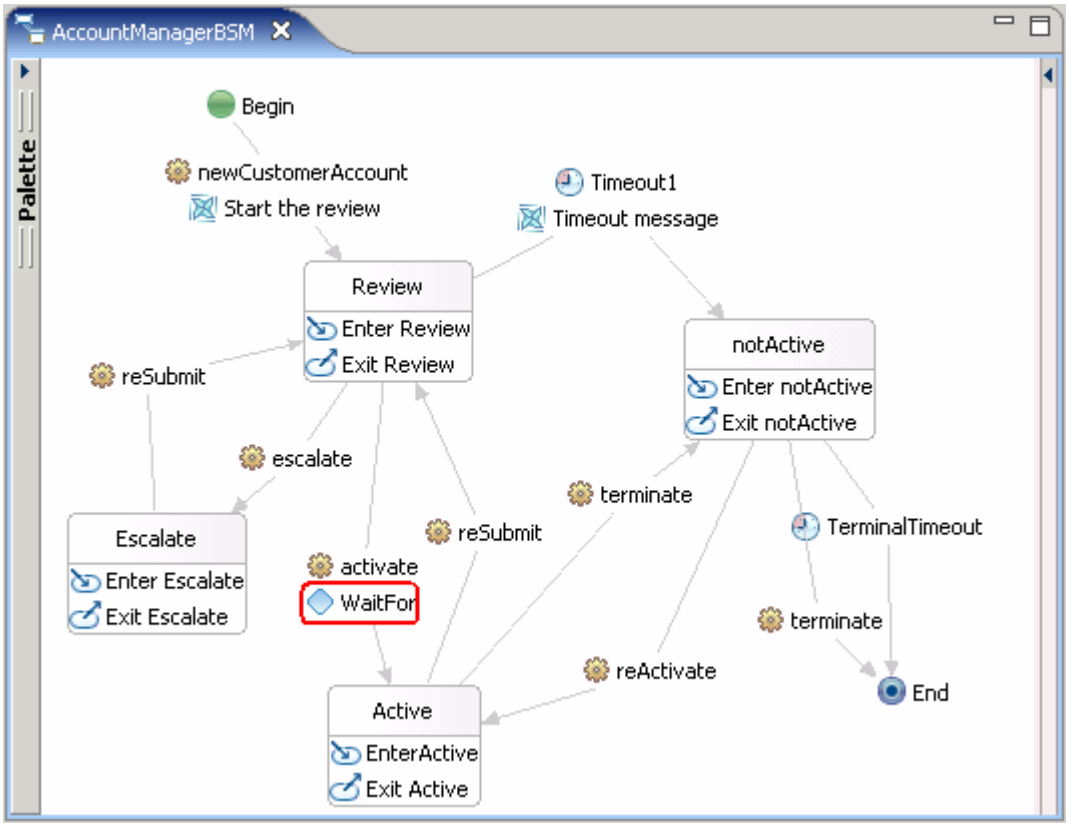
In this part of the exercise you will add a condition to the existing state machine. You might have noticed there is already a condition on the transition from the Review state to the Active state. If you look at the code associated with the condition, you will see that is really just a passthrough. You will add some logic to make it perform checking to ensure that the new account is not activated prematurely.

____ 1. In the project explorer, expand to '**AccountManagerBSM → Business Logic → State Machines**'

__ a. Double-click on **AccountManagerBSM** to open the Business State Machine editor

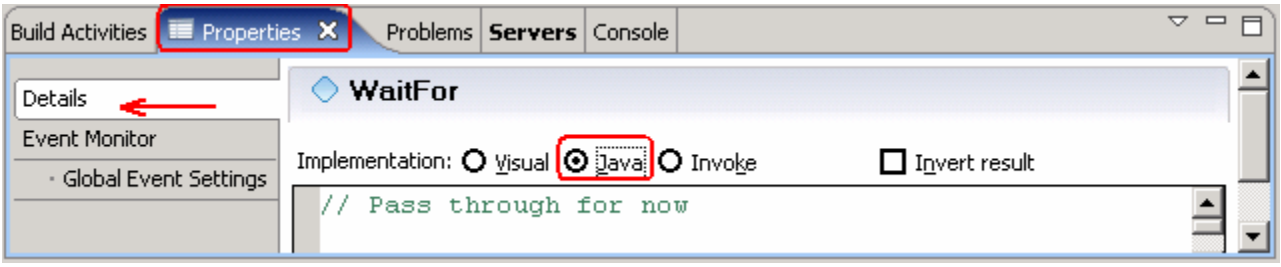
____ 2. Add Java logic to the '**WaitFor**' condition

__ a. Select the '**WaitFor**' condition on the activate transition between the '**Review**' and the '**Active**' states



__ b. While the '**WaitFor**' condition is selected, select the **Properties** tab

__ c. Select the **Details** tab



__ d. Ensure the implementation is set to '**Java**'

__ e. In the text field, enter the following code:

```
// Conditional logic to implement the "grace" requirement.

Date signOffTime = (Date)activate_Input_customerInfo.get("date");
Date currentDateTime = new java.util.Date();

System.out.println("*** BSM Condition - signoff time " + signOffTime.toString());

long t1;
long t2;
long delta;

    System.out.println("Signed Off: " + signOffTime.toString());
    System.out.println("Current:    " + currentDateTime.toString());

    t1 = signOffTime.getTime();
    t2 = currentDateTime.getTime();

    System.out.println(t1);
    System.out.println(t2);
    System.out.println(t2 - t1);

    delta = (t2-t1) /60000;                                  // minutes
    System.out.println( "Delta( > 1 min ?) : " + delta );

        if ( delta > 1 )
         {
          System.out.println("*** The 'grace' period is over.");
          return true;
         }
        else
         {
          System.out.println("*** Waiting for period of time.");
          System.out.println("*** Try again after 1 minute.");
          return false;
         }
```

Note: For your convenience, this code can be found in
<LAB_FILES>\BusinessStateMachines\Snippets\ConditionLogic.txt

---

__ f. **Save** (**Ctrl +S**) the business state machine

____ 3. Add both applications to the server

__ a. Right-click on the server and select '**Add and Remove projects**' from the pop-up menu

__ b. Click the '**Add All >>**' button to add both applications to the server

__ c. Click **Finish**

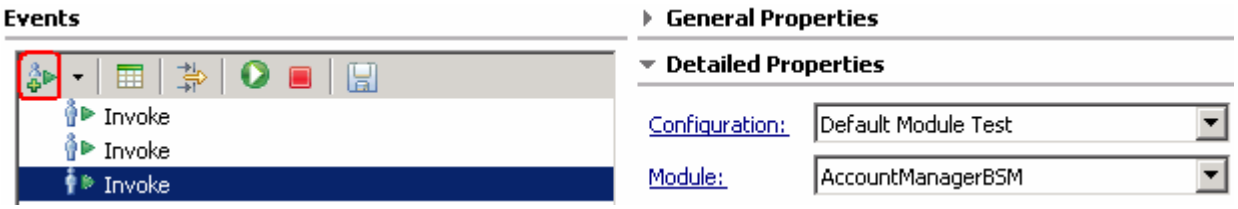
____ 4. Test the condition

__ a. In the project explorer, right mouse click the '**AccountManagerBSM**' module and select '**Test → Test Module**' from the pop-up menu

---

Note: If using a remote testing environment, select **Project → Properties → Integration Test Client**. Unselect the check box for '**Always use the default target in the test client**' if selected and click **OK**. You'll be prompted for your deploy location now when testing.

__ b. Add two additional invokes by clicking the **Invoke** button twice



Note: Remember that the accountId, **act1234**, is used to correlate the entire system. Enter **act1234** for accountID.

__ c. Set the component to **AccountManagerBSMExport** for all the three invokes

__ d. Setup the test conditions

1) For the first 'invoke', set the **Operation** to '**newCustomerAccount**'

a) Enter your own values or use those from Part 2

2) For the second 'invoke', set the **Operation** to **activate**

a) The values for this invoke **must** match those from the step above

3) Select the first invoke (newCustomerAccount) and select **Continue** ( )

__ e. This time you should see a message telling you to wait for one minute before invoking the review operation again

```
*** Waiting for period of time.
*** Try again after 1 minute.
```

__ f. Wait one minute and then with the second "invoke", invoke the **activate** operation manually using the **"Review signed off"** timestamp reported in the log (select the second invoke, edit the *customerInfo date* and select **Continue**). The following should then be displayed in the console along with some other diagnostic information:

```
*** The 'grace' period is over.
*** Leaving the Review state
*** Entering the Active state
```

__ g. For final 'invoke', enter the values and change to the '***terminate***' operation to move from the **active** state to the **non-Active**

__ h. If you wait five minutes in the **nonActive** state, the process will automatically terminate:

```
*** Leaving the notActive state
```

