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 lifecycle 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.

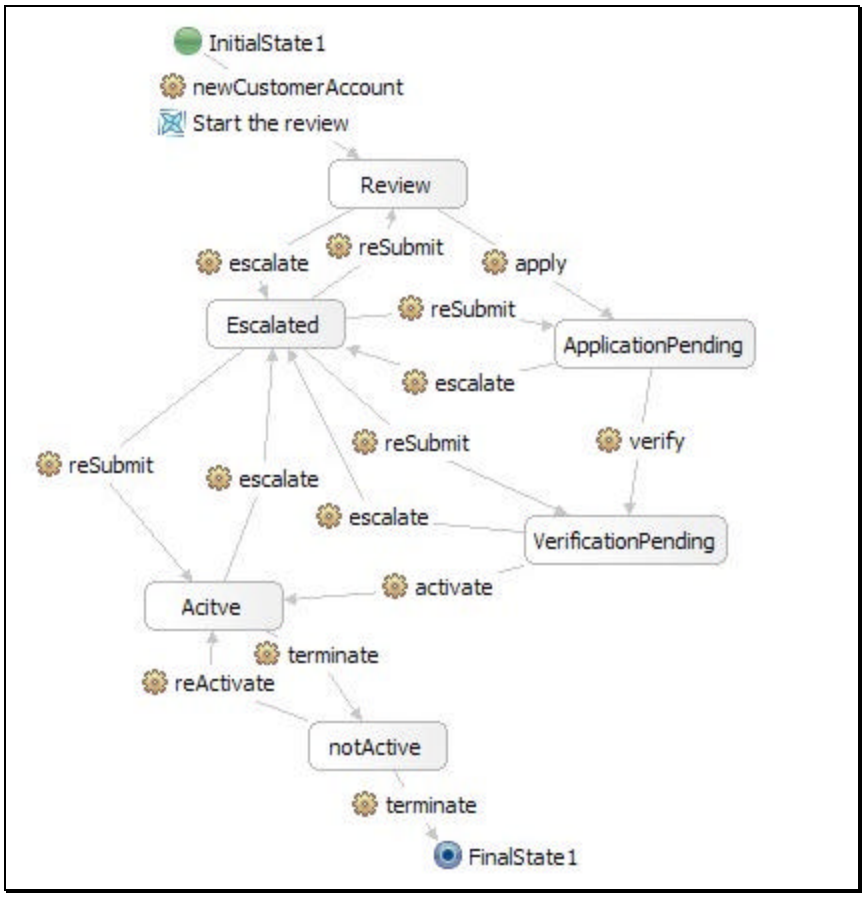
At any point in the business process the flow may be escalated to achieve greater focus and urgency or sent back to the previous step for rework.

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. (***We 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. We 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.



We will be working with a partial implementation of this application, discussing how it works and how to test it. Subsequent releases of this document will guide the student through the steps of adding the additional states, adjusting the logic and updating the SCA module assemblies.

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:\Labfiles60 (Windows) or /tmp/LabFiles60 (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 via *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 usually several phases, each of which can be defined as a BPEL business process. In this exercise you'll 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 lifecycle.

The BPEL business processes, which will be long running, will be invoked via 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 execute 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:\Labfiles60	/tmp/Labfiles60
<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:\LabFiles60\ would be replaced by C:/LabFiles60/

