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.

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.

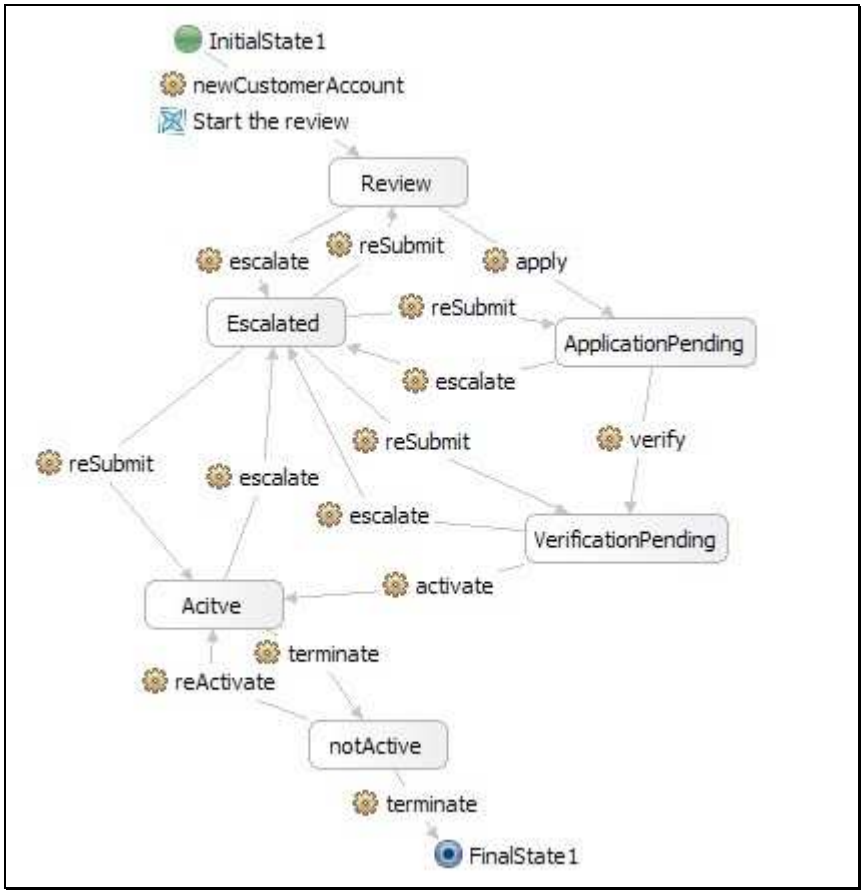
At any point in the business process the flow may be escalated to achieve greater focus and urgency or sent back to the previous step for rework.

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



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

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 2 SCA Modules and 1 Library Module.

1. AccountManagerBSM
2. AccountManagementProcesses
3. AccountManLib

The Library Module contains the interfaces and the schema definitions used by the other 2 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
<WAS_HOME>	C:\WebSphere60\AppServer	/usr/WebSphere60/AppServer /opt/WebSphere60/AppServer
<IRAD_HOME>	C:\Program Files\IBM\WebSphere\ID\6.0	
<LAB_FILES>	C:\LabFiles602	/tmp/LabFiles602
<TEMP>	C:\temp	/tmp

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, C:\LabFiles602\ would be replaced by C:/LabFiles602/

Note that the previous table is relative to where you are running WebSphere Integration Developer. This table is related to where you are running remote test environment:

Reference Variable	Example: Remote Windows test server location	Example: Remote z/OS test server location	Input your values for the remote location of the test server
<SERVER_NAME>	server1	cl1sr01	
<WAS_HOME>	C:\Program Files\IBM\WebSphere\App Server	/etc/cl1cell/AppServerNode1	
<HOSTNAME>	localhost	mvsxxx.rtp.raleigh.ibm.com	
<PORT>	9080	9080	
<BOOTSTRAP_PORT>	2809	2809	
<TELNET_PORT>	N/A	1023	
<PROFILE_NAME>	AppSrv01	default	
<USERID>	N/A	cl1admin	
<PASSWORD>	N/A	fr1day	

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:\LabFiles602\BusinessStateMachines\workspace

<PROJECT_INTERCHANGE>

C:\LabFiles602\BusinessStateMachines\Snippets\AccountManager.beta.pi.zip

---

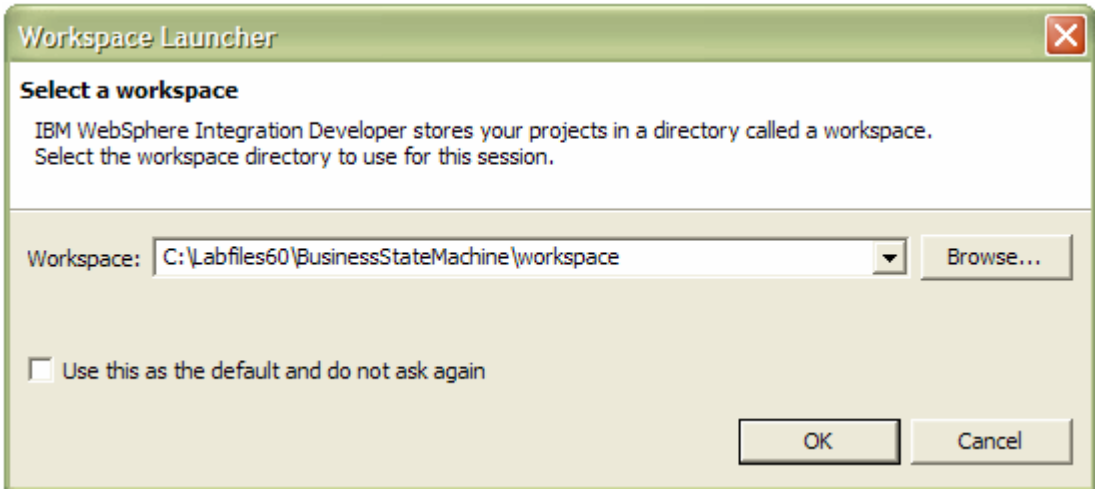
NOTE: When you first import the modules the autobuild feature will start to generate some of the artifacts for the application. Wait until the autobuild has completed before continuing.

Not everything required to run the application will be generated at this time. Some EJB artifacts won't be built until the application is published. These are runtime artifacts and are not required for the inspection tour you're about to do.

____ 2. Start WebSphere Integration Developer V6 with a new workspace located at **<WORKSPACE>**.

__ a. From Windows Explorer, navigate to the **<WID_HOME>** directory and double click on wid.exe.

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



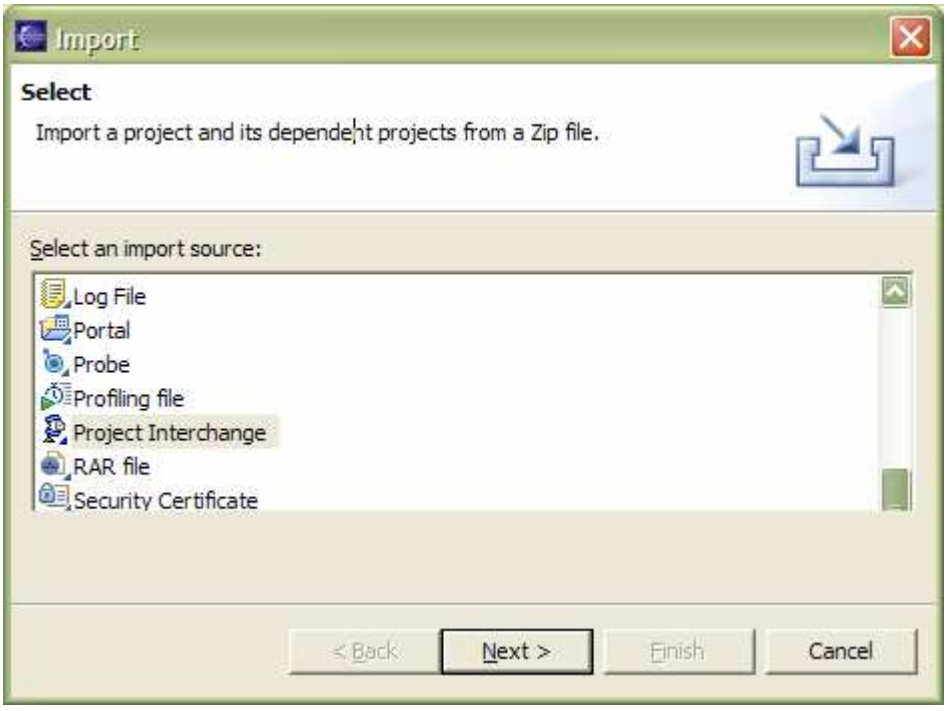
__ c. When WebSphere Integration Developer opens, close the **Welcome page** by clicking on the Go to the workbench icon (bent over arrow at top-right).

__ d. 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 menu bar, select **File -> Import…**

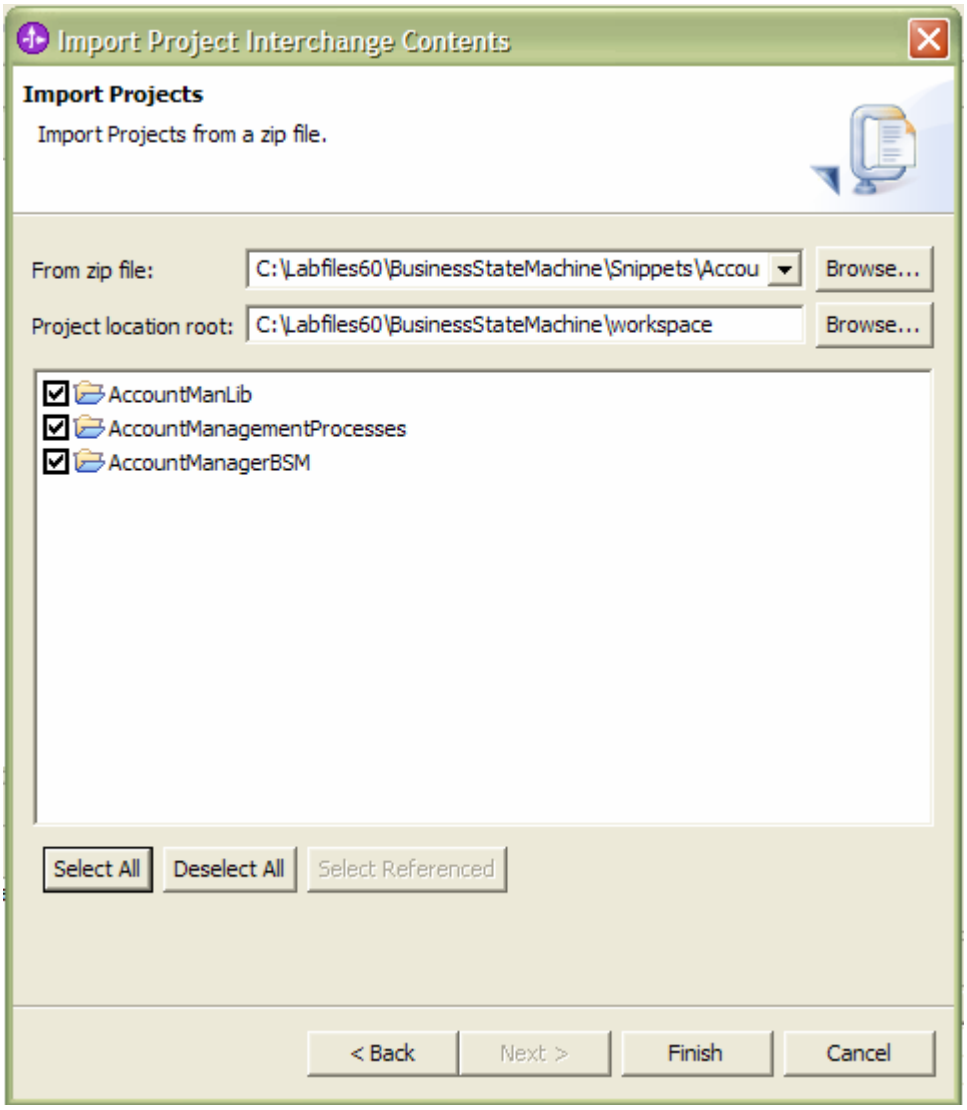
__ b. In the Import dialog, scroll down and select **Project Interchange.**