__ i. Close the test window without saving changes when you are finished

____ 5. Remove all applications from the server

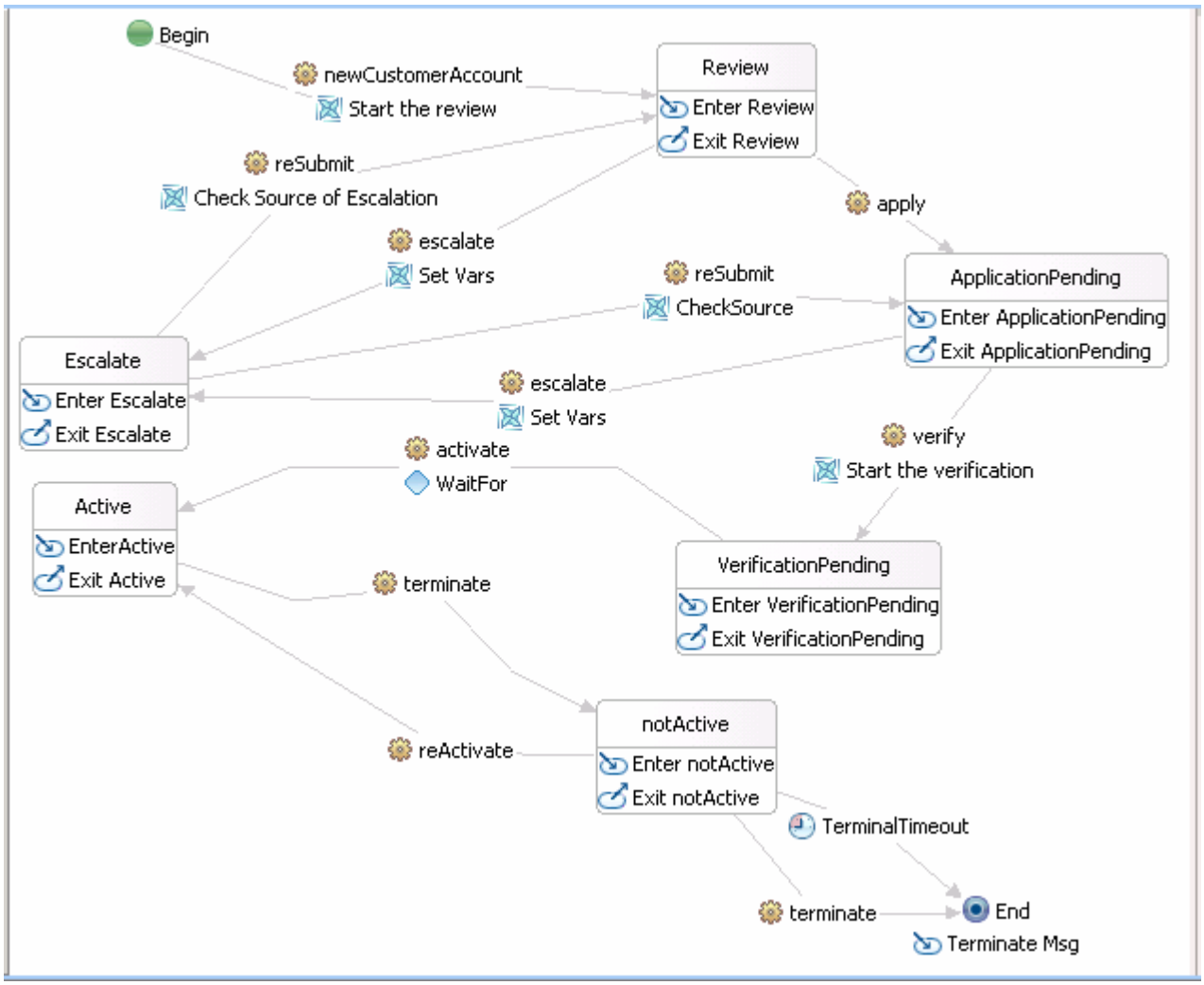
Part 4: Completing the business state machine

To complete the Business State Machine you will need to:

- Edit the '**AccountManagerInterface.wsdl**' to uncomment the operations for '***apply***' and '***verify***' transitions
    - Setup the correlations for the two new events
- Insert two new states, '**ApplicationPending**' and '**VerificationPending'**, between the '***Review***' and '***Active***' states
    - Add '***entry***' and '***exit***' messages to both states
- Update the **ReviewProcess** (BPEL)
    - To invoke the '***apply***' event instead of the '***activate***' event
    - Make adjustments to the comments in the Signed Off activity
    - These changes are incorporated in the updated version of the '**AccountManagementProcess**' module which is available for import
- Create a **VerificationProcess** (BPEL)
    - Create the '***signOffTime***' and invoke the *activate* event to the business state machine while in the '**Verification**' state
    - This will require updates to the module assembly for both the '**AccountManagementProcesses**' and the '**AccountManagerBSM**' modules.
    - The VerificationProcess (BPEL) is provided in the updated version of the '**AccountManagementProcess**' module which is available for import
- Add the transitions to the Escalate state from all the other states and put conditions on the transitions

You can use the '***customerInfo***' field in the '***Form***' business object to pass information that you can use for the condition test. This will be useful for the conditions on the escalate transitions and managing the '***signOffTime***'

Below is a picture of what the basic layout will look like when this section is completed:



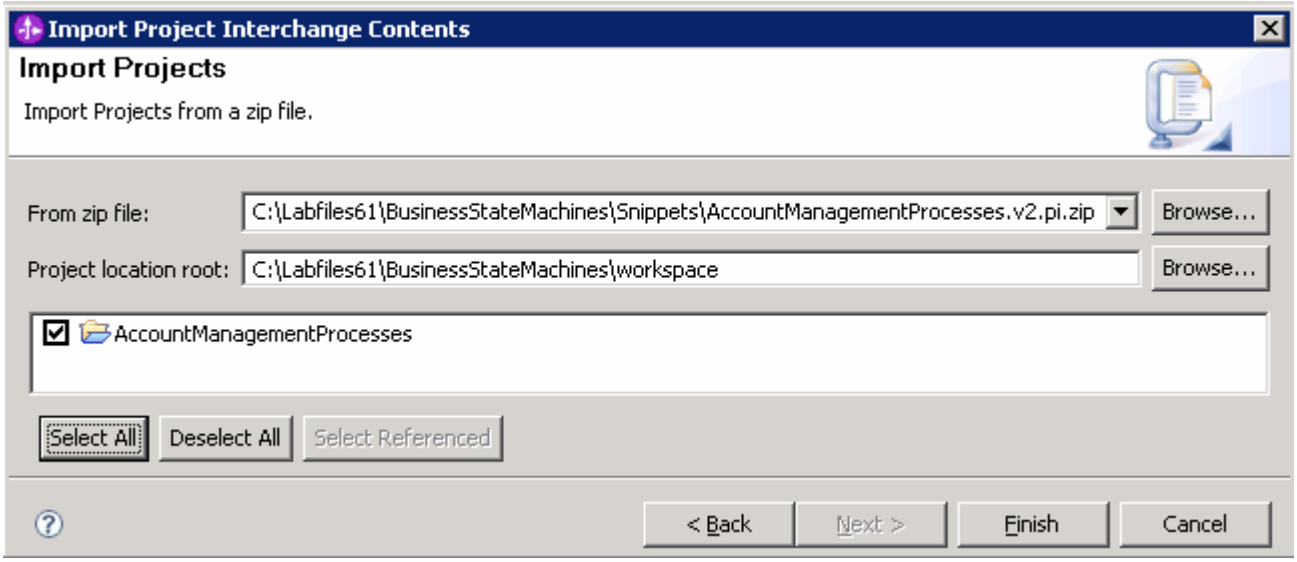
You will also add messages to the '***entry***' and '***exit***' actions of each new state and optionally the BPEL processes that will be invoked as Actions on '***apply***' and '***verify***' transitions.

____ 1. Import the updated version of the **AccountManagementProcess** module

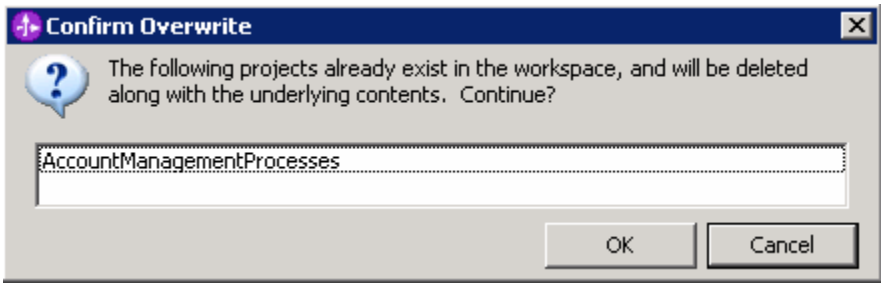
__ a. From the file menu, select **File → Import**

__ b. In the Import dialog, scroll down, expand **Other** and select **Project Interchange**

__ c. In the Import Projects dialog, initialize the From archive file to <LAB_FILES>\BusinessStateMachines\Snippet\AccountManagementProcesses.v2.pi.zip



__ d. Select the module and click **Finish**. A conformation dialog pops up



__ e. Click **OK** to confirm overwrite

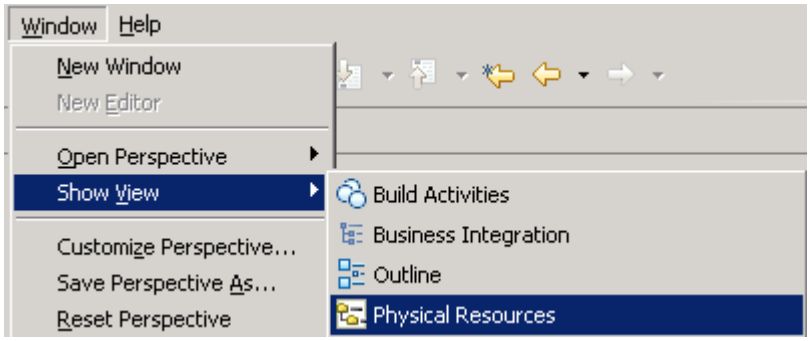
__ f. There will be errors related to the '***apply***' and '***verify***' events. These will be cleared up in subsequent steps

____ 2. Open and remove the commenting in the AccountManagerInterface.wsdl file for the *apply* and *verify* interface operations

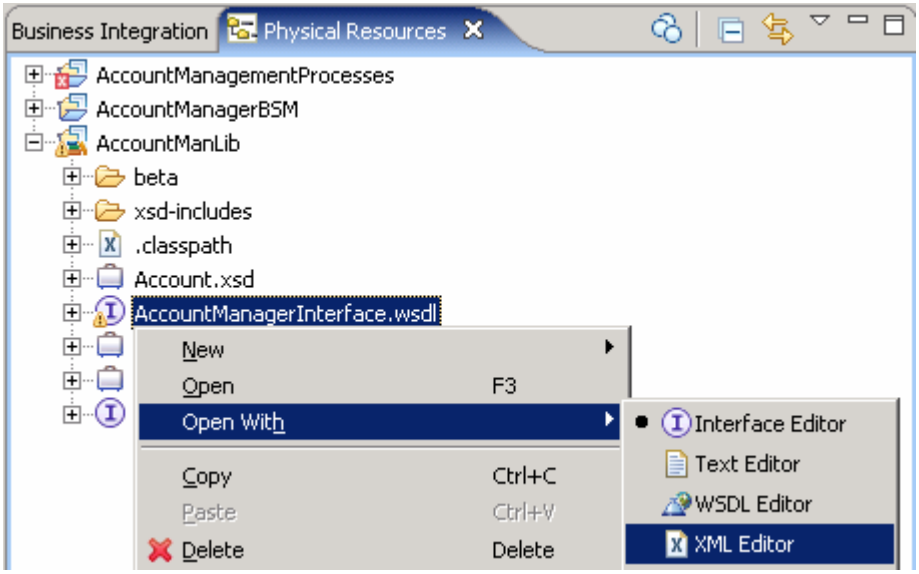
Before modifying the state transition design diagram, the '**AccountManagerInterface.wsdl**' file will need to be edited, that is the '***apply***' and '***verify***' operations in the interface have been commented out. They need to be uncommented.

