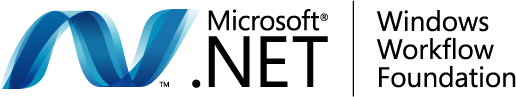
Hands-On Lab

Introduction to State Machine

Lab version: 1.0.1

Last updated: 4/22/2011

Download the [latest version](http://code.msdn.microsoft.com/Windows-Workflow-b4b808a8) of this lab



Contents

[Overview 4](#_Toc288137402)

[Starting Materials 5](#_Toc288137403)

[Exercise 1: States 5](#_Toc288137404)

[Task 0 – Open the Solution 6](#_Toc288137405)

[Task 1 – Set the Target Framework 6](#_Toc288137406)

[Task 2 – Add the Settings Argument 6](#_Toc288137407)

[Task 3 – Add the State Machine and Initialize State 7](#_Toc288137408)

[Task 4 – Add the Insert Card State 9](#_Toc288137409)

[Task 5 – Add the Insert Card Prompt 10](#_Toc288137410)

[Task 6 – Configure the Transition 10](#_Toc288137411)

[Task 7 – Add the Final State 11](#_Toc288137412)

[*Exercise 1 Verification* 11](#_Toc288137413)

[Exercise 2: Card Inserted Scenario 13](#_Toc288137414)

[Task 0 – Open the Solution 14](#_Toc288137415)

[Task 1 – Add the New States 14](#_Toc288137416)

[Task 2- Create the Transition from Insert Card to Enter PIN 15](#_Toc288137417)

[Task 3 – Create the CardResult Variable 15](#_Toc288137418)

[Task 4 – Implement the Card Valid trigger 16](#_Toc288137419)

[Task 5 – Create a Shared Trigger for an Invalid Card 17](#_Toc288137420)

[Task 6 – Configure the Card Invalid Trigger 18](#_Toc288137421)

[Task 7 - Configure the Power Off Transition 19](#_Toc288137422)

[Task 8 - Create the Card Removed transition 20](#_Toc288137423)

[Task 9 – Create the PIN Entered Transition 20](#_Toc288137424)

[*Exercise 2 Verification* 21](#_Toc288137425)

[Exercise 3: PIN Entry 23](#_Toc288137426)

[Task 0 – Open the Solution 24](#_Toc288137427)

[Task 1 – Add the Main Menu State 24](#_Toc288137428)

[Task 2 – Change the PIN Entered Transition 24](#_Toc288137429)

[Task 3 – Add Cancel Transition from Enter PIN 25](#_Toc288137430)

[Task 4 – Create the Remove Prompt Variable 26](#_Toc288137431)

[Task 5 – Initialize and Use the Remove Prompt Variable 26](#_Toc288137432)

[Task 6 – Add Timeout Transition from Enter PIN 27](#_Toc288137433)

[*Exercise 3 Verification* 28](#_Toc288137434)

[Exercise 4: Implement the Main Menu 31](#_Toc288137435)

[Task 0 – Open the Solution 32](#_Toc288137436)

[Task 1 – Add the Inner State Machine 32](#_Toc288137437)

[Task 2 – Configure the Transaction Menu Initial State 32](#_Toc288137438)

[Task 3 – Add the Final State 33](#_Toc288137439)

[Task 4 – Add the Timeout Transition 33](#_Toc288137440)

[Task 5 – Add the Withdraw State 33](#_Toc288137441)

[Task 6 – Add the Deposit State 34](#_Toc288137442)

[Task 7 – Add Null Transitions from Withdraw and Deposit to Exit 34](#_Toc288137443)

[*Exercise 4 Verification* 35](#_Toc288137444)

[Summary 37](#_Toc288137445)