__ c. Click **Next.**

__ d. In the Import Projects dialog, initialize the From zip file: field to <**PROJECT_INTERCHANGE>.**

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



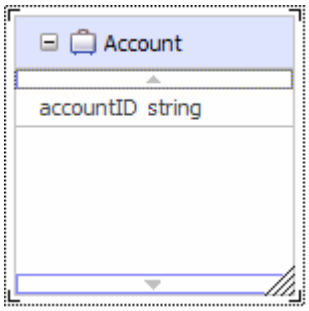
__ f. Click **Finish**

__ g. Inspect the data objects in the AccountManLib library.

NOTE: The reporting feature will give you a nice PDF document describing the interface, objects and components.

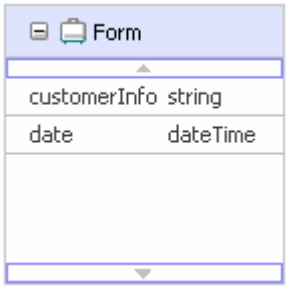
__ h. In the Business Integration perspective, expand **AccountManLib** and **Data Types**.

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



__ j. Repeat step b to inspect **Customer** business objects.

__ k. Repeat step b to inspect **Form** business objects



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**.

____ 4. Inspect the Interfaces in the AccountManLib library.

__ a. Expand **AccountManLib** and **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

____ 5. Inspect the business process in the AccountManagementProcess module.

__ a. Expand **AccountManagementProcess**

__ b. Expand **Business Logic**

__ c. Expand Processes

__ d. Double click on **ReviewProcess** to open Business Process Editor

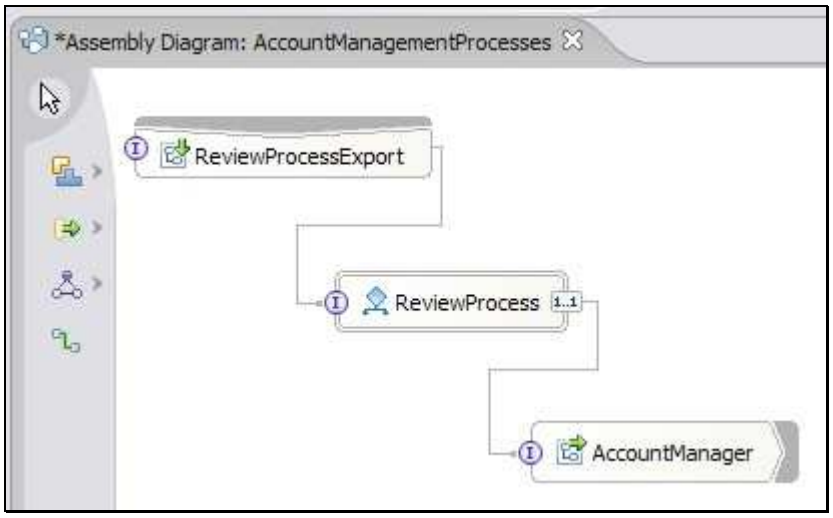
1) ReviewProcess is a simple and incomplete business process at this time. There will eventually be one BPEL business process per state. It is not a requirement to implement an Action as a business process; these instructions use a long running business process to illustrate what can be done and to demonstrate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

____ 6. Inspect the module assemblies for AccountManagementProcess and AccountManagerBSM.

__ a. Expand **AccountManagementProcess**

__ b. Double-click on **AccountManagementProcess** to open the Assembly Diagram.

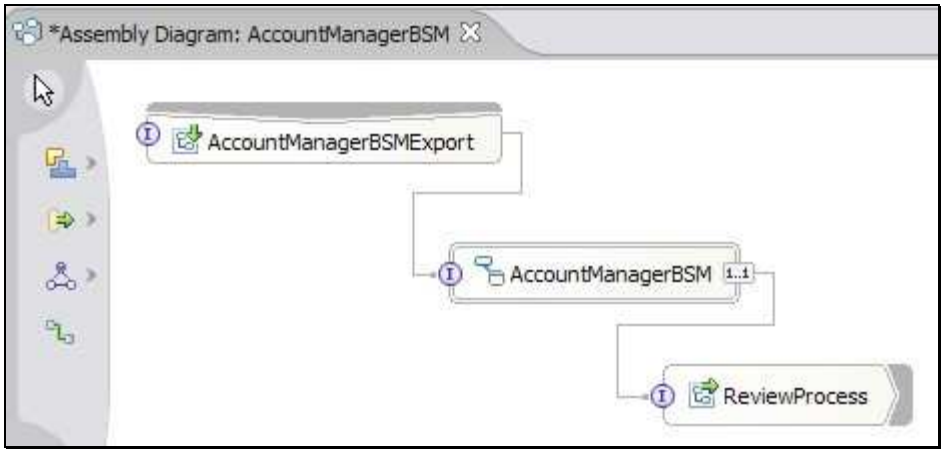


1) There is nothing special except the simplicity.

2) AccountManagementProcess uses SCA bindings.

__ c. Expand **AccountManagerBSM**

__ d. Double-click on **AccountManagerBSM** 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 3 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.

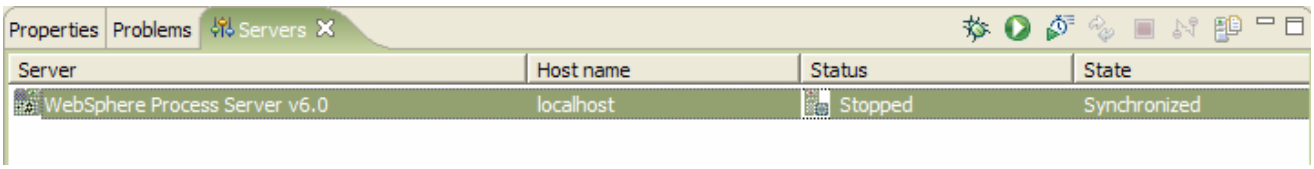
If you are interested in the *Escalate* state then take a look at the *ReviewProcess*.

____ 1. Start the test server.

__ a. Change to the Servers view.

__ b. 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.

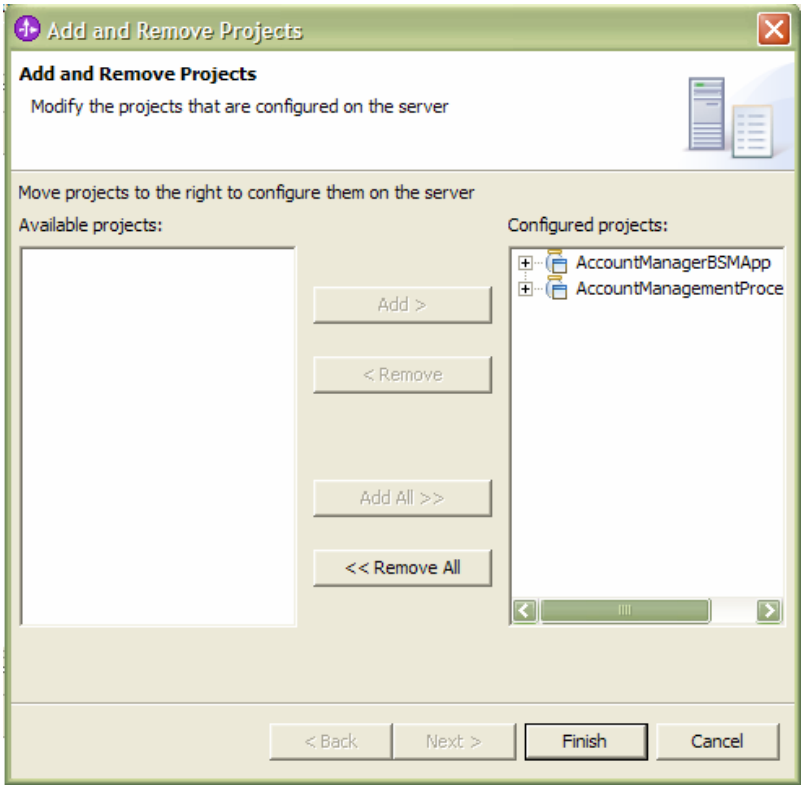
If using a local testing environment, right-click on **WebSphere Process Server v6.0** and select **Start**.



____ 2. Add both applications to the server when the server has finished starting.

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

__ 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.

1) Right click on **AccountManagerBSM** module.