Note: When developing a WSDL interface all the operations that are defined must be used. Unused operations will be flagged as errors. You can import or develop your interface completely and comment out the operations that you know you will not be using until a later date.

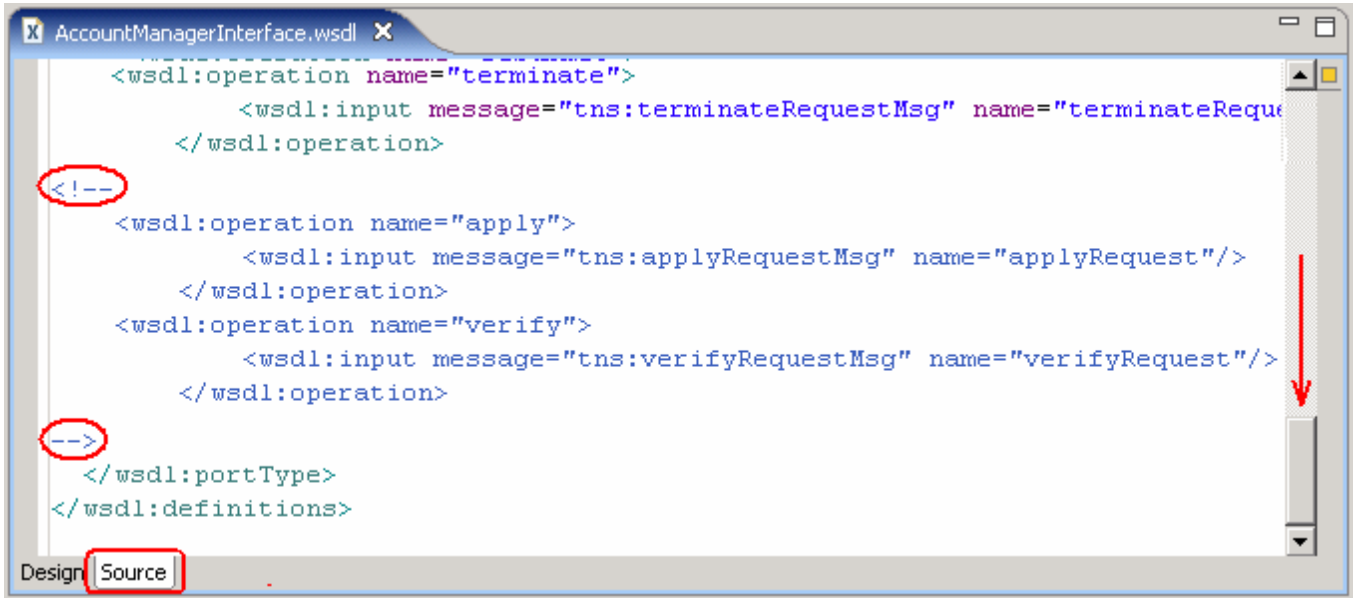
__ a. Go to the Physical Resources view by selecting **Window → Show View → Physical Resources**



__ b. In the physical resources explorer, expand '**AccountManLib'**, right-click on '**AccountManagerInterface.wsdl**' and select '**Open With → XML Editor**' from the pop-up menu



__ c. In the XML Editor, select the '**source**' tab and scroll to the bottom of the page. Delete the **<!-- -->** comment lines for the apply and verify operation coding



__ d. Save the changes (**Ctrl+S**)

__ e. Assign the new correlations for use in the **AccountManagerBSM** module

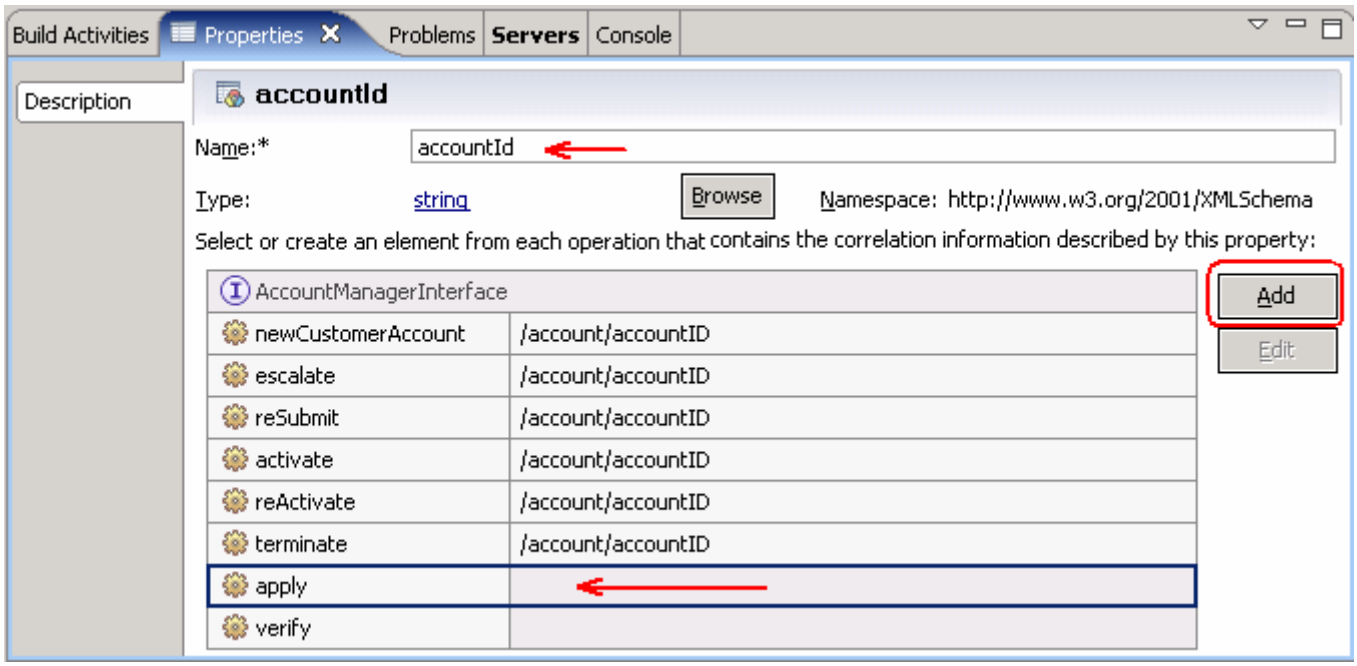
__ f. Switch back to the Business Integration explorer. You can close the **Physical Resources** explorer

__ g. In the project explorer, expand **AccountManagerBSM → Business Logic → State Machines**

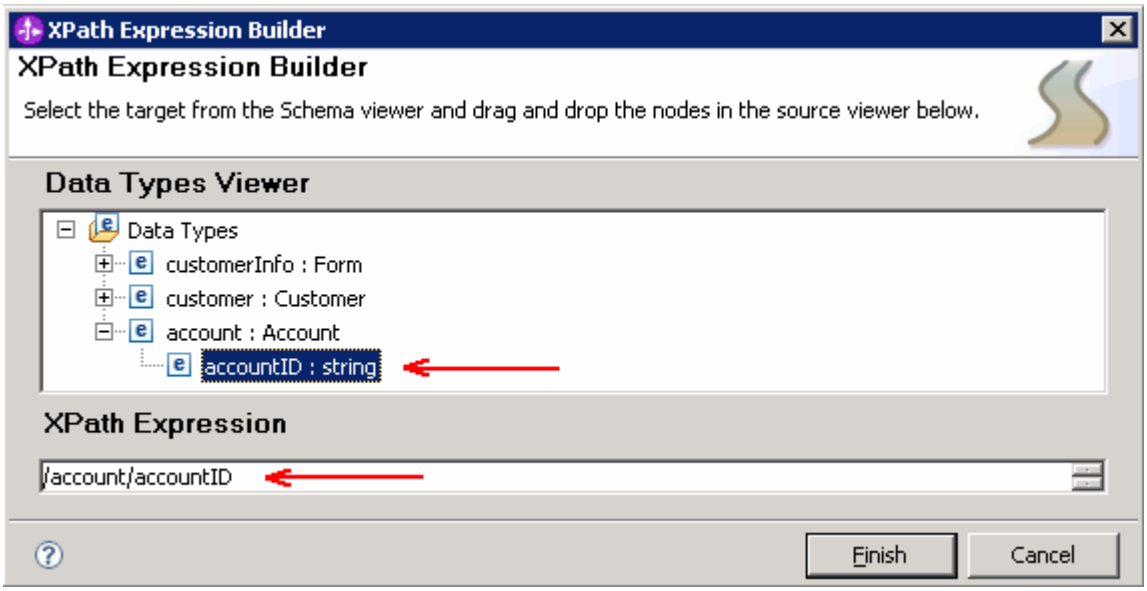
__ h. Double-click **AccountManagerBSM** to open the Business State Machine editor

__ i. Using the palette on the right, select the '**accountId**' property under the **Correlation Properties**

__ j. In the **Properties** tab, click on the column next to '**apply**' operation



__ k. Click the '**Add**' button. The '**XPath Expression Builder**' opens



__ l. In the '**Data Types Viewer**' section, expand '**account: Account**' and select **accountID**

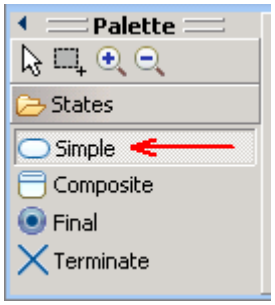
__ m. Click **Finish**

__ n. Repeat the same for '**verify**' operation

__ o. Save the changes (**Ctrl+S**)

____ 3. Add the **ApplicationPending** state to the business machine

__ a. Select the '**Simple State**' icon from the palette to the left



__ b. Click anywhere on the Business State Machine Editor canvas to create the new state.

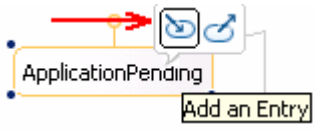
__ c. Rename the state as '**ApplicationPending**'

____ 4. Add the **VerificationPending** state to the business machine

__ a. Repeat step 3, naming the newly created state **VerificationPending**

____ 5. Add the ***Entry*** and ***Exit*** actions to **ApplicationPending**

__ a. Select the **ApplicationPending** state. A hover box will appear



__ b. Select '**Add an Entry**'

__ c. Rename the entry to **Enter ApplicationPending**

__ d. Select the entry, **Enter ApplicationPending** and then select **Properties** view

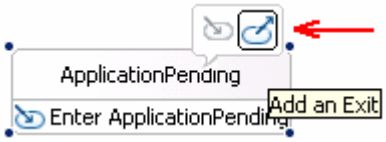
__ e. In the **Properties** view, click the **Details** tab and select **Java** implementation