Part 1: Getting Started

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

<WORKSPACE>

C:\Labfiles60\BusinessStateMachines\workspace

<PROJECT_INTERCHANGE>

C:\Labfiles60\BusinessStateMachines\Snippets\AccountManager.beta.pi.zip

<MODULE>

n/a

<DEPENDENT_LIBRARIES>

n/a

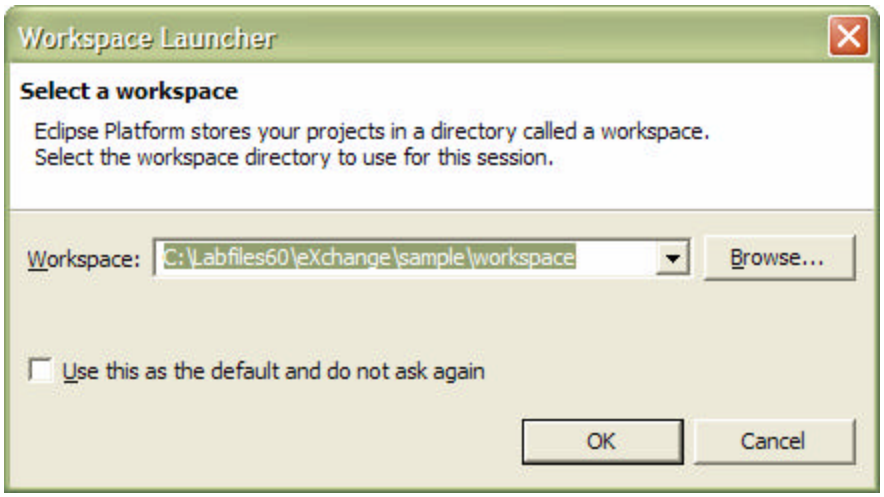
NOTE: When you first import the modules the autobuild feature will start to generate some of the artifacts for the application. Wait until the autobuild has completed before continuing.

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 we're about to do.

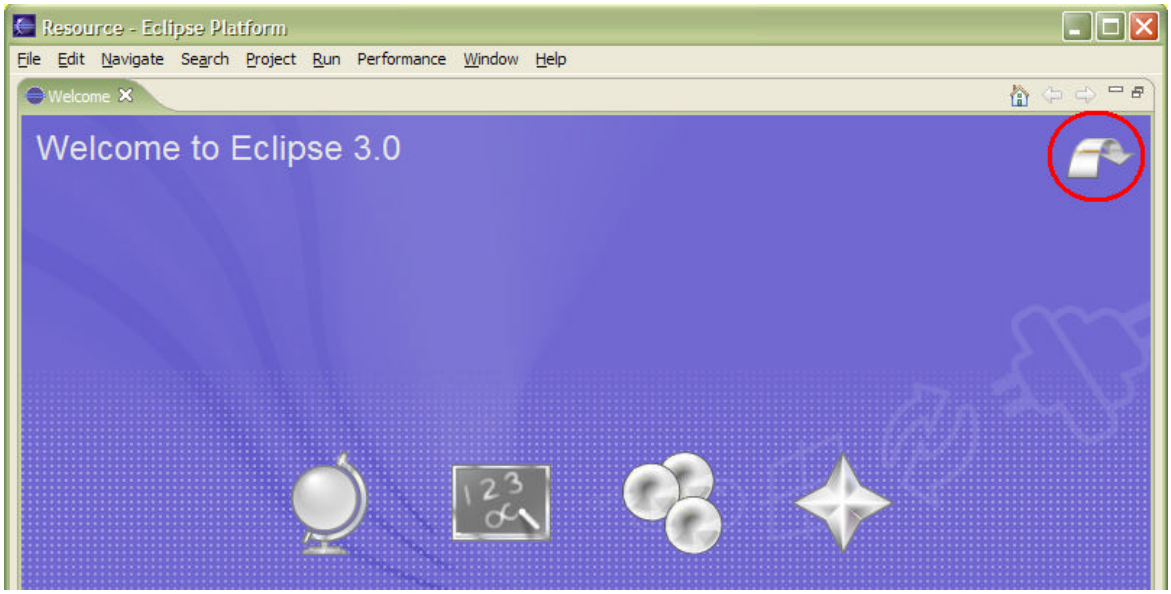
____ 2. Start WebSphere Integration Developer V6 with a new workspace located at **<WORKSPACE>**.

__ 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 V6.0 opens, close the **Welcome page** by clicking on the Go to the workbench icon (bent over arrow at top-right).