Overview

1. In this lab, you will learn how to build workflows using Windows Workflow Foundation and the State Machine activity.
   1. ** What is a State Machine?**
   2. **"**It is a behavior model composed of a finite number of [states](http://en.wikipedia.org/wiki/State_(computer_science)), transitions between those states, and actions, similar to a flow graph in which one can inspect the way [logic](http://en.wikipedia.org/wiki/Logic) runs when certain conditions are met."
   3. **- Wikipedia –** [**Finite State Machine**](http://en.wikipedia.org/wiki/Finite-state_machine)
2. In Windows Workflow Foundation (WF4) a State Machine is a workflow activity that is best for scenarios where the workflow responds to events outside of itself. Throughout this lab, you will implement a State Machine that will direct the flow of the user interface for an Automated Teller Machine (ATM). The UI designers have done their analysis and come up with a set of scenarios that you will implement using the State Machine activity.
   1. ****The lab also includes pre-written code that implements a WPF UI using the Model-View-ViewModel pattern and unit tests to verify the behavior of the State Machine. Feel free to explore the Solution to learn how you can implement this pattern in your applications.
   2. Estimated time to complete this lab: **45 minutes per exercise**.
   3. ** Do I have to do all the exercises?**
   4. This lab is long. To do all of it will probably take several hours. Each exercise is independent of the others so you can do them one at a time. Just open the solution found under the Begin folder for the exercise. For example, if you did Exercise 1 yesterday, today you would start with Exercise 2 and open the solution from the **Source\Exercise 2\Begin** folder

# Objectives

* + Understand the basics of the State Machine, States and Transitions
  + Create Entry and Exit Actions
  + Use Transitions and Triggers including Shared Triggers and Null Triggers

# Pre-Requisites

* 1. This lab assumes basic knowledge of Windows Workflow Foundation (WF4). If you are new to WF4, we suggest you complete the Introduction to WF4 Hands on Lab prior to this lab.

# System Requirements

* 1. You must have the following items to complete this lab:
  + [Microsoft Visual Studio 2010 Service Pack 1](http://www.microsoft.com/downloads/en/details.aspx?FamilyID=75568aa6-8107-475d-948a-ef22627e57a5).
  + [Microsoft .NET Framework 4 Platform Update 1 – Design-time Package for Visual Studio 2010 SP1 (KB2495593)](http://support.microsoft.com/kb/2495593)

### Starting Materials

* 1. This Hands-On Lab includes the following starting materials.
  + **Visual Studio Solutions.** The lab provides the following Visual Studio Solutions that you can use as starting point for the exercises.
  + **Assets.** This lab includes activities, unit tests and a WPF application that will use the state machine you build.
    1. **** Inside each exercise folder, you will find an **end** folder containing a Solution with the completed lab exercise.

Exercise 1: States

|  |  |
| --- | --- |
| C:\Users\rojacobs\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.IE5\DF10ZJWG\MC900431606[1].png | [Watch Exercise 1 on endpoint.tv](http://channel9.msdn.com/Shows/Endpoint/endpointtv-WF4-State-Machine-Hands-On-Lab-Exercise-1) |

* 1. In this exercise, you will implement the first scenario for the ATM machine.

|  |  |  |
| --- | --- | --- |
| Given | When | Then |
| A Powered Off ATM | The Power Is Turned On | The ATM Initializes the hardware and displays a please wait message. After initialization it prompts the user to insert a card |

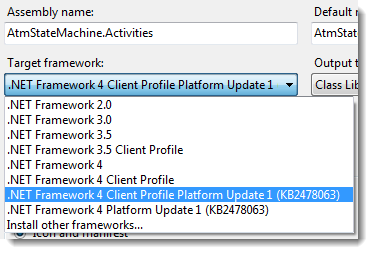
* 1. The first task when implementing the scenario is to determine the set of states. You can add more states as you go but it is a good idea to spend some time thinking about the states that you want before you start creating the workflow. In the first scenario there are three states which occur in the following order

|  |  |
| --- | --- |
| State | Description |
| Power Off | The ATM begins in the PowerOff state. This state will not be included in the state machine. |
| Initialize | The ATM is initializing |
| Insert Card | The ATM is waiting for the user to insert a card |