__ f. Select '**Yes**' to the dialog box that appears

__ g. Enter the following code:

```
System.out.println("*** Entering the ApplicationPending state ***");
```

__ h. Select the **ApplicationPending** state and select **Add an Exit** from the hover box



__ i. Rename the exit to '**Exit ApplicationPending**'

__ j. Select the exit, **Exit ApplicationPending** and then select **Properties** view

__ k. Click the **Details** tab and select **Java** implementation

__ l. Select '**Yes**' on the dialog box that appears

__ m. Enter the following code:

```
System.out.println("*** Exiting the ApplicationPending state ***");
```

____ 6. Add the ***entry*** and ***exit*** for **VerificationPending**

__ a. Repeat step 5 for Verification Pending using the following entry/exit names and code snippets:

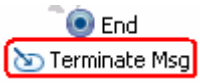
Entry name: **Enter VerificationPending** Exit name: **Exit VerificationPending**

```
System.out.println("*** Entering the VerificationPending state ***");
```

```
System.out.println("*** Exiting the VerificationPending state ***");
```

____ 7. Add the ***exit*** for the **End** final state

__ a. Add an entry action name **Terminate Msg**



__ b. Click the **Details** tab and select **Java** implementation

__ c. Enter the following code:

```
System.out.println("*** The Account Management Business State Machine has
terminated");
```

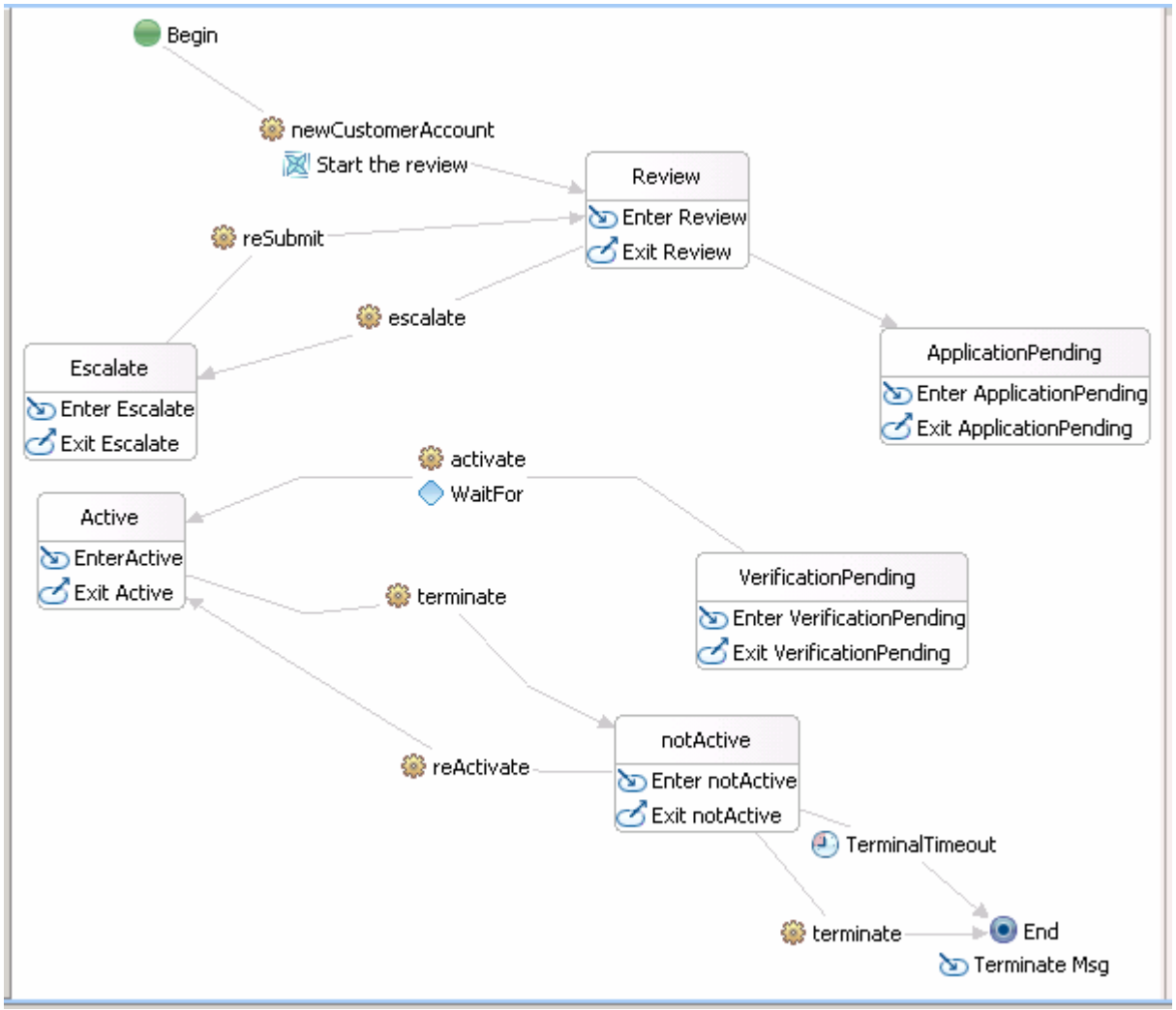
____ 8. Arrange the transitions so that you go from **Review** to **ApplicationPending** to **VerificationPending** to **Activate**. (See the diagram at the beginning of this section.)

__ a. Select the transition from the **Review** state to the **Active** state and drag the source end and attach it to the **VerificationPending** state

__ b. Select the wire for the **Timeout1** operation connecting the **Review** and **notActive** states; press the **delete** key to remove it

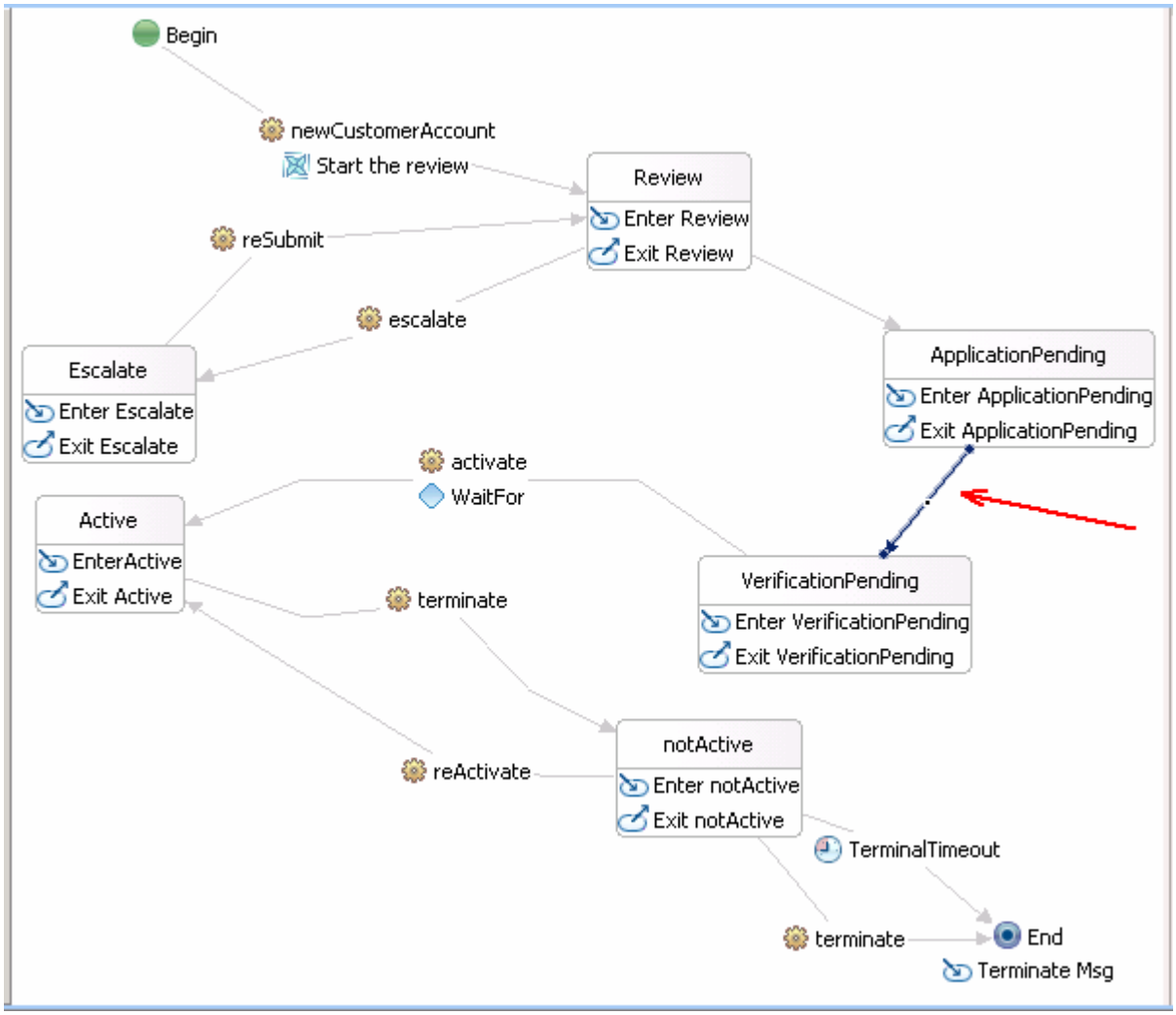
__ c. Select the wire for the **reSubmit** operation connecting the **Review** and **Active** states; press the **delete** key to remove it

__ d. Hover over the **Review** state until a yellow handle appears above the state. Then drag the handle to **ApplicationPending** to wire the two states together and create a transition