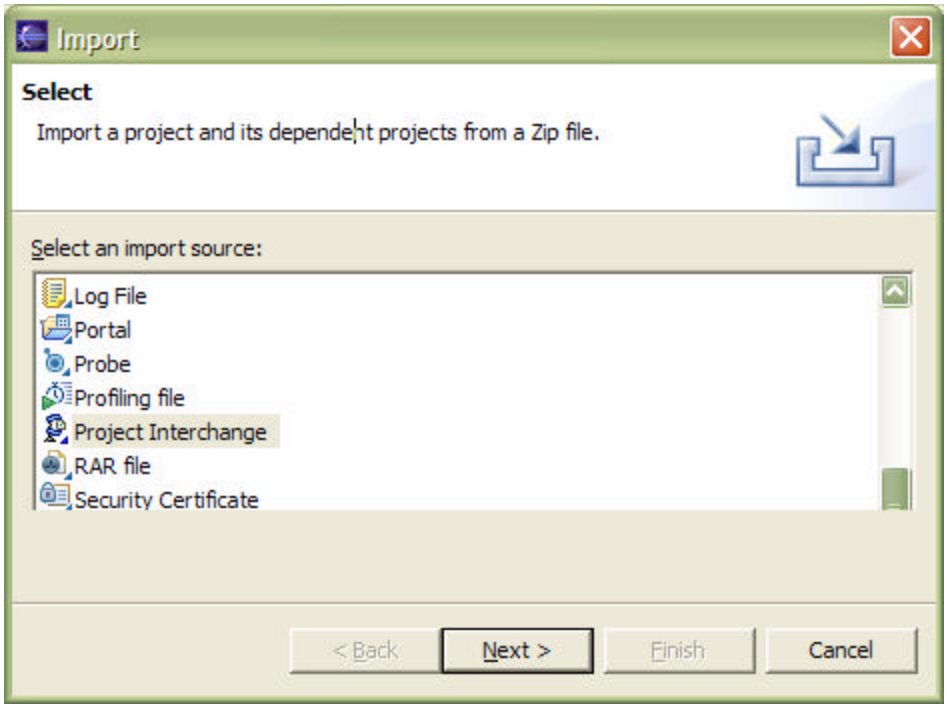


__ d. Ensure you are in the **Business Integration** perspective.

____ 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…**

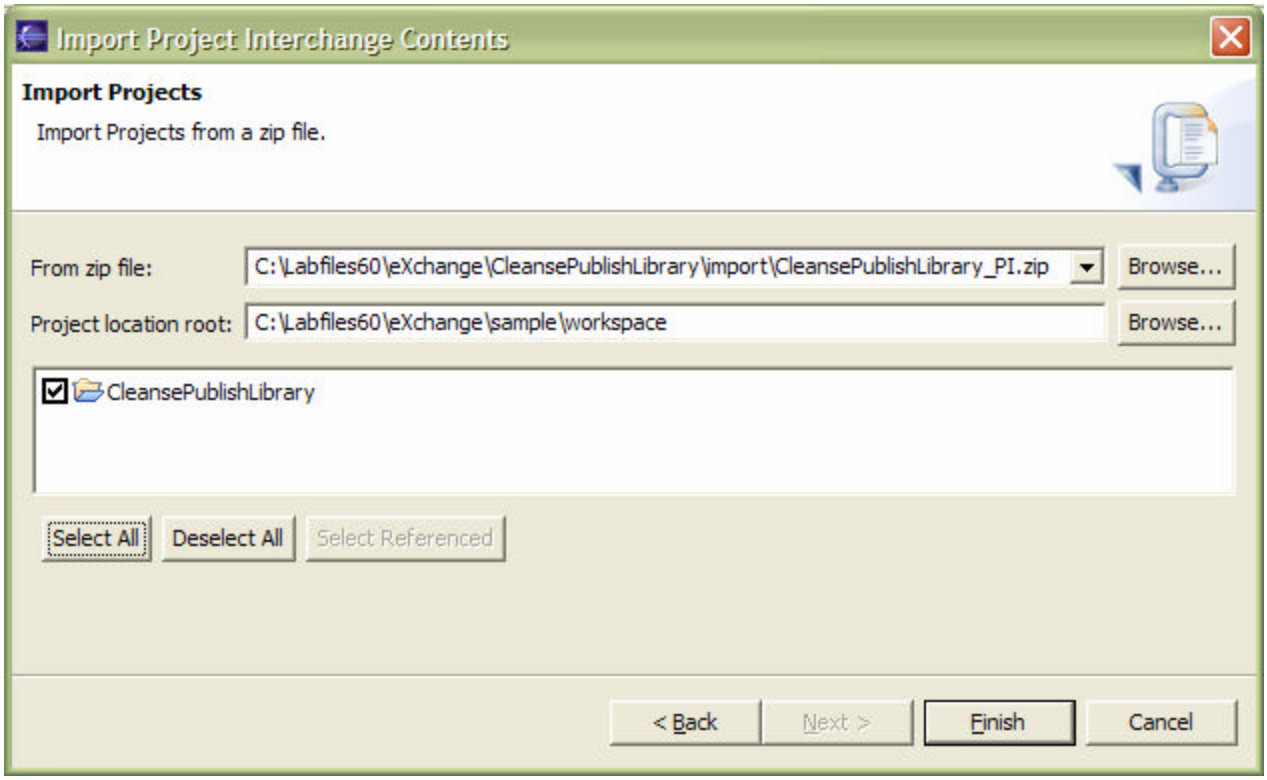
__ b. In the Import dialog, scroll down and select **Project Interchange.**