2) Select **Test -> Test Module**

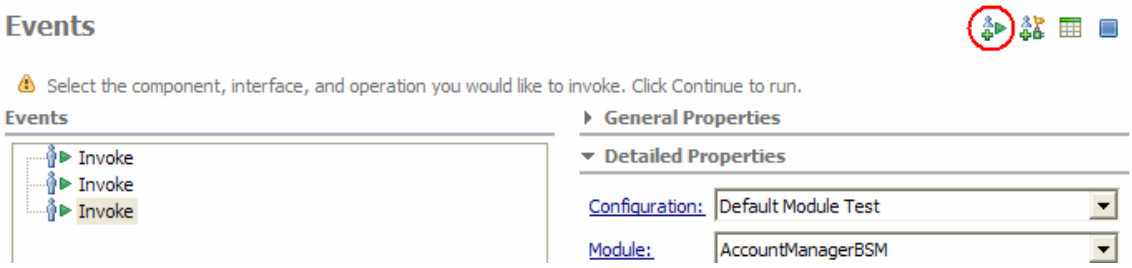
3) If using a remote testing environment, Select **Project-> Properties -> Integration Test Client.** Notice 'Always use the default target in the test client.' box is checked. Uncheck it and select **OK**. You'll be prompted for your deploy location now when testing.

____ 4. Run the tests.

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

__ a. Add 2 invokes.

1) Click on the **Invoke** button two times; there should now be three invokes under Events.



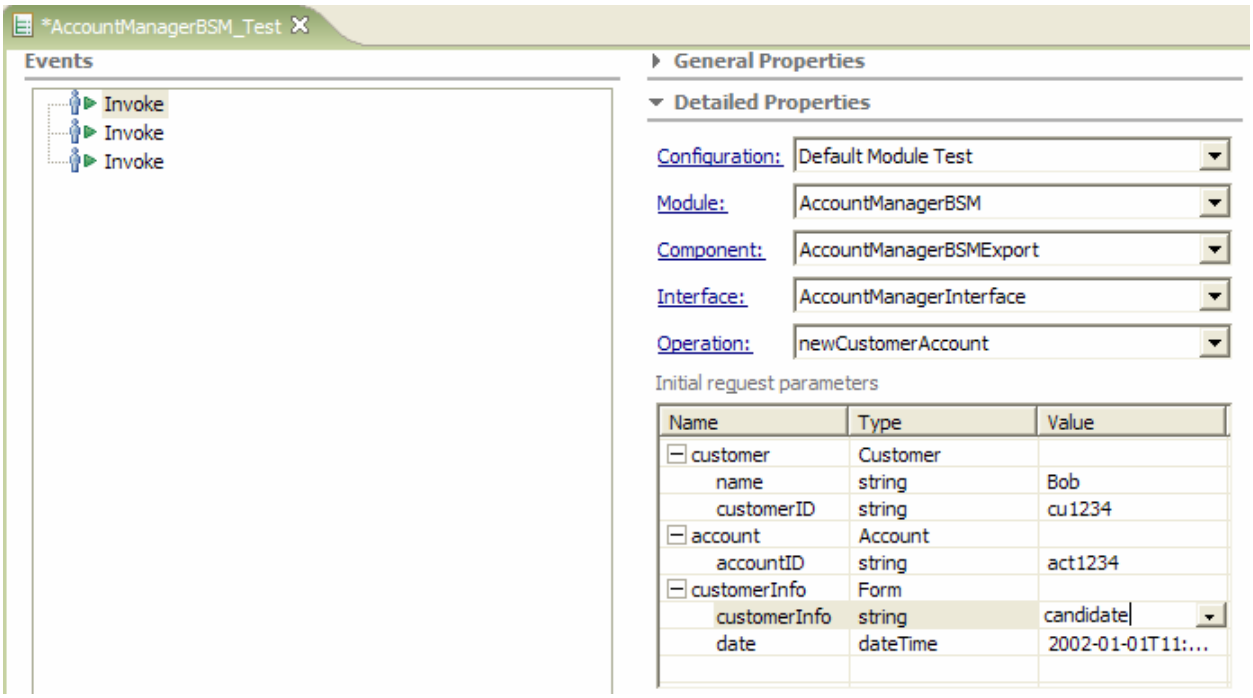
__ b. Highlight the Invoke 1 and select **AccountManagerBSMExport** for **Component** in Detailed Properties.

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

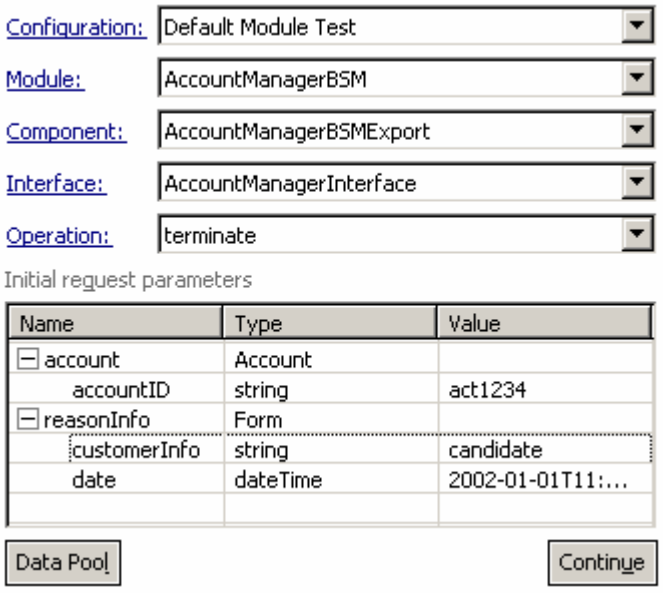
__ d. Set the input variables

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.

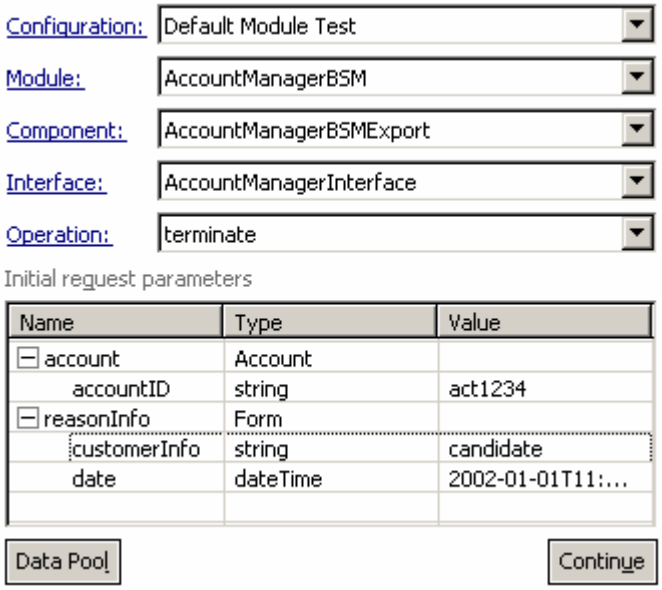
Looking at the console messages you will see that the ReviewProcess business process was kicked off and at the end it sends the *activate* event to the business state machine, which moves it to the *Review* state.

__ f. Step through the state machine using the other 2 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/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 v6.0** and select **Add and remove projects…**

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

__ c. Click **Finish.**

Part 3: Add 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. Expand the AccountManagerBSM.

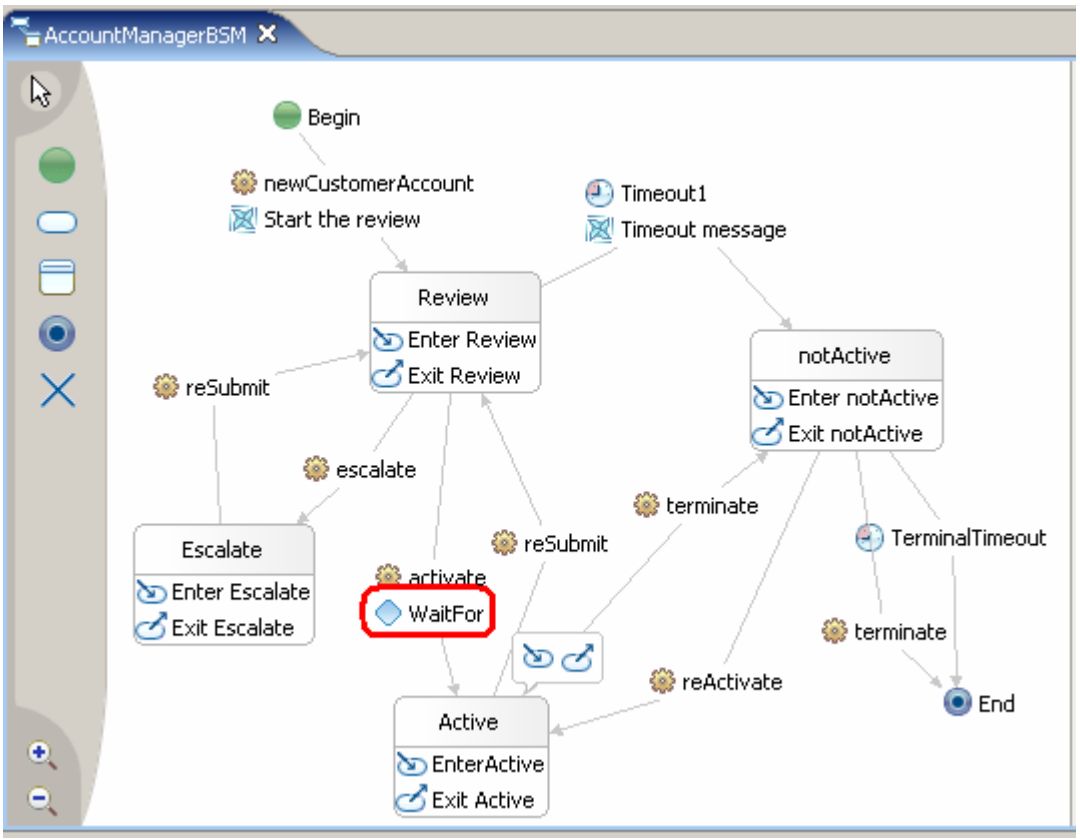
__ a. Expand **Business Logic**

__ b. Expand **State Machines**.

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

____ 2. Add Java logic to the WaitFor condition.

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



__ b. Select the **Properties** tab.

__ c. Select the **Details** tab.

__ d. Make sure the implementation is 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** the business state machine.

____ 3. 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 add both applications to the server.

__ c. Click **Finish**.

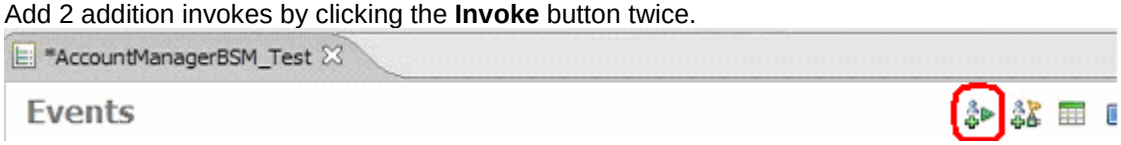
____ 4. Test the condition.