__ e. Create a transition from **ApplicationPending** to **VerificationPending**

__ f. The state machine diagram should look like:

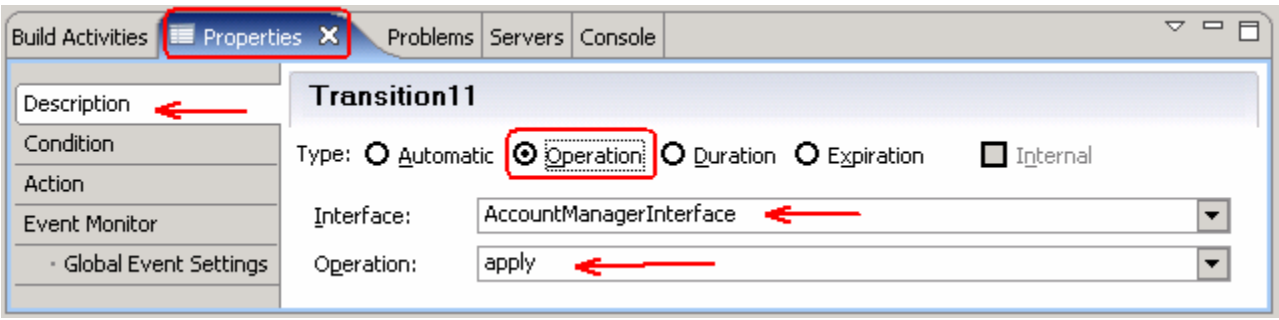


____ 9. Create operations for the new transitions

__ a. Select the transition connecting **Review** to **ApplicationPending**

__ b. Select the **Properties** tab at the bottom of the screen followed by the **Description** tab

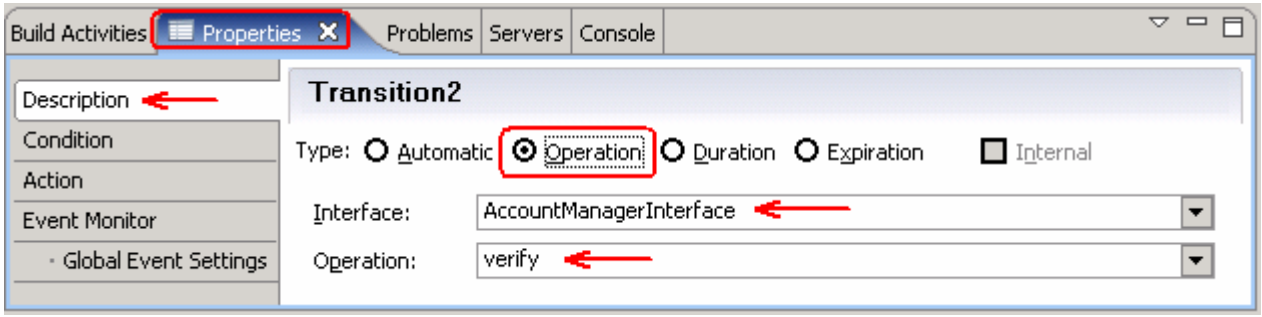
__ c. Set the Type to **Operation**, the Interface to **AccountManagerInterface**, and the Operation to **apply**



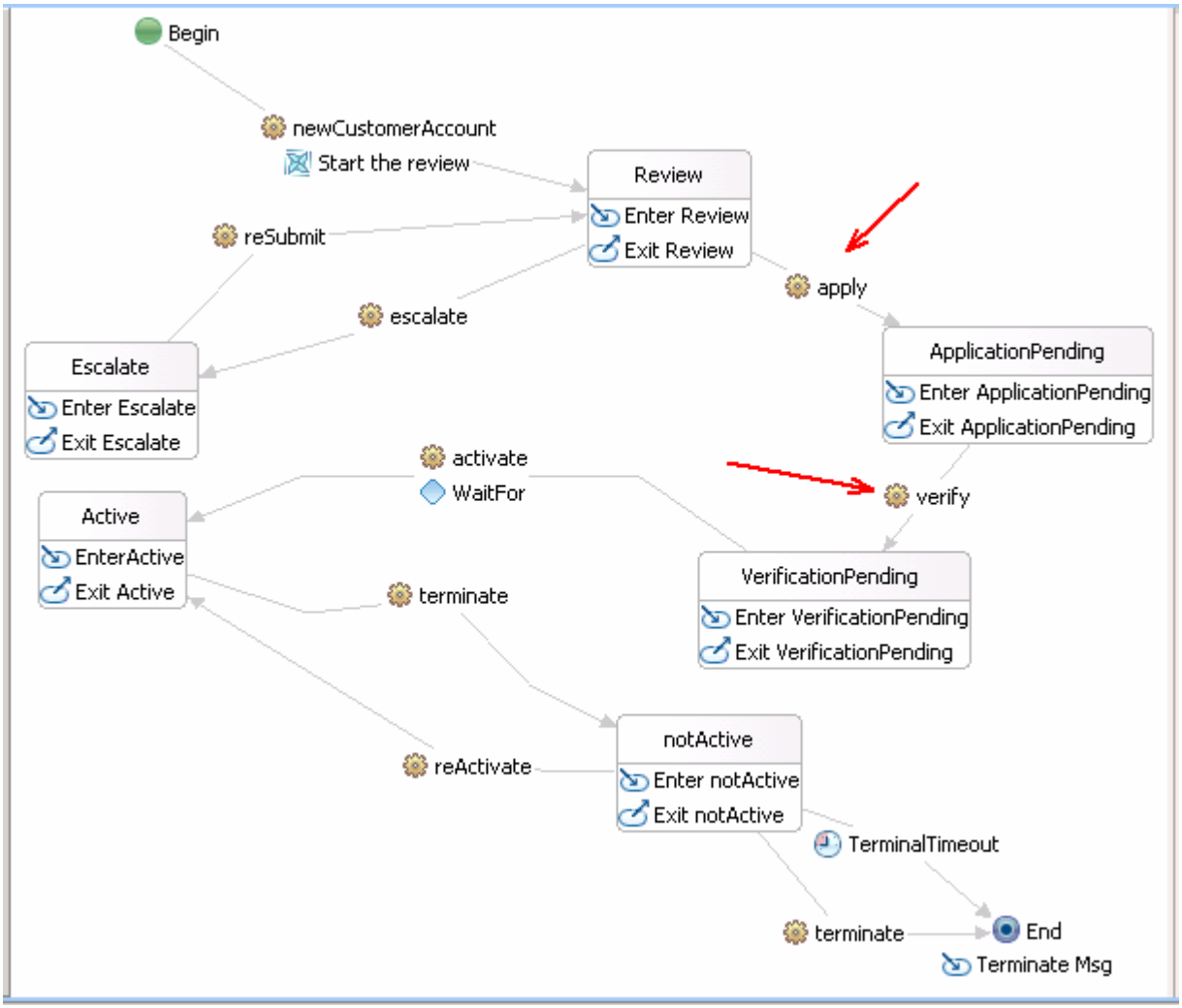
__ d. Now, select the transition connecting **ApplicationPending** to **VerificationPending**

__ e. Select the **Properties** tab at the bottom of the screen followed by the **Description** tab

__ f. Set the Type to **Operation**, the Interface to **AccountManagerInterface**, and the Operation to **verify**



__ g. The Business State Machine should now look like:



____ 10. Adjust the output in the Entry action of the Review state

__ a. Highlight Enter Review action of Review state and click on **Details** in Properties view

__ b. Comment out these lines which are no longer appropriate here

```
// System.out.println("*** If the Review has not been completed in 5min");
// System.out.println("*** then move to the notActive state");
// System.out.println("*** .. you cannot activate until 1 min after it has been signed off");
```

____ 11. Update the ReviewProcess (BPEL) to post the ***apply*** event

__ a. These changes were incorporated when the new version of the AccountManagementProcesses module was imported. Therefore, review these changes for your understanding

__ b. Change the **Activate** activity at the end of the business process to invoke the ***apply*** operation and change the name of the activity to **Apply**

____ 12. Setup the action to invoke the VerificationProcess (BPEL) on the ***verify*** transition from **ApplicationPending** to **VerificationPending**

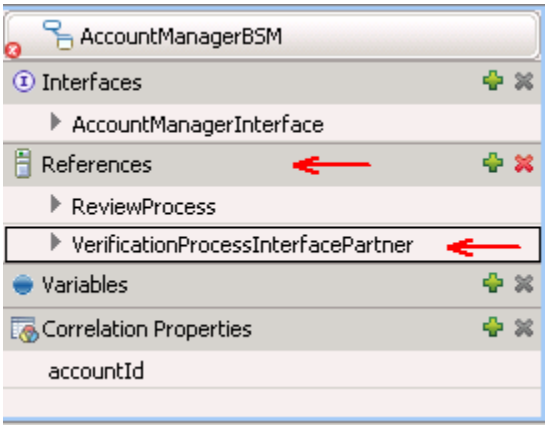
NOTE: The VerificationProcess is new with version 2 of the AccountManagementProcesses and is available if you did the import.

__ a. Open AccountManagerBSM editor again if you have closed it

__ b. Using the palette on the right, click ( ) to add the reference

1) Type **VerificationProcess** for reference name

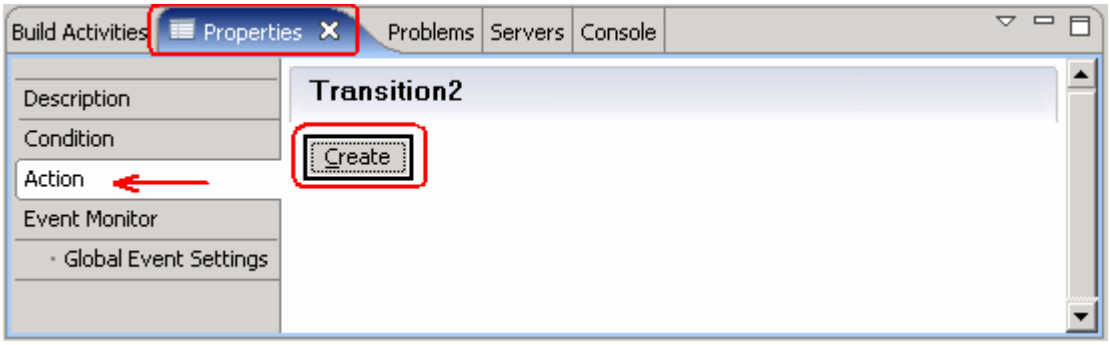
2) Select V**erificationProcessInterface** for interface



__ c. Create the action on the transition

1) Highlight the transition from **ApplicationPending** to **VerificationPending**

2) In Properties view, select Action and click on **Create** button



3) Highlight the new action and change the name to **Start the verification**

__ d. Add the reference that you just created for this action

__ e. Use this new reference and set variables in one to one mapping

1) Click on **Details** in Properties view of the action and select **Invoke** for Implementation