__ c. Click **Next.**

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

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

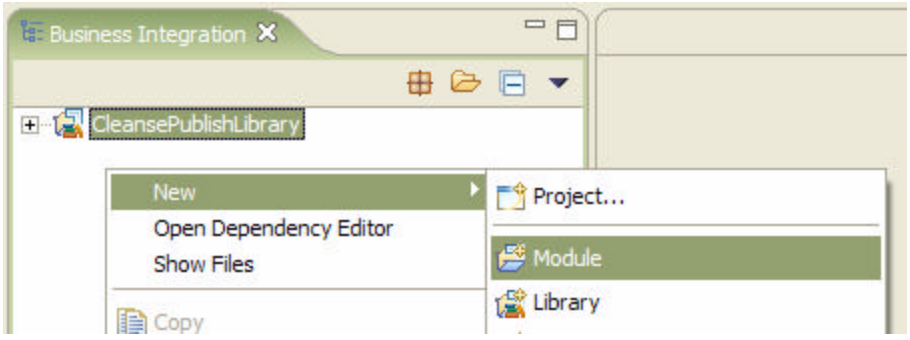


__ f. Click **Finish.**

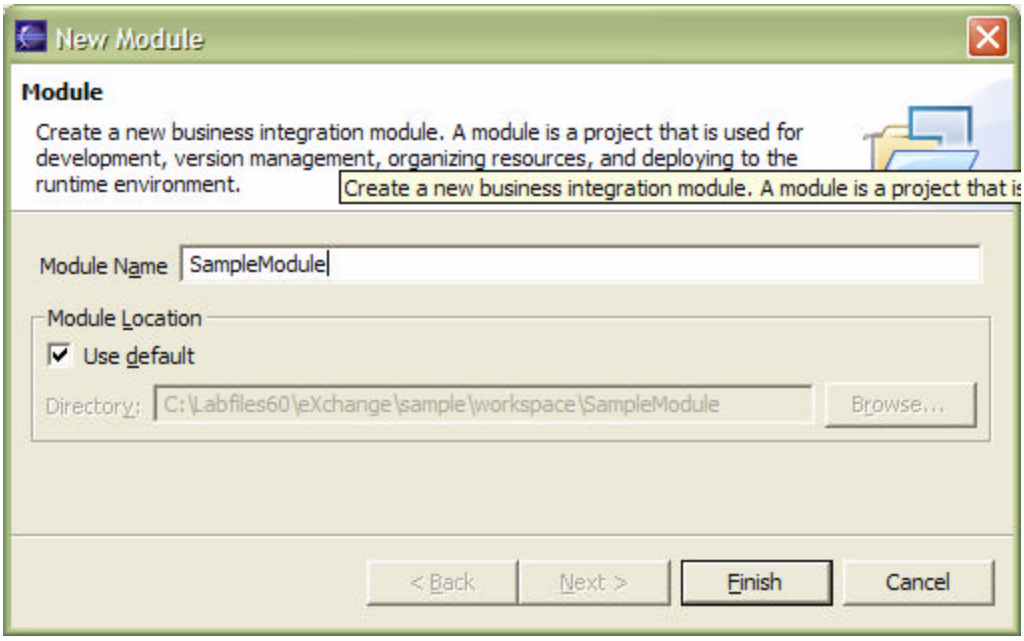
____ 4. If this lab requires that you create a Business Integration module called **<MODULE>**, complete these steps.