__ a. Select the AccountManagerBSM module.

__ b. Right mouse click and select **Test > Test Module**.

__ c. If using a remote testing environment, Select **Project-> Properties -> Integration Test Client.** Notice 'Always use the default target in the test client.' box is checked. Uncheck it and select **OK**. You'll be prompted for your deploy location now when testing.

__ d. Add 2 addition invokes by clicking the **Invoke** button twice.



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

__ e. Set the component to **AccountManagerBSMExport** for each invoke.

__ f. 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**.

__ g. 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.
```

__ h. Wait 1 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
```

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

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

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

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

____ 5. Remove all applications from the server.

__ a. Change to the servers view, right-click on **WebSphere Process Server v6.0** and select **Add and remove projects…**

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

__ c. Click **Finish.**

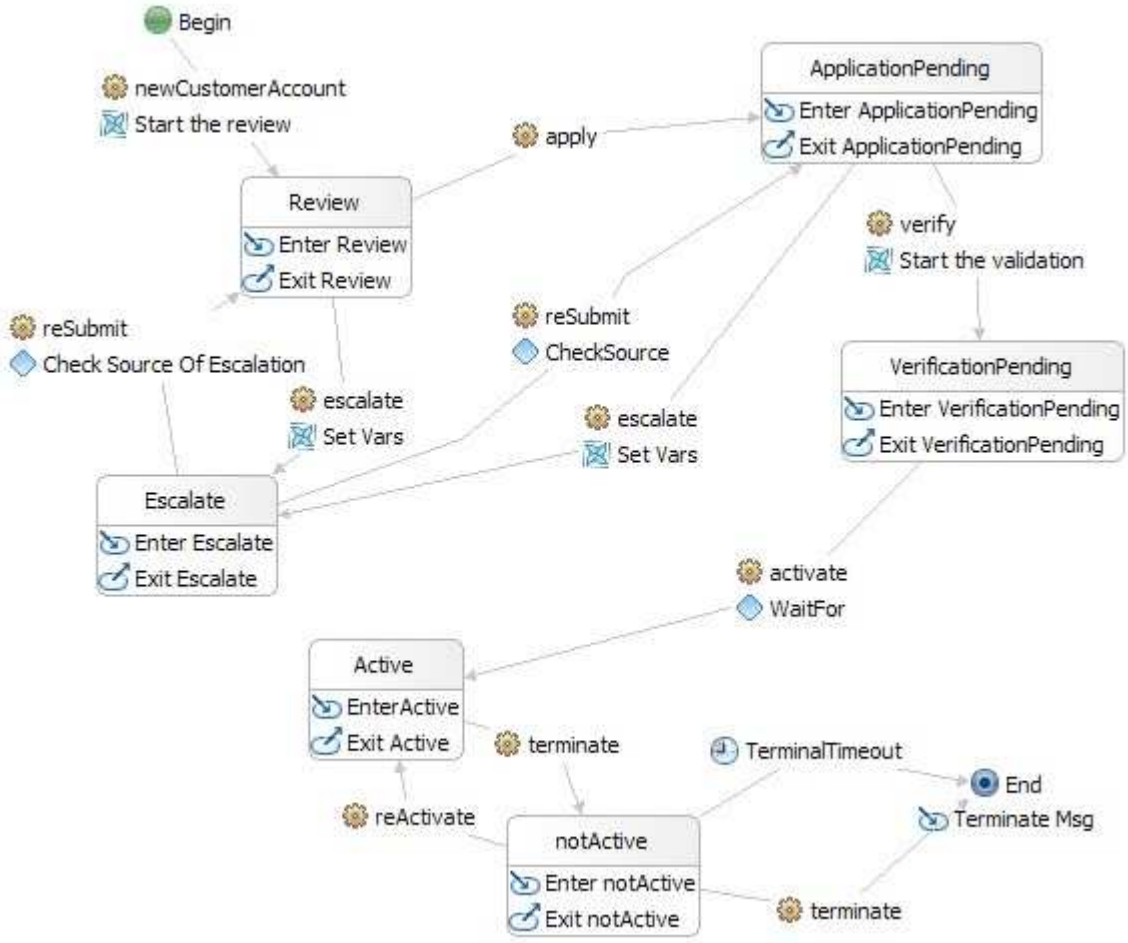
Part 4: Completing the business state machine

To complete the Business State Machine you will need to:

- Edit the AccountManagerInterface.wsdl to uncomment the operations for the *apply* and *verify* transitions.
    a. Setup the correlations for the 2 new events.
- Insert 2 new states, **ApplicationPending** and **VerificationPending**, between the Review and Active states.
    a. Add *entry* and *exit* messages to both states.
- Update the **ReviewProcess** (BPEL)
    a. To invoke the *apply* event instead of the *activate* event.
    b. Make adjustments to the comments in the Signed Off activity.
    c. These changes are incorporated in the updated version of the **AccountManagementProcess** module which is available for import.
- Create a VerificationProcess (BPEL)
    a. Create the ***signOffTime*** and invoke the *activate* event to the business state machine while in the **Verification** state.
    b. This will require updates to the module assembly for both the **AccountManagementProcesses** and the **AccountManagerBSM** modules.
    c. 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.

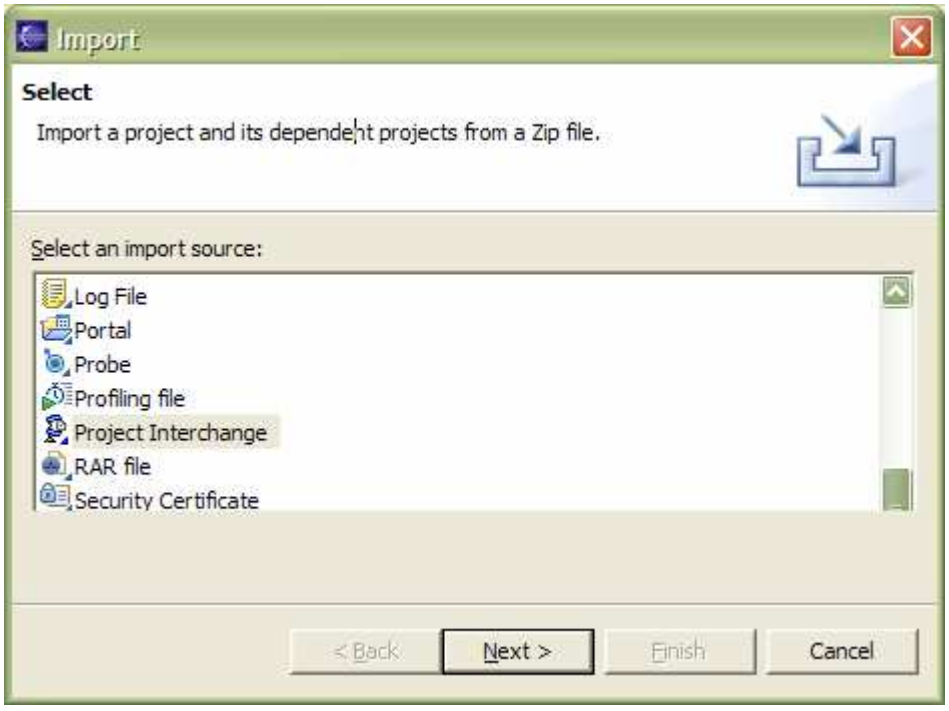


You will also add messages to the *entry* and *exit* actions of each new state and optionally the BPEL processes that will be invoked as Actions on the *apply* and *verify* transitions.

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

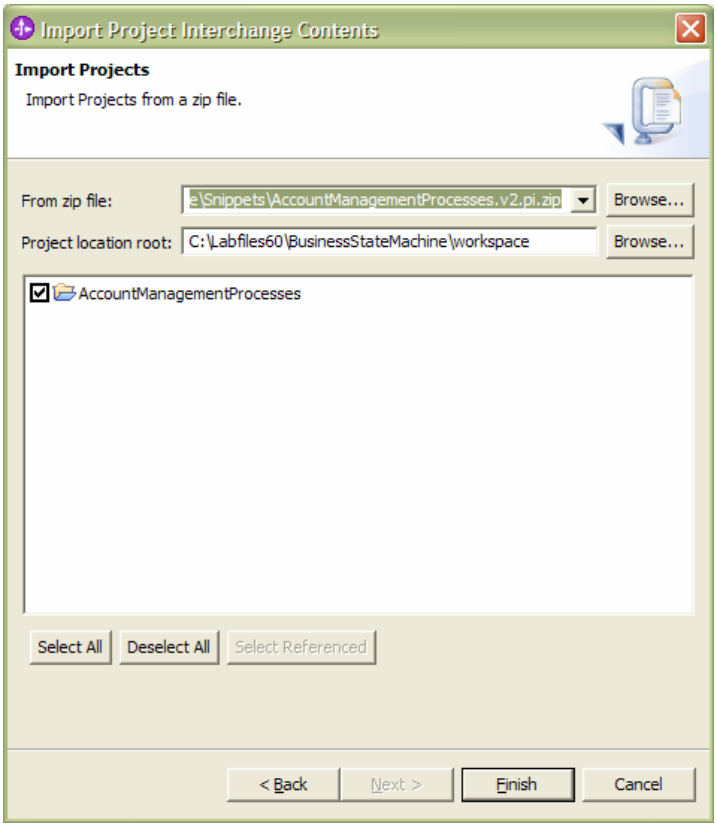
__ a. From the menu bar, select **File -> Import**

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



__ c. AccountManagementProcesses.V2.pi.zip

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



__ e. Select the module and click **Finish**

__ f. Click **Okay** to confirm overwrite

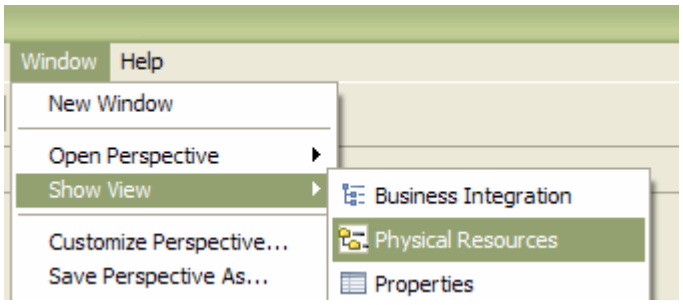
__ g. There will be errors related to the *apply* and *verify* events. These will be cleared up in subsequent steps.