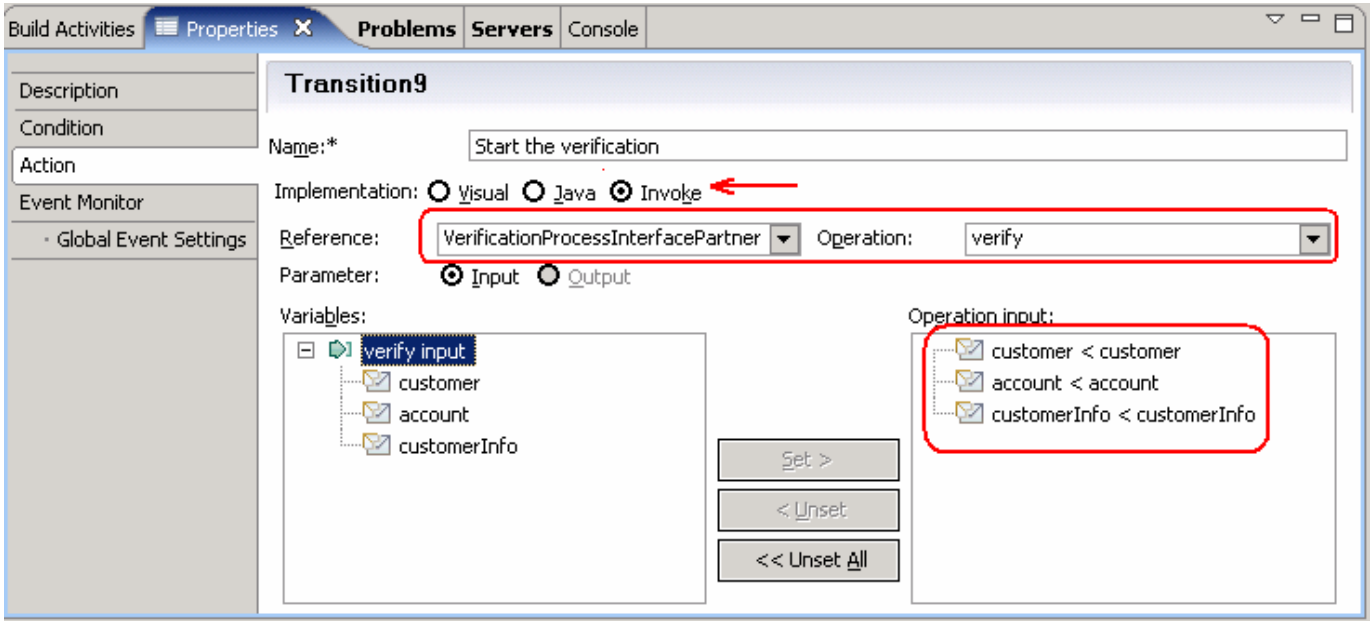
2) Select **VerificationProcess** for Reference and **verify** for Operation

3) Highlight **customer** in verify input of Variable field and highlight **customer** in Operation input

4) Click on **Set >**

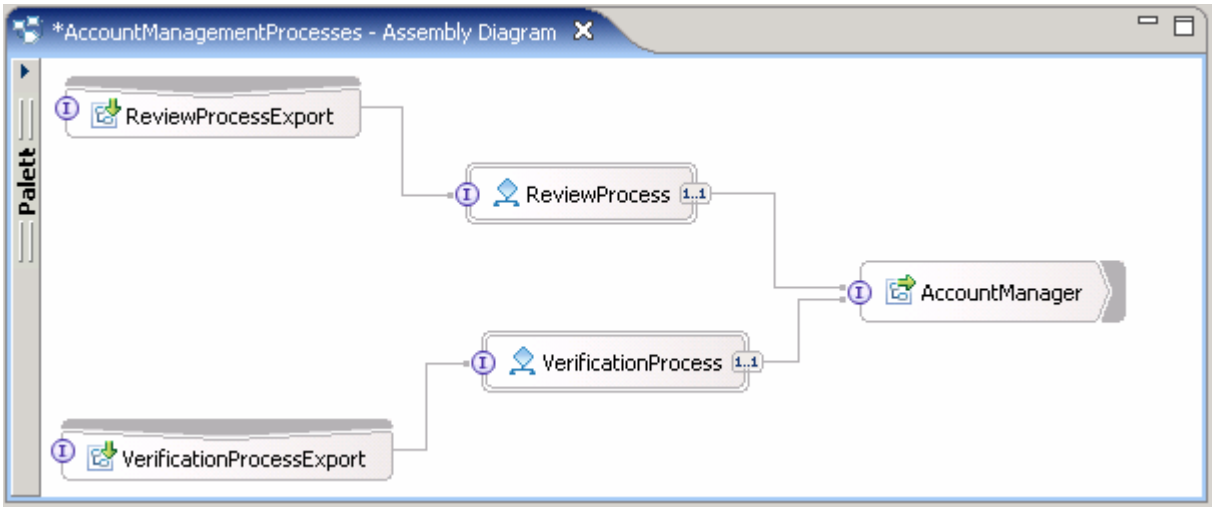
5) Repeat the step 3-4 for **account** and **customerInfo**

6) Save the changes (**Ctrl+S**)



__ f. Inspect the Module Assembly for the **AccountManagerProcesses** module. It was updated by the module provider and was picked up when the module was imported

1) The VerificationProcess was added to the **AccountManagerProcesses** module assembly was wired up using SCA bindings/



____ 13. Update the module assembly for the **AccountManagerBSM** module

__ a. Add the import for the **VerificationProcess** to the **AccountManagerBSM** module assembly and wire it to the **AccountManagerBSM**

1) Open the assembly diagram for the **AccountManagerBSM** module

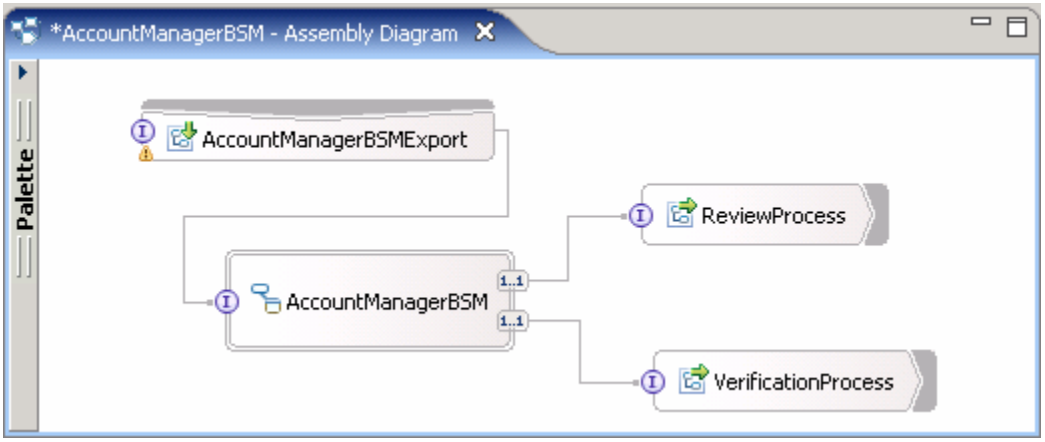
2) Drag the **VerificationProcessExport** from the **AccountManagementProcesses** module and drop it onto the assembly diagram and select "**import with SCA bindings**" when prompted

3) Rename to **VerificationProcess**

4) Delete the **AccountManagerBSM** from the diagram and save

5) Drag the **AccountManagerBSM** state machine back in and wire it to the export and imports

a) There should now be two references on the **AccountManagrerBSM** component



6) Save and run Clean the project

Test the changes that were made.

____ 14. Add both applications to the server

__ a. Right-click on the server and select **Add and remove projects**

__ b. Click the **Add All >>** button to remove both projects from the server

__ c. Click **Finish**

____ 15. Start the Component Tester on the **AccountManagerBSM** module.

__ a. Right click on the **AccountManagerBSM** module

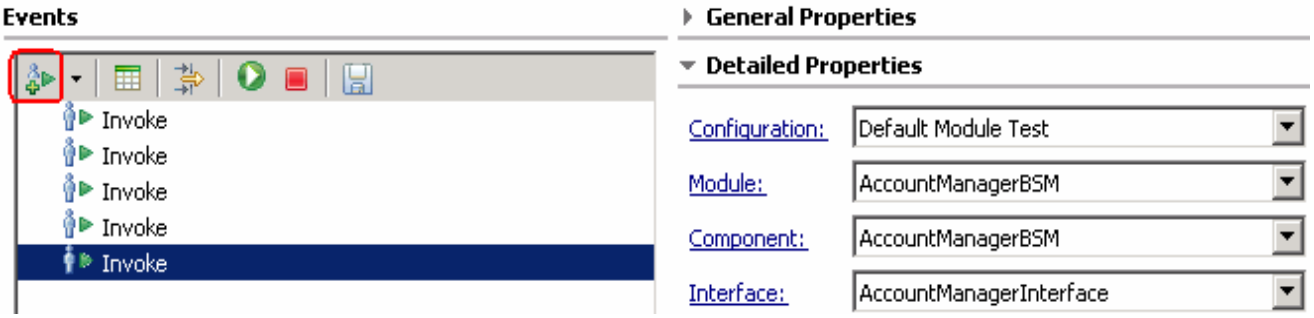
__ b. Select **Test → Test Module** from the pop-up menu

____ 16. Run the tests

Note: Remember that the **accountId**, ***act1234***, is used to correlate the entire system.