__ a. Right click on the background of the Business Integration view to access the pop-up menu.

__ b. Select **New > Module.**



__ c. In the New Module dialog, enter **<MODULE>** for the Module Name.

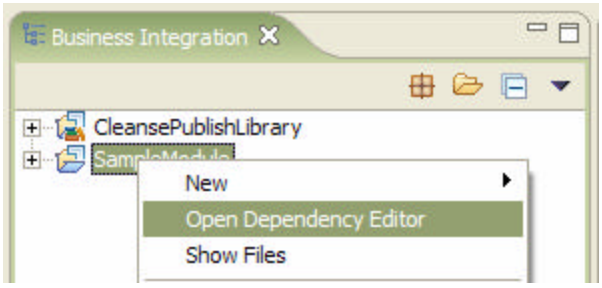


__ d. Click **Finish.**

____ 5. If this lab requires that **<MODULE>** needs any **<DEPENDENT_LIBRARIES>**, complete these steps

__ a. In the Business Integration view, right click on the **<MODULE>** you just created to access the pop-up menu.

__ b. Select **Open Dependency Editor.**



__ c. In the Dependency Editor, click the **Add…** button.

__ d. In the Library Selection dialog, select from the list the **<DEPENDENT_LIBRARIES>.**

__ e. Click **OK.**

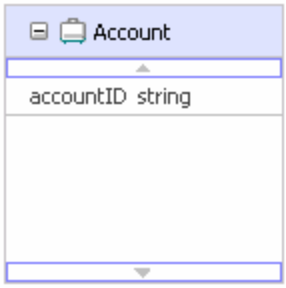
__ f. Press **Ctrl+S** to save the dependencies for this module.

____ 6. Inspect the data objects in the AccountManLib library.

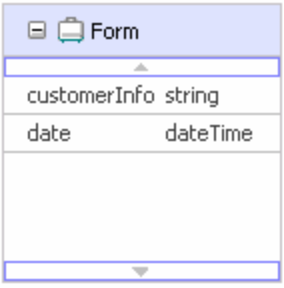
NOTE: The reporting feature will give you a nice PDF document describing the interface, objects and components.

__ a. In the Business Integration perspective, expand **AccountManLib** and **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** and **Form** business objects.



__ d. 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**.

____ 7. 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 simply invoke one of these methods on an instance of the business state machine. We will use the component test environment to do this for most of the work we'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.

ReviewProcesssInterface

ActivateProcessInterface

ApplicationProcessInterface

VerificationProcessInterface

____ 8. Inspect the business process in the AccountManagementProcess module.

__ a. Expand **AccountManagementProcess** and **Business Logic**.

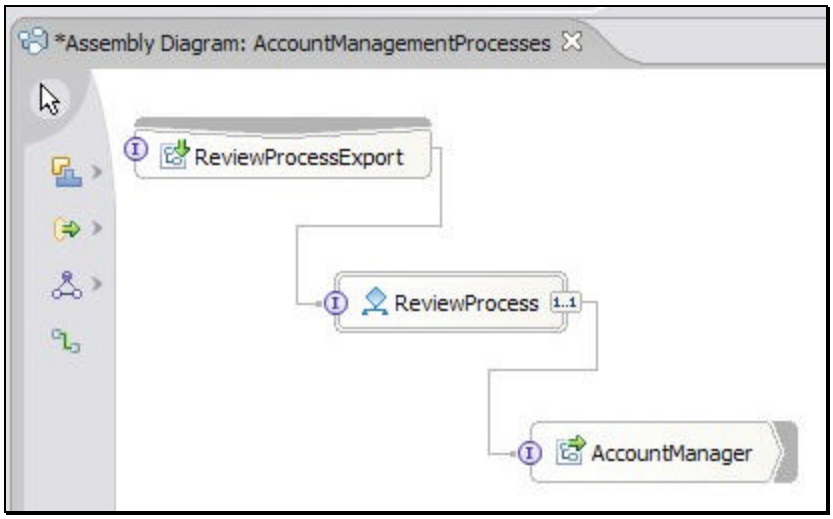
__ b. Double-click **ReviewProcess** to open the Business Process Editor.

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; we're using long running business process here 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.