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

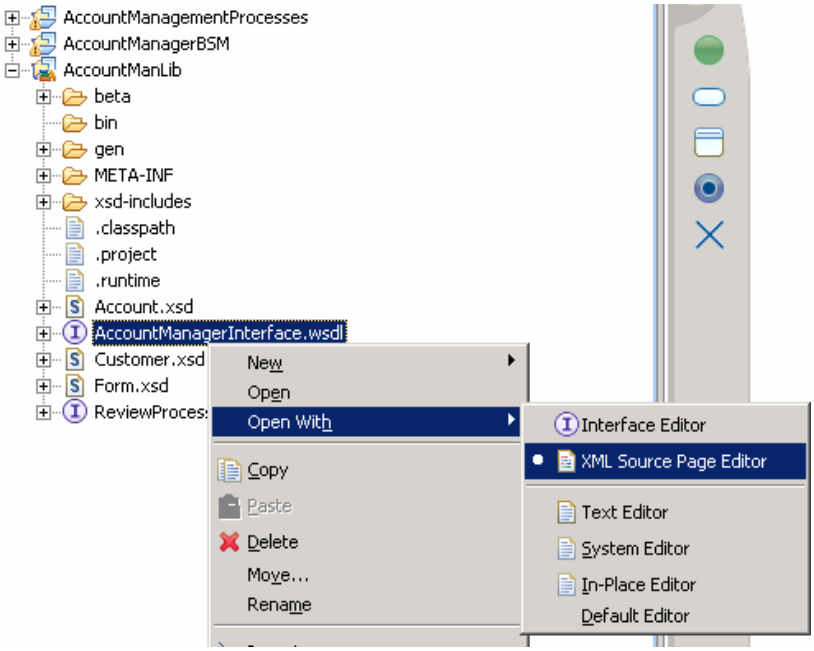
Before the state transition design diagram is modified, the AccountManagerInterface.wsdl file will need to be edited. The *apply* and *verify* operations in the interface have been commented out. They need to be uncommented.

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

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



__ b. Expand **AccountManLib,** and then right-click on **AccountManagerInterface.wsdl** and select **Open With > XML Source Page Editor.**



__ c. In the XML Source Page Editor, scroll to the bottom of the page. Highlight and delete the **<!-- -->** comment lines for the apply and verify operation coding.

```
<!--
    <wsdl:operation name="apply">
        <wsdl:input message="tns:applyRequestMsg" name="applyRequest"/>
        </wsdl:operation>
    <wsdl:operation name="verify">
            <wsdl:input message="tns:verifyRequestMsg" name="verifyRequest"/>
    </wsdl:operation>
-->
```

__ d. Press **Ctrl+S** to save.

____ 3. Assign the new correlations for use in the AccountManagerBSM.

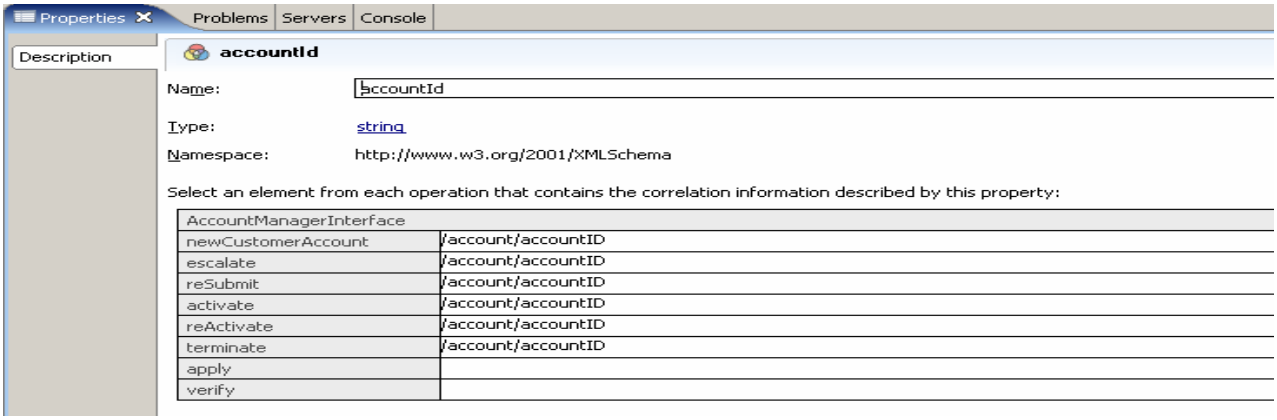
__ a. Switch back to the Business Integration View; close the **Physical Resources** View (by clicking the X) located in the top left.

__ b. Expand **AccountManagerBSM -> Business Logic -> State Machines.**

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

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

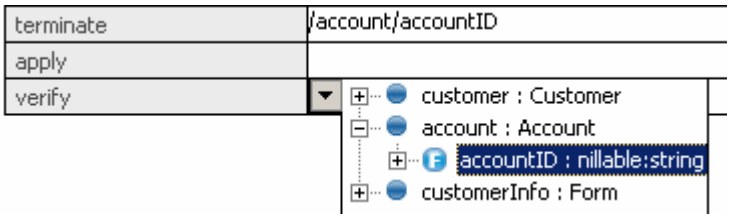
__ e. In the **Properties** tab, click on the column next to **apply** operation.



__ f. Expand **account** and select **accountID**

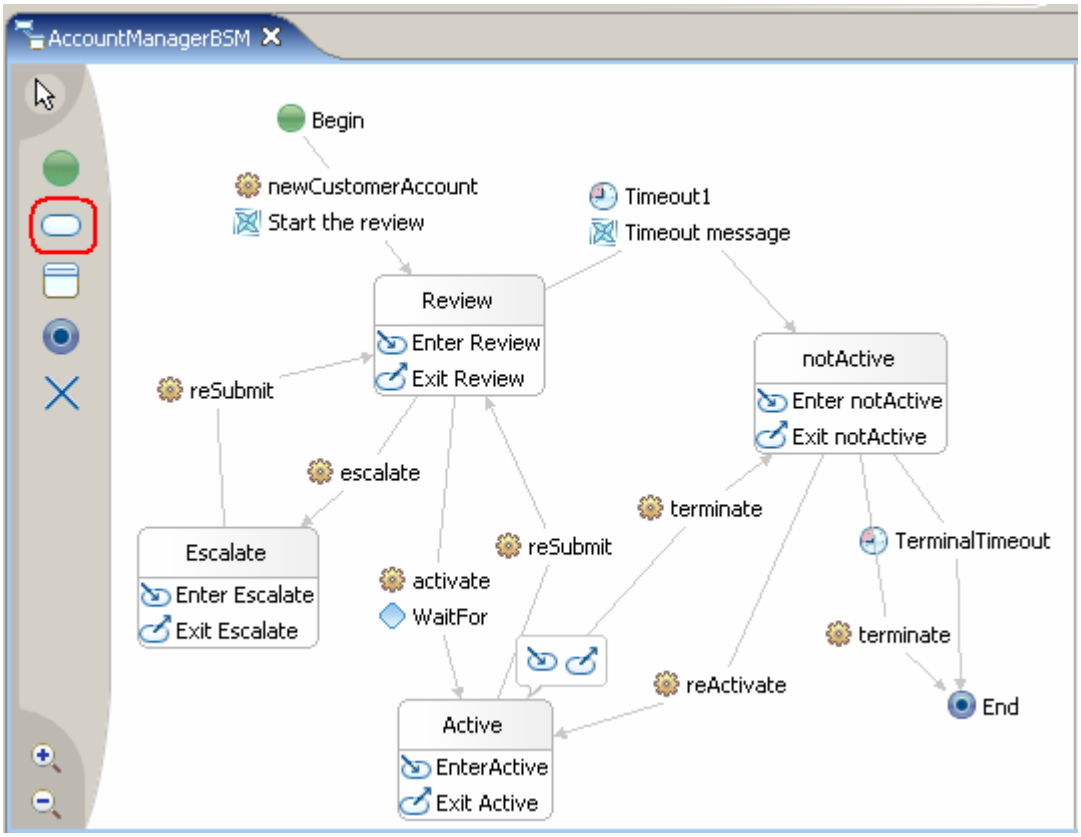
__ g. Do the same step for **verify** operation by expand **account** and click on **accountID.**

__ h. Press **Ctrl+S** to save.



____ 4. Add the ApplicationPending state to the business machine.

__ a. Select the **State** icon from the palette.



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

____ 5. Add the VerificationPending state to the business machine.

__ a. Repeat step 2, naming the newly created state **VerificationPending**.

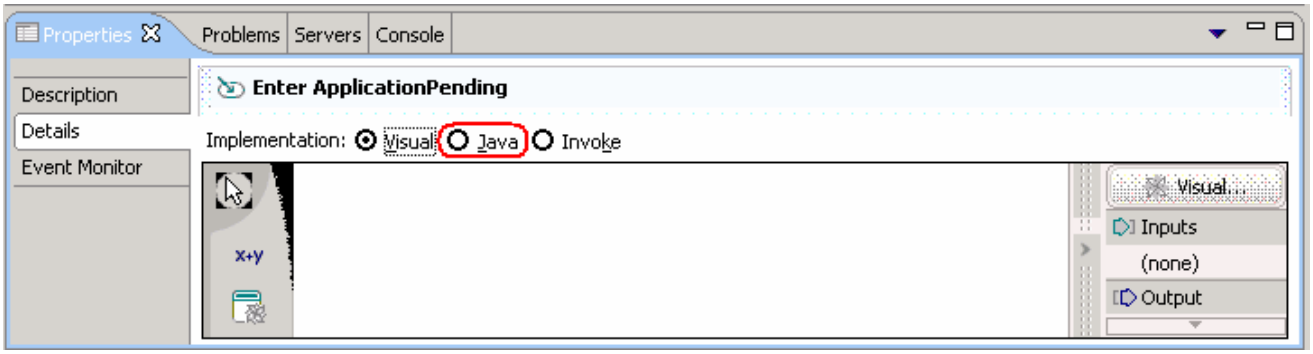
____ 6. Add the Entry and Exit actions to ApplicationPending.