__ a. Add four additional invokes

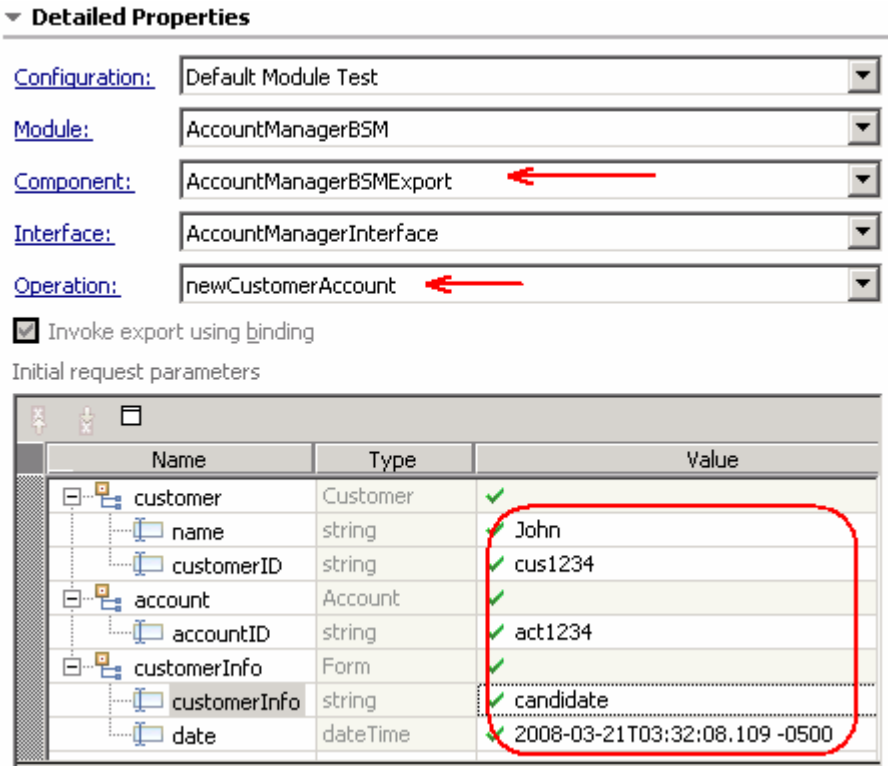
1) Click on the **Invoke** button four times; there should be five invokes listed under Events section



__ b. Set the **Component** to **AccountManagerBSMExport** for all the invokes

__ c. Begin testing the components.

1) For the first invoke, set the **Operation** to **newCustomerAccount**.



2) Select **Continue** ( ) to begin the test. You should see this output displayed to the console:

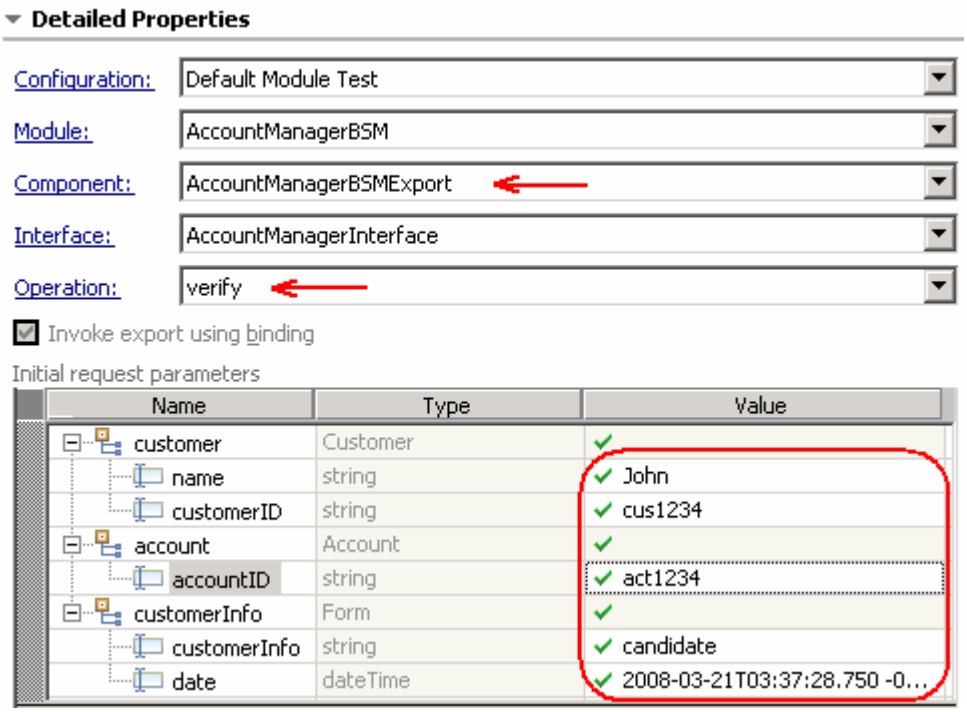
```
*** Entering the Review state
***** Review Process *****
*** We have a regular customer
*** Review signed off at: Fri Mar 21 03:30:27 CDT 2008
*** Leaving the Review state
*** Entering the ApplicationPending
```

Notice that the state machine went right through the Review state this time and behaved like an automatic transition. This because you send the apply event/operation form the context of the ReviewProcess (BPEL).

__ d. Use the second invoke to transition from the ApplicationPending state to the VerificationPending state

1) Set the **Operation** for the second invoke to **verify**

2) Enter the accountId, **act1234**



3) Select **Continue** ( ) and the following should be displayed to the console:

```
*** Exiting the ApplicationPending state ***
*** Entering the VerificationPending state ***
*** Verification Business Process
*** When complete, activate the new account
*** there is a rule, implemented as a condition on the transition
*** that requires a grace period before activating the new account
*** Request signed off and validated at: Fri Mar 21 03:40:40 CDT
2008
*** BSM Condition - signoff time Fri Mar 21 03:40:40 CDT 2008
Signed Off: Fri Mar 21 03:40:40 CDT 2008
Current:    Fri Mar 21 03:40:40 CDT 2008
1206088840000
1206088840359
359
Delta (> 1 min?): 0
*** Waiting for period of time.
*** Try again after 1 minute.0000008e SystemOut
```

__ e. Wait a least a minute and then use the third invoke to send the activate event/operation, using the ***RequestSignedOff*** time from your console log.

1) Enter the accountID (act1234).

2) Change the customerInfo date to the ***RequestSignedOff*** time from your console log

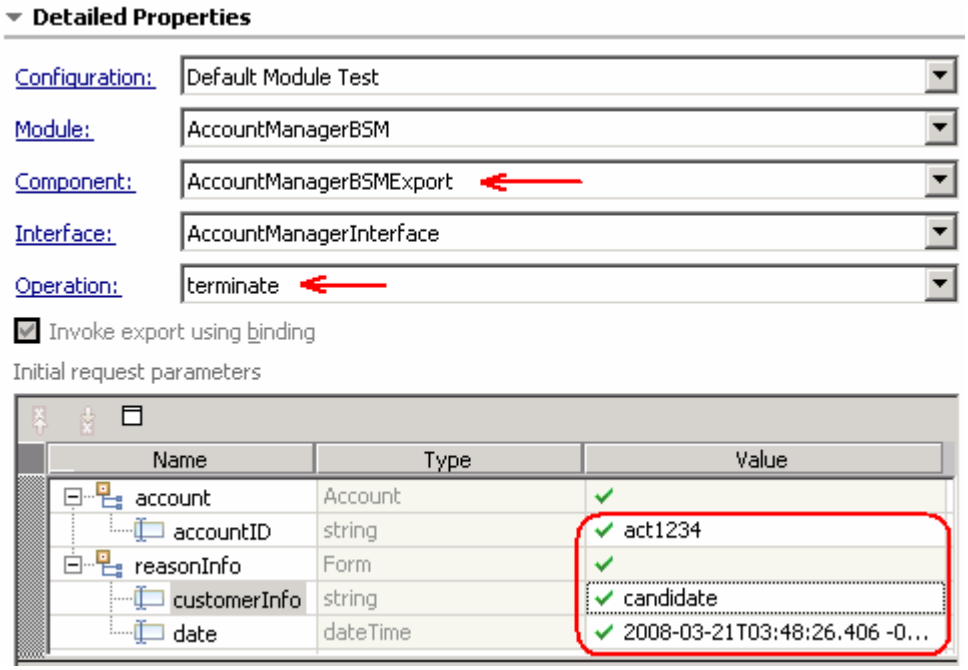
3) Set the third invoke to **activate**

4) Select **Continue** ( ) and the following should be displayed to the console:

```
*** The 'grace' period is over.
*** Exiting the VerificationPending state ***
*** Entering the Active state
```

__ f. Use the fourth invoke to send the terminate event/operation. This will move the state machine to the **notActive** state where it will wait for another terminate message or timeout after five minutes

1) For the final two invokes set the operations to **terminate**



2) Select the **terminate** operation and then **Continue** ( )

a) The log will display…

```
*** Entering the notActive state
*** Leaving the notActive state
```

3) Send the terminate event/operation again

a) The log will display…

```
*** The Account Management Business State Machine has terminated
```

4) Close the test window without saving changes when finished

Part 5: Add the escalate transitions (Optional)

To implement the escalations there must be distinguishing conditions on each of the reSubmit transitions so the state machine will return to the state from which it came. This can be achieved with the aid of a global variable which gets set during the transition to the escalate state, with a value that indicates the state the Business State Machine was in when the escalation event was received.

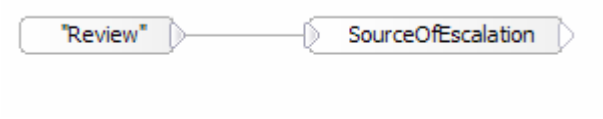
Example: If the Business State machine is in the *Review* state when the *escalate* event is received, then the value of the global variable is set to “Review”. The value of the global variable can then be checked in each of the reSubmit transitions.

____ 1. Create a global variable called **SourceOfEscalation,** of type string

__ a. Add an ***action*** to the ***escalate*** transition that goes from **Review** to **Escalate**

1) Name the action, **Set Vars**

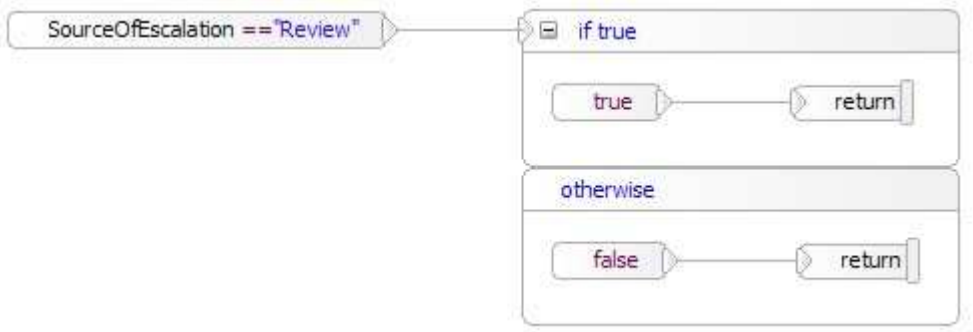
2) In the **Set Vars** action, set the **SourceOfEscalation** global variable to “Review”



__ b. Add a *condition* to the ***reSubmit*** transition that goes from **Escalate** to **Review**.

1) Call the condition, **Check Source of Escalation**

2) In this condition, check to see that the value in the SourceOfEscalation is equal to “Review”.