____ 9. Inspect the module assemblies for AccountManagementProcess and AccountManagerBSM.

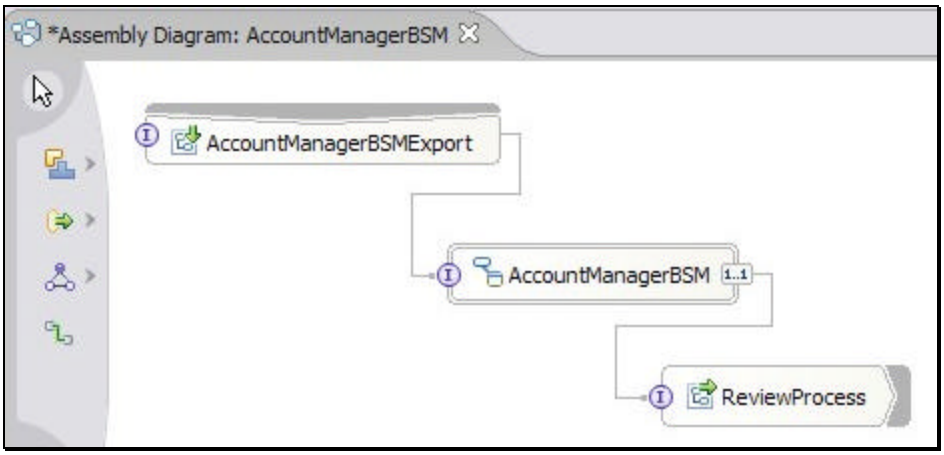
__ a. Expand **AccountManagementProcess** and double-click **AccountManagementProcess** to open the Assembly Diagram.



1) There is nothing special except the simplicity.

2) AccountManagementProcess uses SCA bindings.

__ b. Expand **AccountManagerBSM** and double-click **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. Right-click on **WebSphere Process Server v6.0**. 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 mouse click on the AccountManagerBSM module.

2) Select **Test > Test Module.**

____ 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.

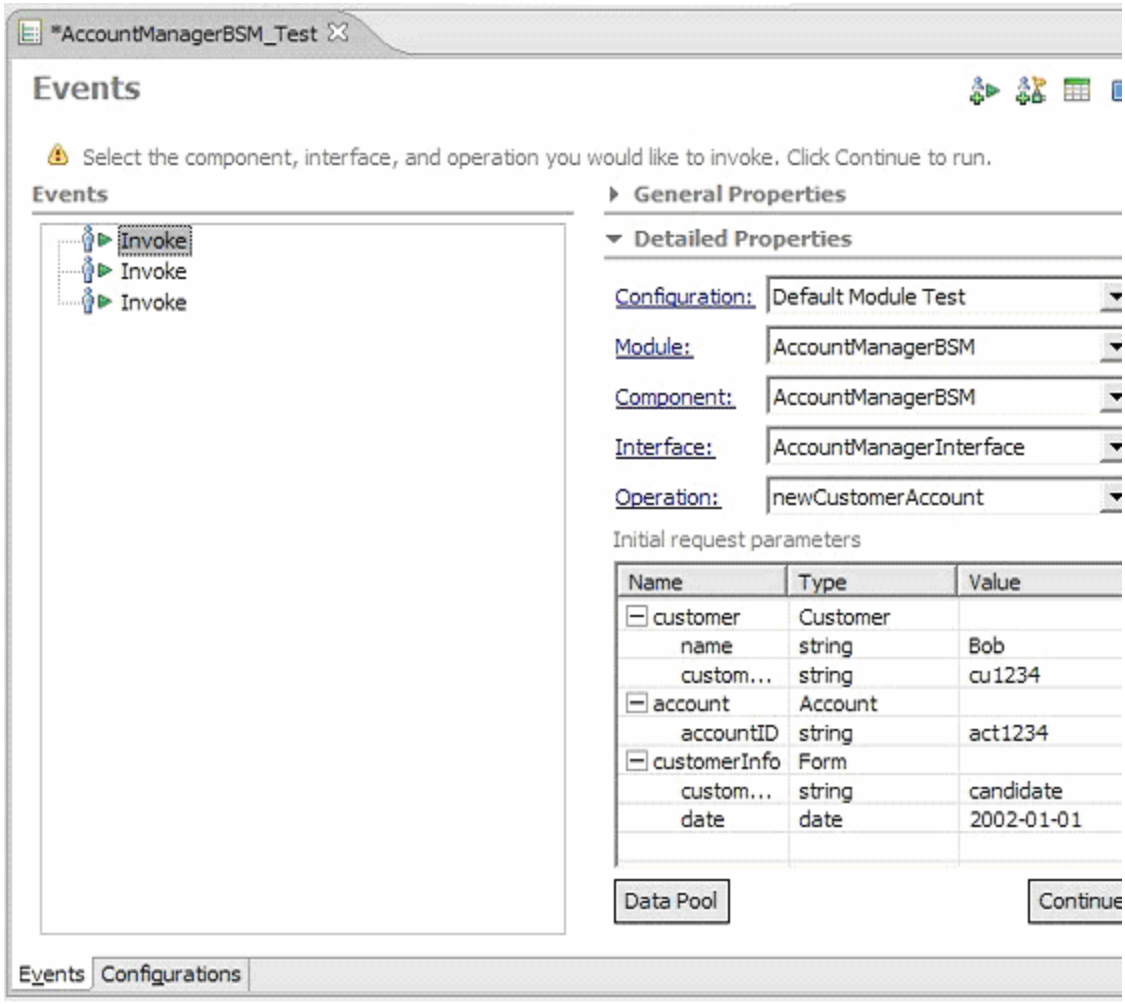
1) Click on the **Invoke** button two times; there should now be three invokes under Events.