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

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

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

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



__ e. Select **Yes** to the dialog box that appears.

__ f. Enter the following code:

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

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

__ h. Rename the exit **Exit ApplicationPending**.

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

__ j. Select **Yes** on the dialog box that appears.

__ k. Enter the following code:

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

____ 7. Add the entry and exit for VerificationPending.

__ a. Repeat step 4 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 ***");
```

____ 8. 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");
```

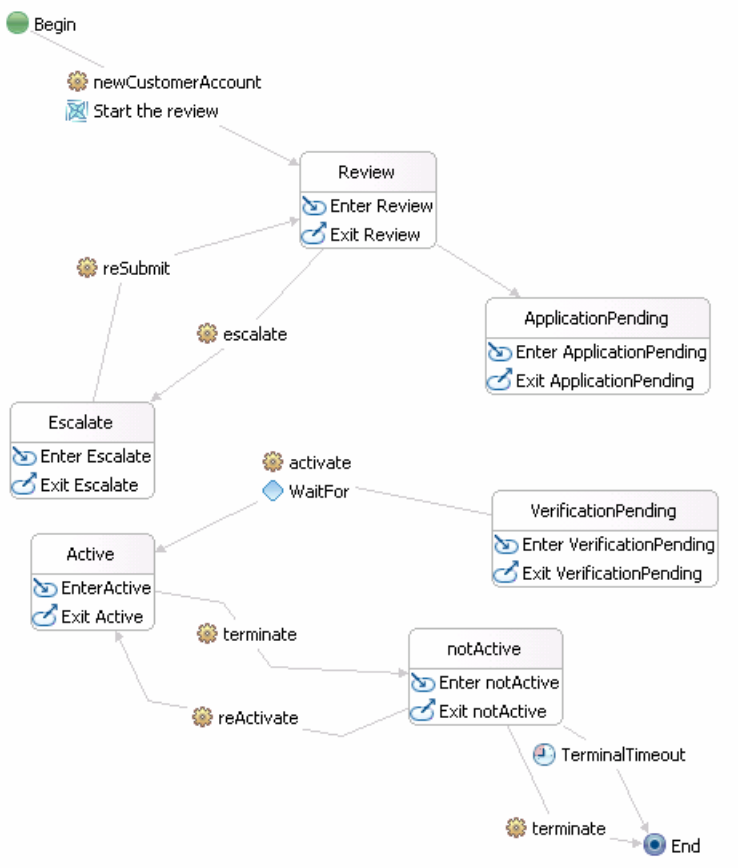
____ 9. Arrange the transitions so that you go from **Review** to **ApplicationPending** to **VerificationPending** to **Activate**. (See the diagram at the beginning of this section.)

__ 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.

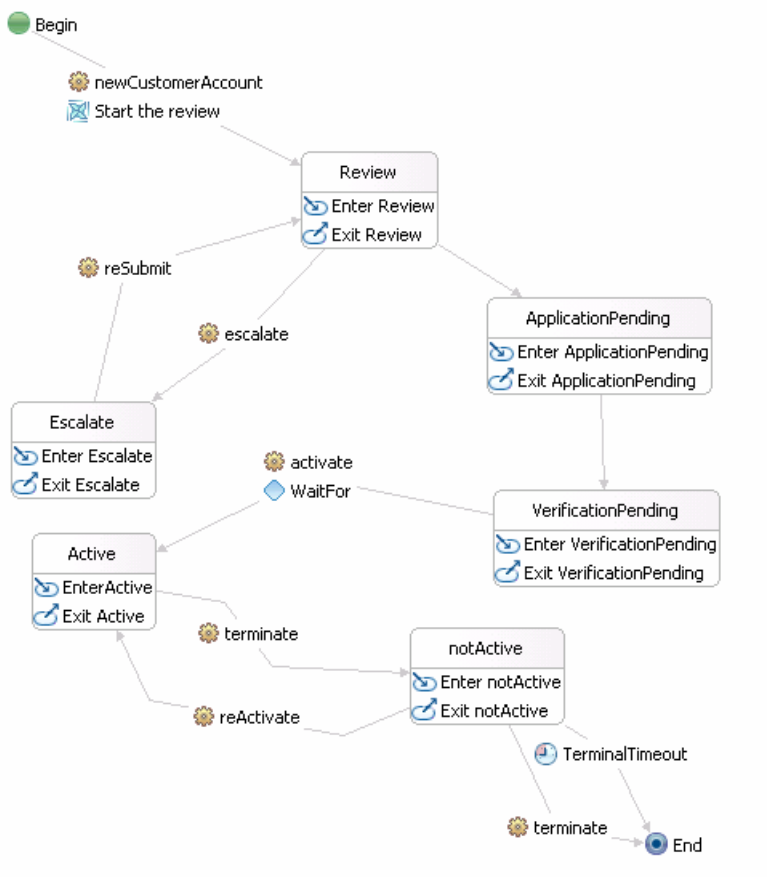
__ c. Select the wire for the reSubmit operation connecting the **Review** and **Active** states; press the **delete** key to remove it.

__ 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:

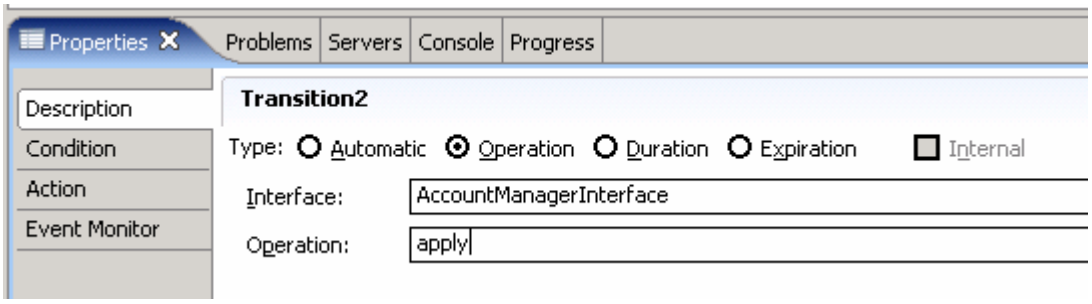


____ 10. Create operations for the new transitions.

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

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

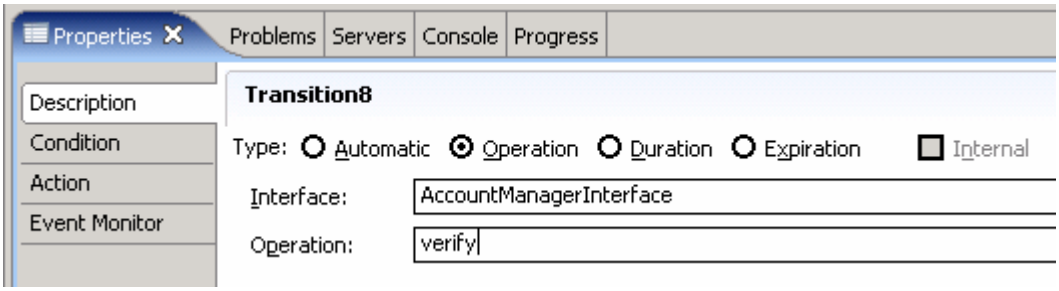
__ c. Set the Type to **Operation**, the Interface to **AccountManagerInterface**, and the Operation to **apply**.



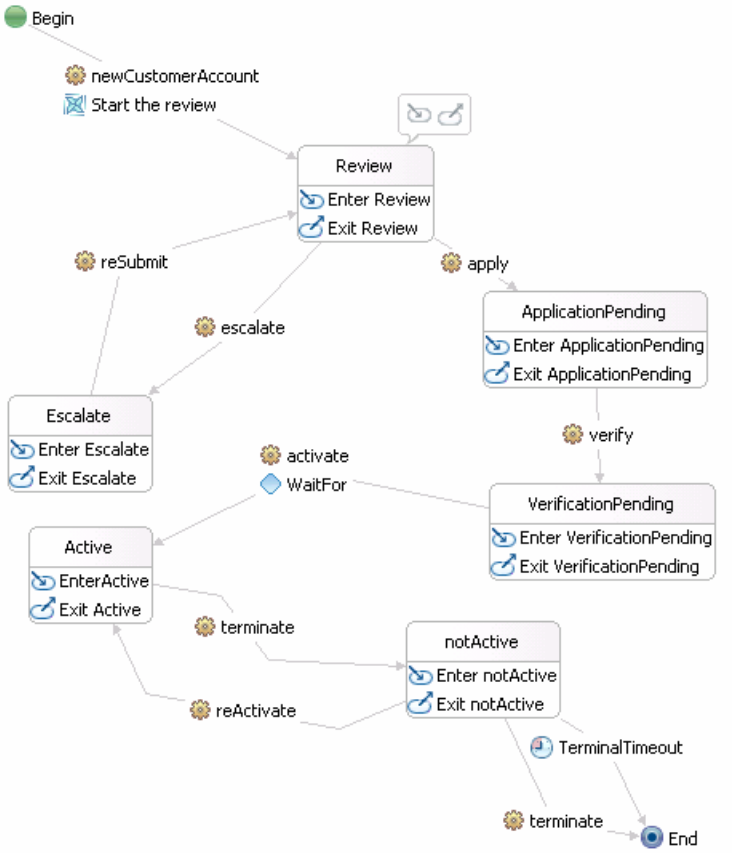
__ d. Select the transition connecting **ApplicationPending** to **VerificationPending**.