__ c. Add *escalate* and *reSubmit* transitions between the **Escalate** and **ApplicationPending** states.

1) Follow the same procedure used for the **Review** state but this time the value assigned to **SourceOfEscalation** global variable and compared in the condition is “ApplicationPending”

2) Complete the escalate/reSubmit pairs for the remaining states, VerificationPending and Active, using the appropriate values for **SourceOfEscalation** global variable and the condition check

---

Note that this can quickly become difficult to manage. This is a place where the composite state would provide benefit.

____ 2. Do another test cycle to verify the escalate transitions.

____ 3. Remove all applications from the server

__ a. Change to the servers view, right-click on **WebSphere Process Server v6.1** and select **Add and Remove projects…**

__ b. Click the **<< Remove All** button to remove both projects from the server

__ c. Click **Finish**

____ 4. Stop the server

__ a. Change to the servers view, right-click on **WebSphere Process Server v6.1** and select **Stop**

What you did in this exercise

- Created and tested a Business State Machine that collaborates with a BPEL business process by way of *Actions* on the transition
- Added *Conditions* to the Business State Machine to apply business restrictions to the overall business process flow
- Used the WebSphere Test Environment to move through the states of the Business State Machine.
- Programmatically sent events/operations to the Business State Machine from a long running BPEL business process

Solution instructions

No solution is available at this time

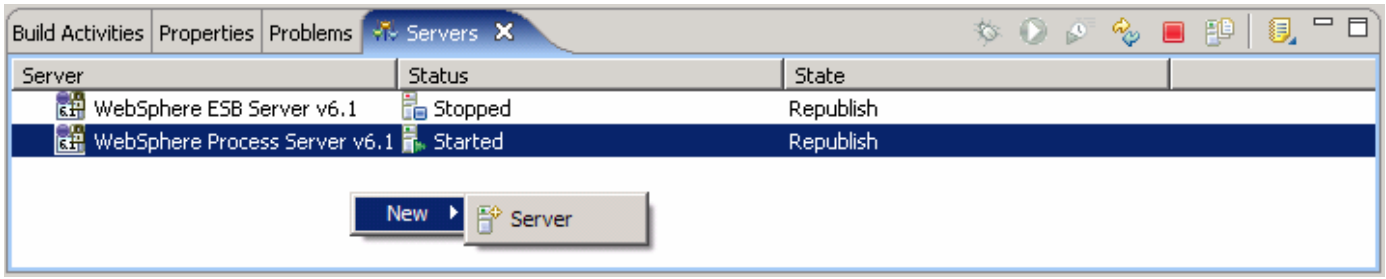
Task: Adding remote server to WebSphere Integration Developer test environment

This task describes how to add a remote server to the WebSphere Integration Developer test environment. This example uses a z/OS machine.

Create a new remote server.

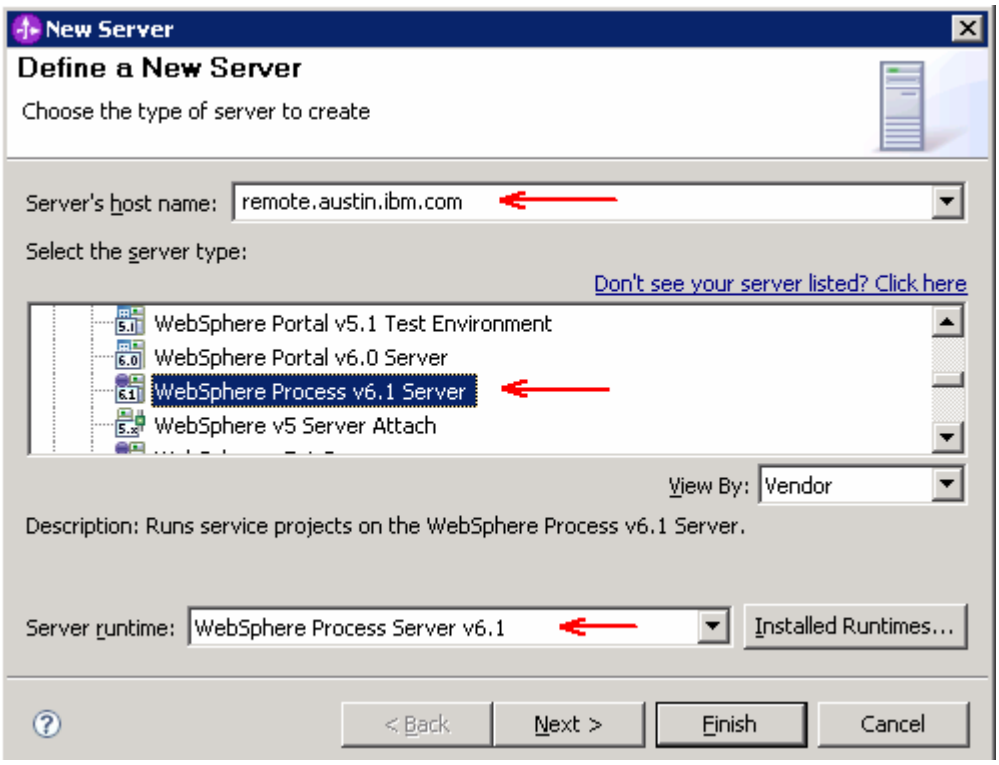
____ 1. Right click on the background of the Servers view

____ 2. Select **New → Server** from the pop-up menu



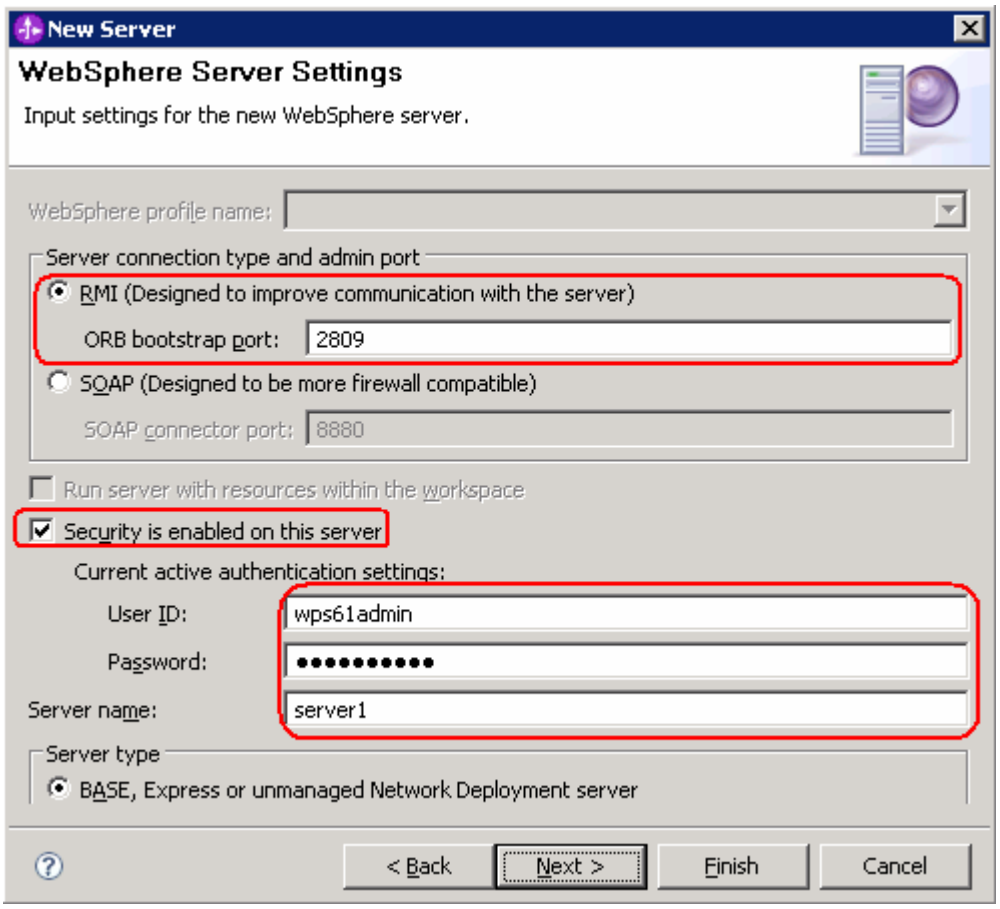
____ 3. Specify host name to the remote server, **<HOSTNAME>**

____ 4. Ensure that '**WebSphere Process v6.1 Server**' is selected from the server type list



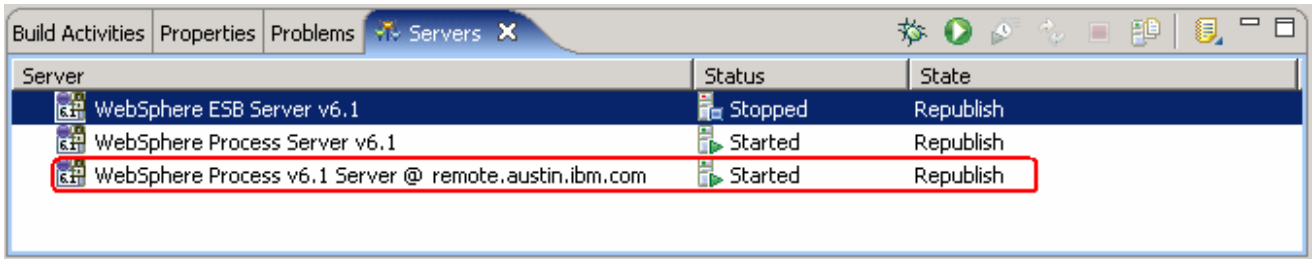
____ 5. Click **Next**

____ 6. On the WebSphere Server Settings page, select the radio button for **RMI** and change the ORB bootstrap port to the correct setting (**<BOOTSTRAP_PORT>**).



____ 7. Click **Finish**

____ 8. The new server should be seen in the Server view.



____ 9. Start the remote server if it is not already started. WebSphere Integration Developer does not support starting remote servers from the Server View.

____ 10. From a command prompt, telnet to the remote system if needed:

'**telnet <HOSTNAME> <TELNET_PORT>**'

user ID: **<USERID>**

password: **<PASSWORD>**

____ 11. Navigate to the bin directory for the profile being used:

cd <WAS_HOME>/profiles/<PROFILE_NAME>/bin

____ 12. Run the command file to start the server: **./startServer.sh <SERVER_NAME>**

____ 13. Wait for status message indicating server has started:

```
ADMU3200I: Server launched. Waiting for initialization status

ADMU3000I: Server cl1sr01 open for e-business; process id is 0000012000000002
```