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

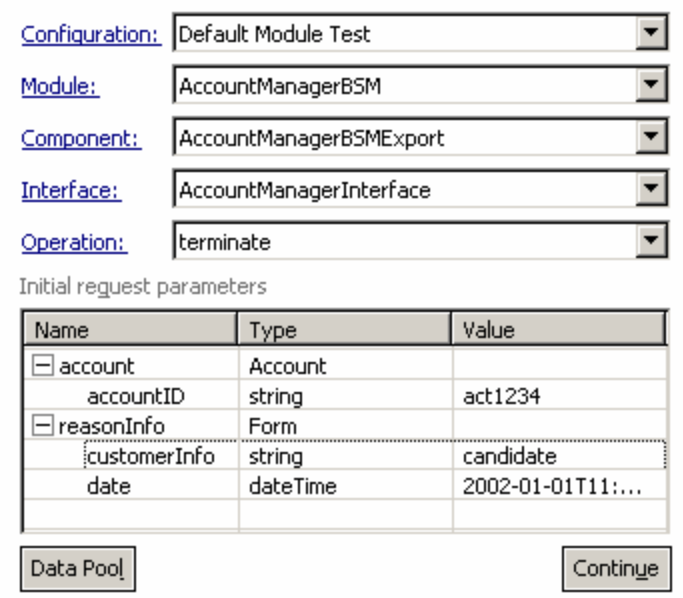
__ c. Set the input variables.

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.



__ d. Select the first invoke and press **Continue** to invoke the operation.

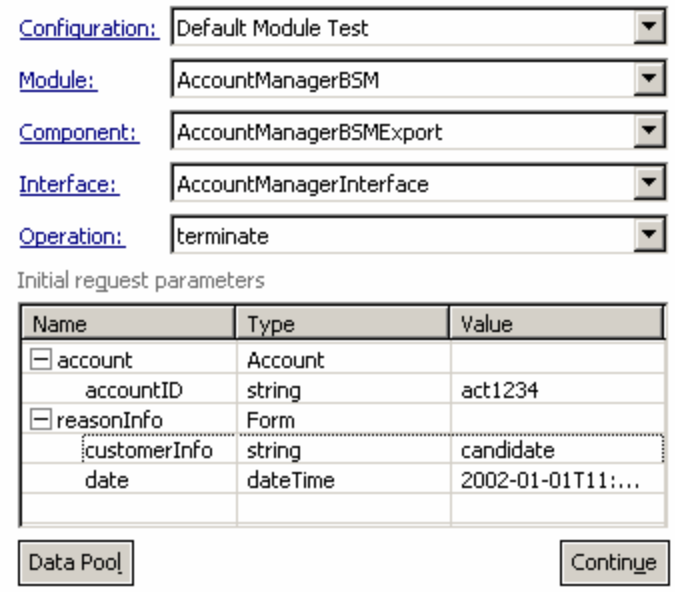
Looking at the console messages you will see that the ReviewProcess business process was kicked off and at the end it sends the *activate* event to the business state machine, which moves it to the *Review* state.

__ e. Step through the state machine using the other 2 invokes.

1) Select **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.