__ 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:



____ 11. 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");
```

____ 12. 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**.

____ 13. 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

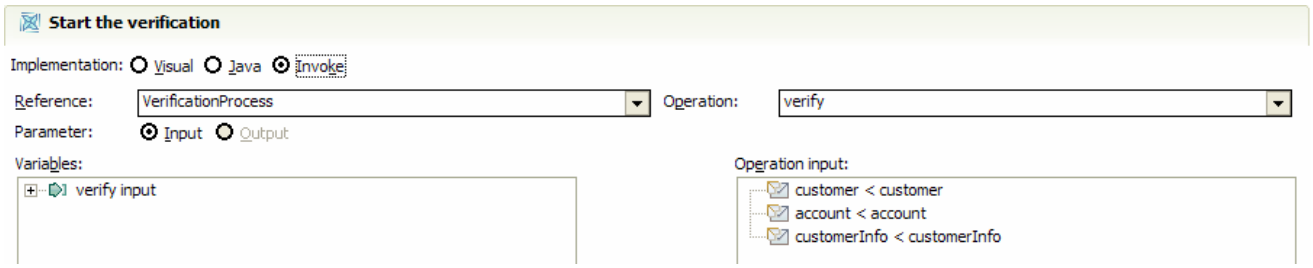
2) Select **VerificationProcess** for Reference and **verify** for Operation

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

4) Click on **Set >**

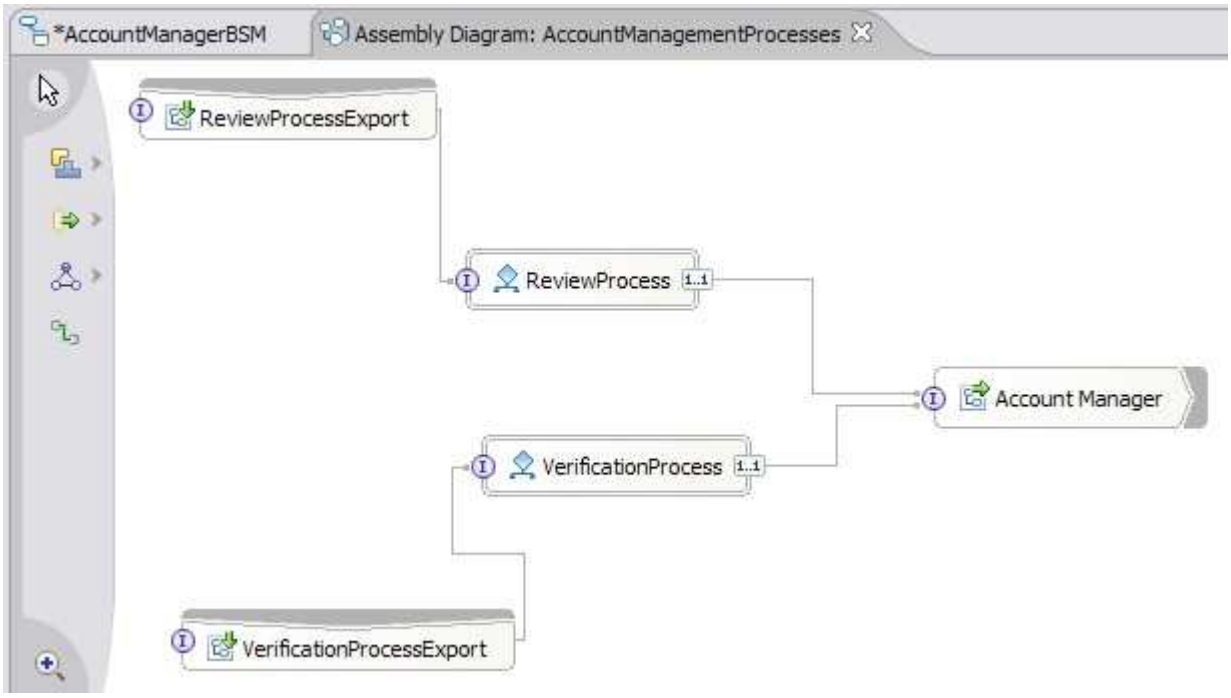
5) Repeat the step 3-4 for **account** and **customerInfo**

6) Save the changes



__ 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/



____ 14. 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.

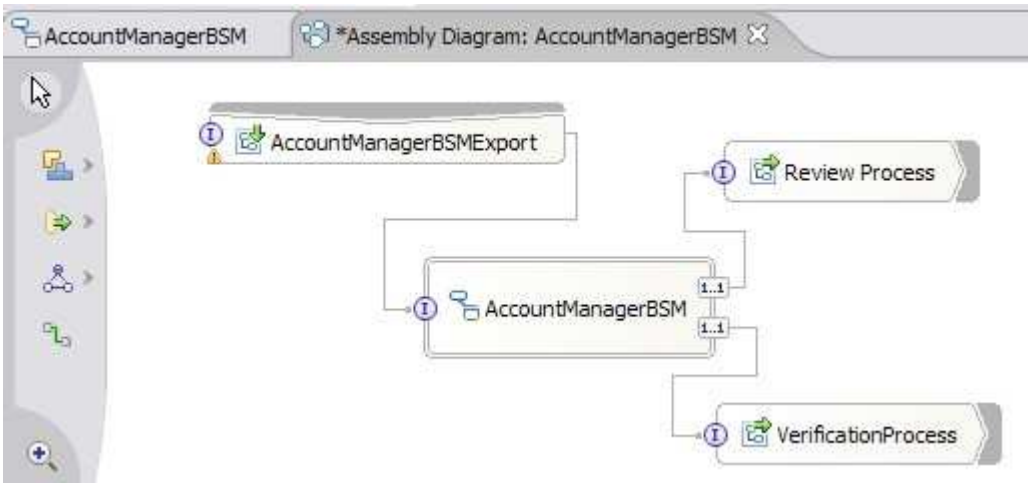
2) Drag the **VerificationProcessExport** from the **AccountManagementProcesses** module and drop it onto the assembly diagram and select "**import with SCA bindings**" when prompted.

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 2 references on the AccountManagrerBSM component.



6) Save and run Clean the project.

Test the changes that were made.

____ 15. 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.**

____ 16. Start the Component Tester on the AccountManagerBSM module.

__ a. Right click on the AccountManagerBSM module.

__ b. Select **Test > Test Module.**

__ c. If using a remote testing environment, Select **Project-> Properties -> Integration Test Client.** Notice 'Always use the default target in the test client.' box is checked. Uncheck it and select **OK**. You'll be prompted for your deploy location now when testing.

____ 17. Run the tests

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

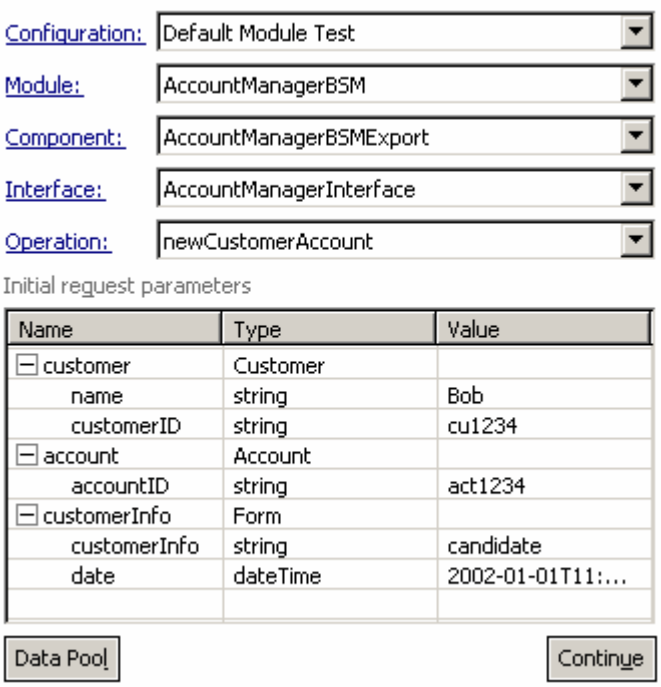
__ a. Add 4 addition invokes.

1) Click on the **Invoke** button 4 times; there should be 5 total invokes in under Events.

__ b. Set the component to **AccountManagerBSMExport** for each invoke.

__ 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 the following output displayed to the console:

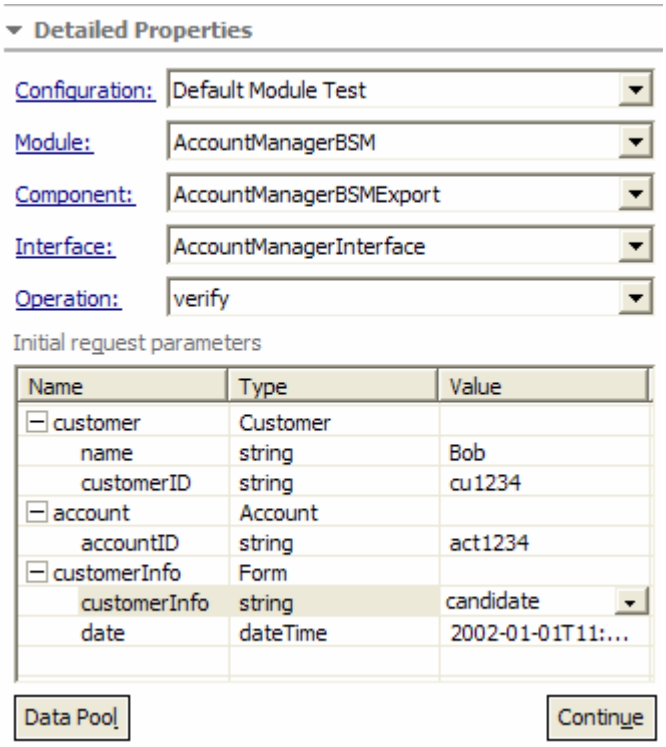
```
*** Entering the Review state
***** Review Process *****
*** We have a regular customer
*** Review signed off at: Fri Apr 20 20:26:27 EST 2007
*** Leaving the Review state
*** Entering the ApplicationPending
```

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 accounted, 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 Apr 20 20:27:49 EST
2007
*** BSM Condition - signoff time Fri Apr 20:27:51 EST 2007
Signed Off: Fri Apr 20 20:28:04 EST 2007
Current:    Fri Apr 20 20:28:05 EST 2007
1131633303564
1131633304936
1372
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**

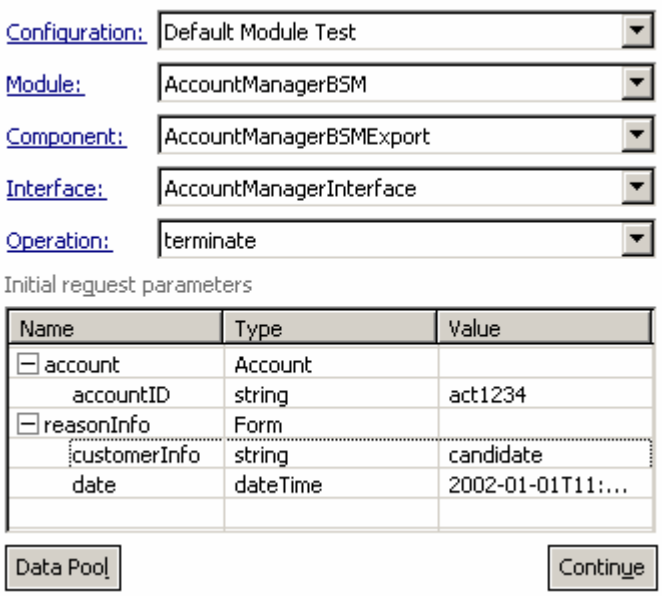
4) Select **Continue** and the following should be displayed to the console:

```
*** 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 5 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.

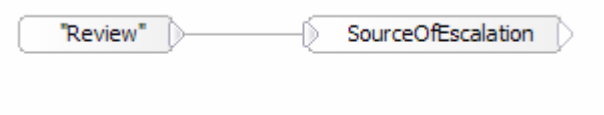
Example: If the Business State machine is in the *Review* state when the *escalate* event is received, then the value of the global variable will be set to “Review”. The value of the global variable can then be checked in each of the reSubmit transitions.