Task 0 – Open the Solution

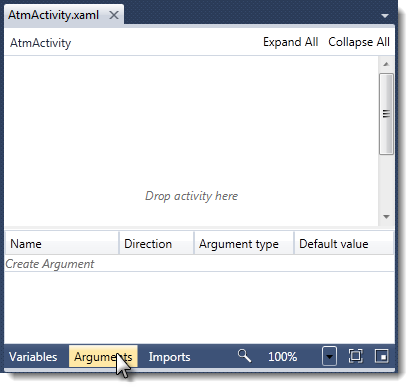
* 1. Open the begin Solution under **(Lab Folder)\Exercise 1\Begin\AtmStateMachine.sln**
  2. Build the Solution

Task 1 – Observe the Target Framework

* 1. Right click on the **AtmStateMachine.Activities** project and select **Properties**
  2. Click on the **Application** tab
  3. Notice that the project's target framework is **.NET Framework 4 Client Profile Platform Update 1 (KB2478063)** 
     1. 
     2. **** To use the State Machine you must set the Target Framework to .NET Framework 4 Platform Update 1 (KB2478063) or .NET Framework 4 Client Profile Platform Update 1 (KB2478063).

Task 2 – Add the Settings Argument

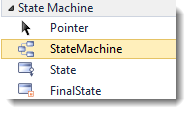
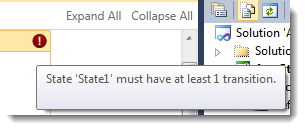
Our activities will use settings provided by the host. We need to create an argument to receive the settings.

* 1. Open **AtmActivity.xaml** from the **AtmStateMachine.Activities** project
  2. Click on **Arguments** to open the Arguments window
     1. 
  3. Click on *Create Argument* to add a new Argument

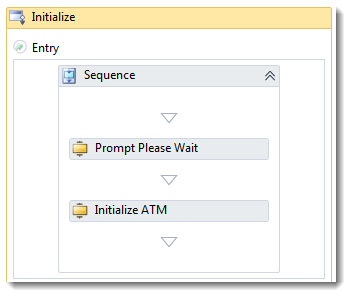
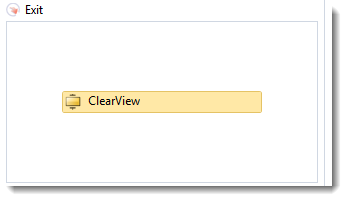
|  |  |  |  |
| --- | --- | --- | --- |
| Name | Direction | ArgumentType | Default Value |
| Settings | In | AtmStateMachine.Activities.AtmSettings |  |

* + 1. ****In the type drop down select **Browse for Types…** to open the type selector dialog. Then type **AtmSettings** to locate the type.

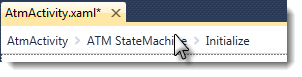
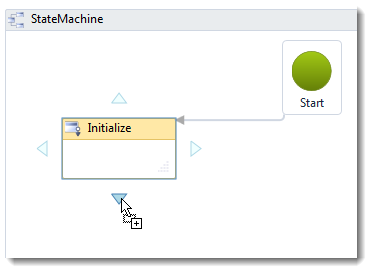
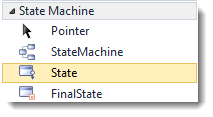
Task 3 – Add the State Machine and Initialize State