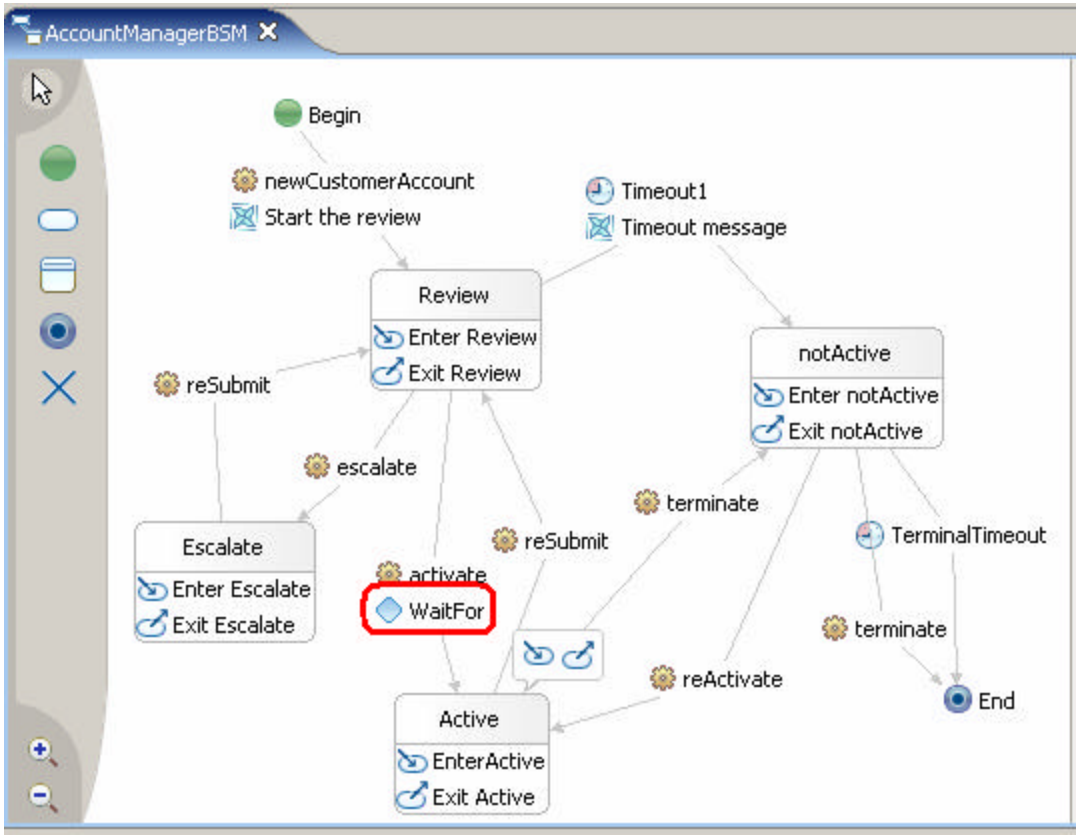
____ 1. Open the AccountManagerBSM.

__ a. Expand **AccountManagerBSM, Business Logic,** and **State Machines**.

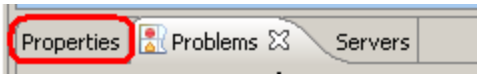
__ b. Double-click **AccountManagerBSM** to open the Business State Machine editor.

____ 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. The logic is contained in the file called **ActivateCondition.bsm.txt**.

<labfiles60>\BussinessStateMachines\Snippets\ConditionLogic.txt

__ e. Copy and paste to overwrite the code in the condition with the new code.

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

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

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

__ e. 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**.

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

```
00000065 SystemOut     O *** Waiting for period of time.
00000065 SystemOut     O *** Try again after 1 minute.
```

__ g. Wait 1 minute and then invoke the **activate** operation manually using the Component Test Client (select it and select **Continue**). The following should then be displayed in the console:

```
00000083 SystemOut     O *** Cool off period
00000083 SystemOut     O *** Leaving the Review state for: <tbd>
00000083 SystemOut     O *** Entering the Active state for: <tbd>
00000085 SystemOut     O *** Leaving the Review state for: <tbd>
00000085 SystemOut     O *** Time is up. Exiting review process
00000085 SystemOut     O *** Entering the notActive state for: <tbd>
```

__ h. If you wait five minutes, the process will automatically terminate:

```
0000008a SystemOut     O *** Leaving the notActive state for: <tbd>
```

__ i. 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.**

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.