____ 18. 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**.

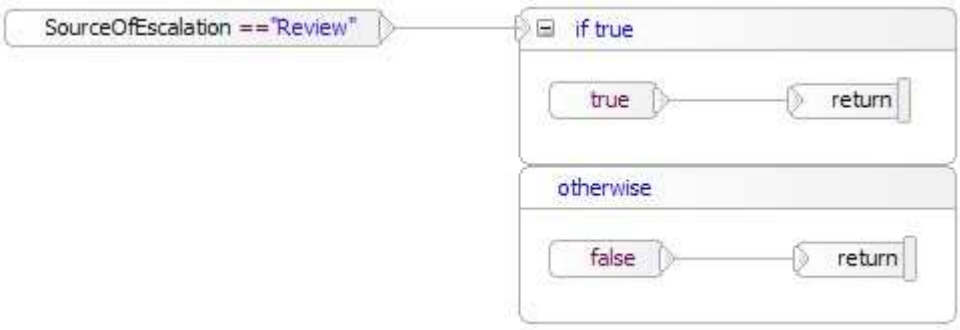
2) In the **Set Vars** action, set the **SourceOfEscalation** global variable to “Review”.



__ 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 the 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.

____ 19. Do another test cycle to verify the escalate transitions.

____ 20. Remove all applications from the server.

__ a. Change to the servers view, right-click on **WebSphere Process Server v6.0** and select **Add and remove projects…**

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

__ c. Click **Finish.**

____ 21. Stop the server.

__ a. Change to the servers view, right-click on **WebSphere Process Server v6.0** 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.

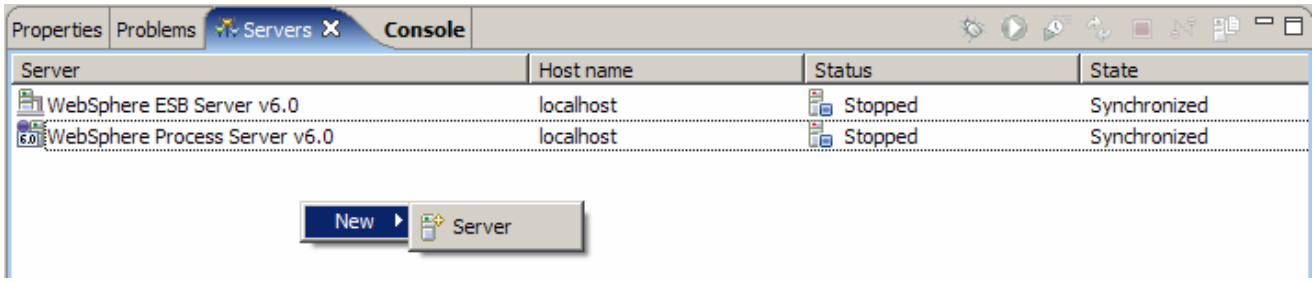
Task: Adding remote server to WebSphere Integration Developer test environment

This task describes how to add a remote server to the WebSphere Integration Developer Test environment. The sample you will use is a z/OS machine.

Create a new remote server.

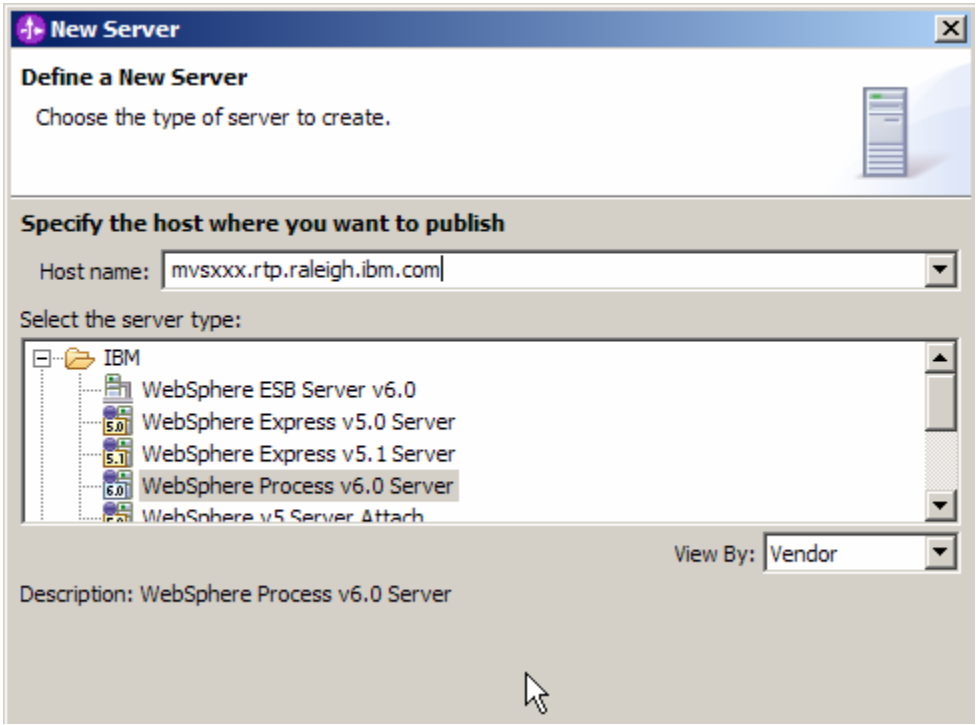
____ 1. Right click on the background of the Servers view to access the pop-up menu.

____ 2. Select **New → Server.**



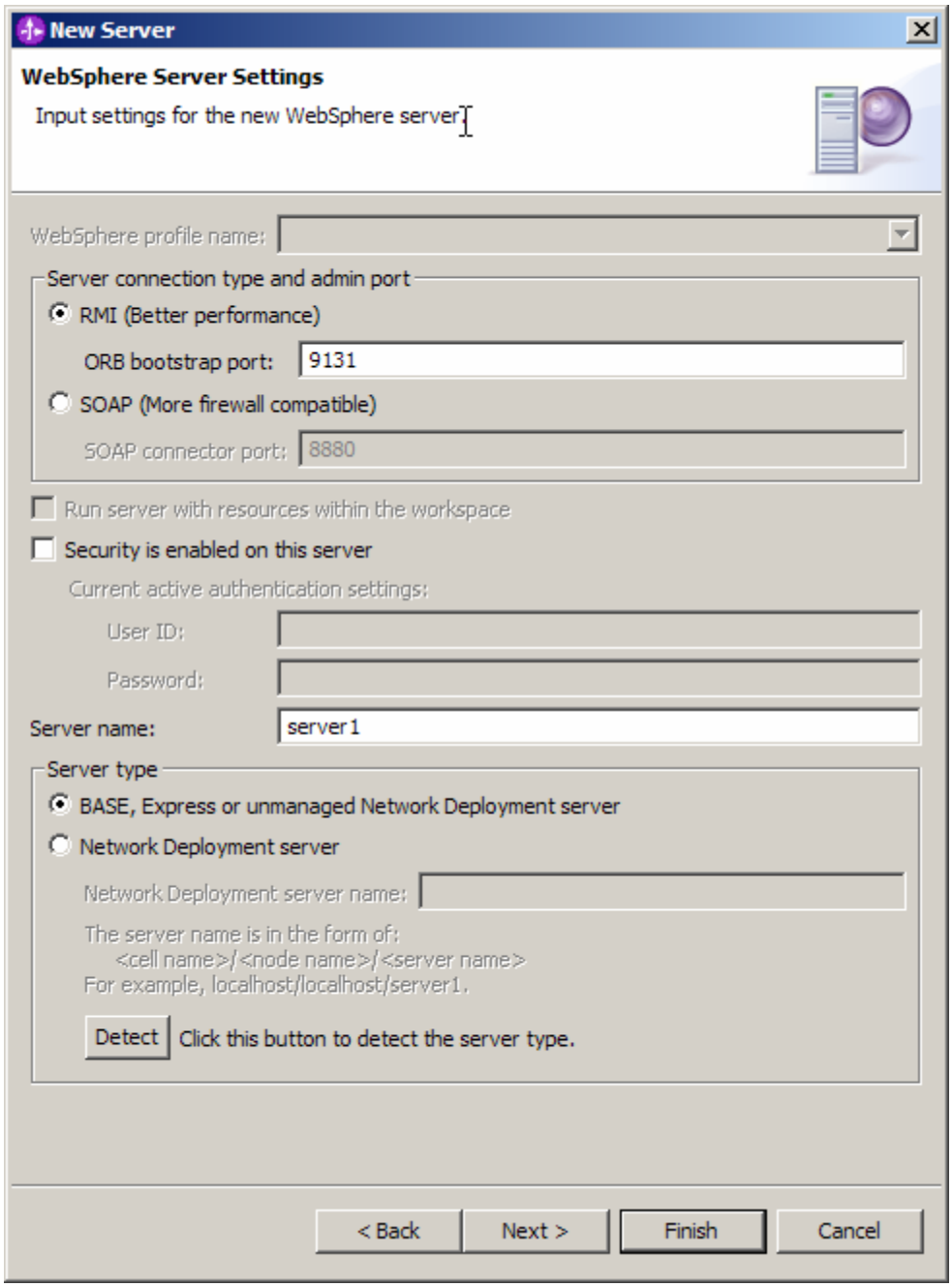
____ 3. Specify host name to the remote server, **<HOSTNAME>**.

____ 4. Ensure that '**WebSphere Process V6.0 Server**' is highlighted in the server type list.



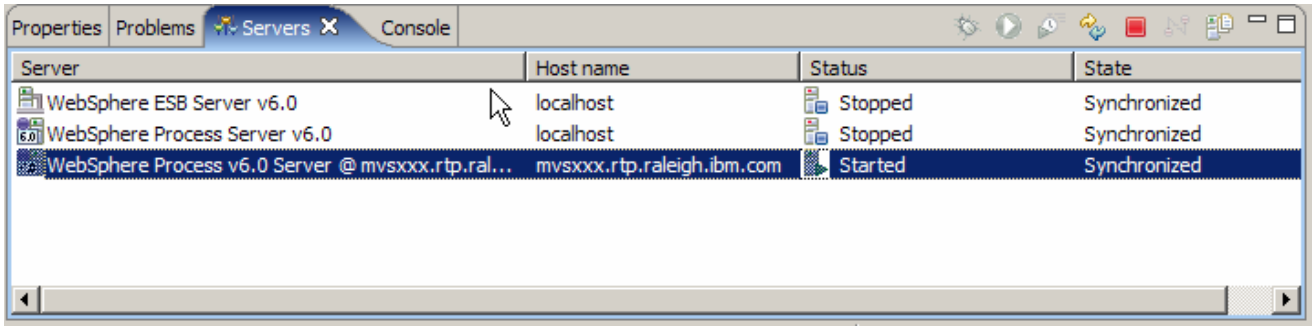
____ 5. Click **Next.**

____ 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>**'

userid : **<USERID>**

pw : **<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
```