* 1. Drop a **StateMachine** onto the Workflow Designer
     1. 
     2. ** What if there is no State Machine group?**
     3. Make sure you have installed the **Multi-Targeting Pack for Microsoft .NET Framework 4 Platform Update 1 (KB2495638)** and set the target framework for the **AtmStateMachine.Activities** project (Task 1)
     4. **What Should I do about the Validation error?**
     5. 
     6. You will correct this error as you complete the exercise.
  2. Change the **DisplayName** of the **StateMachine** to **ATM StateMachine**
     1. **** The Unit Tests and UI for this lab rely on the **DisplayName** property of both states and the State Machine to be a known value. Be sure to enter the **DisplayName** exactly as shown.
  3. Select the **State1** activity and set the **DisplayName** property to **Initialize**.Thisis the first state of our state machine.
     1. **State Entry / Exit Actions**
     2. When the ATM Machine enters the Initializing state, the scenario says it must initialize the hardware and display a "Please Wait" prompt. The State activity includes entry and exit actions where you can add activities to execute on entry and exit.
  4. Double Click on the **Initialize** State to open it.
  5. In the **Entry** pane drop a **Sequence**
  6. Drop a **Prompt** activity from the **AtmStateMachine.Activities** group on the **Sequence** andset the **Prompt** properties using the property grid**.** The **Prompt** activity will display a prompt on one of 4 lines supported by the display hardware.

|  |  |
| --- | --- |
| Property | Value |
| Display Name | Prompt Please Wait |
| Line | 2 |
| Text | Prompts.PleaseWait |
| Transition | None |

* + 2. **What is the Prompts type?**
    3. The Prompts type is generated from the **Prompts.resx** resource file which contains the localizable resources used by the ATM machine. It is a good practice to use resources rather than literal text in a workflow when localization is a concern.
  1. Drop an **InitializeAtm** activity from the **AtmStateMachine.Activities** group below the **Prompt** activity and set the Display Name to **Initialize ATM**
     1. 
  2. When exiting the **Initialize** state we want the ATM Display to clear the prompt. Drop a **ClearView** activity in the **Exit** actions pane.
     1. 

Task 4 – Add the Insert Card State

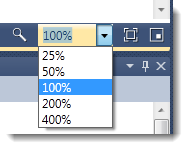
* + 1. After the ATM has initialized, the scenario calls for it to transition to the **Insert Card** state.
  1. Navigate back up to the **ATM StateMachine** by clicking on the Breadcrumb at the top of the designer window
     1. 
  2. Drop a **State** activity onto the design surface and drag it near the **Initialize State** activity. As you drag it near, arrows will appear around the **Initialize State** activity. Drop the new State activity onto the arrow that appears below the **Initialize State**. This will auto-connect the states with a transition.
     1. 
  3. Set the **DisplayName** of the new activity to **Insert Card**

Task 5 – Add the Insert Card Prompt

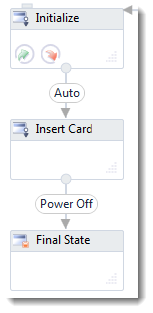
* 1. Double click the **Insert Card** state to open it
  2. Drop a **Prompt** activity to the **Entry** actions and set the properties

|  |  |
| --- | --- |
| Property | Value |
| Display Name | Prompt Insert Card |
| Line | 2 |
| Text | Prompts.InsertCard |
| Transition | None |

Task 6 – Configure the Transition

* + 1. A state is like a destination on a journey. The state machine begins in the initial state and the journey to every other state is a ***Transition***. Transitions usually involve an activity that waits for some event to occur. In this case, because the hardware initialization is completed we do not need to wait so we will use a ***Null Transition*** (atransition with no triggering action).
    2. The **DisplayName** of the transition between **Initialize** and **Insert Card** is the auto-generated value **T1**. Typically, you name the transition something that describes the event that causes the transition. In this case, because there is no event that causes the transition, name it **Auto**.
  1. Navigate back up to the **ATM StateMachine** by clicking on the Breadcrumb at the top of the designer window
  2. Select the **T1** transition by clicking on the line
     1. **** To make it easier to select the line you can zoom in the designer view using Ctrl+ Mouse Scroll Wheel or by selecting a zoom amount from the dropdown.
     2. 
  3. Change the **DisplayName** of the transition from **T1** to **Auto**.

Task 7 – Add the Final State

* + 1. State machines end with a Final State (you may have more than one Final State). In this state, you have one last chance to do something before the State Machine ends.
  1. Drop a **FinalState** activity on the arrows below the **Insert Card** state.
  2. Set the **DisplayName** of this transition to **Power Off**.
     1. 

### *Exercise 1 Verification*

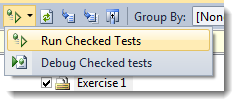
* 1. To verify that the first scenario is working correctly

#### Build the Solution

* 1. Select **Build / Build Solution**. It should compile without error.
  2. Correct any compile errors.
     1. **** In this release, Workflow validation errors will not result in build errors. Future releases of Visual Studio will have the option to generate build errors if there are Workflow validation errors.

#### Unit Test Verification

The lab includes a test project that will verify that you have correctly implemented the first scenario by running the tests for Exercise 1

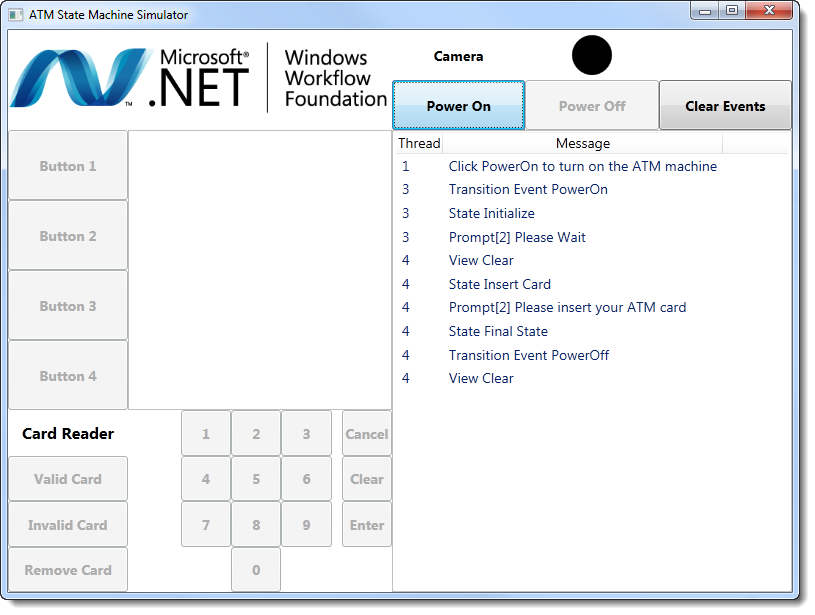
* 1. Open the **Test List Editor**. From the menu, select **Tests** / **Windows / Test List Editor.**
  2. Check the Test List named **Exercise 1**
     1. 
  3. On the Test List Toolbar Run the Checked Tests
     1. 
  4. Verify that the test passed, if it did not, check the troubleshooting guide or look at the completed end solution for Exercise 1.
     1. **** The Unit Tests for the ATM StateMachine use the **Microsoft.Activities.UnitTesting** test framework (available at <http://wf.codeplex.com>). This framework makes it easier to build unit tests for all kinds of workflow activities with any unit test framework.

#### Manual Verification

The lab includes a WPF application that will provide a mock user interface for our ATM machine. You can verify the behavior using this application. If the ATM machine does not behave as expected, check the completed end solution for Exercise 1.

* 1. Right click on the **AtmStateMachine.WPF** project and select **Set as StartUp Project**
  2. Select **Debug** / **Start Without Debugging**

|  |  |
| --- | --- |
| Action | Expected Result |
| Click **Power On** | The ATM Machine will quickly move through PowerOn to PowerOff since we have null transitions. You will see messages indicating that the app moved through the Initialize, Insert Card and Final State states. |

* + 1. 

Exercise 2: Card Inserted Scenario

|  |  |
| --- | --- |
| C:\Users\rojacobs\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.IE5\DF10ZJWG\MC900431606[1].png | [Watch Exercise 2 on endpoint.tv](http://channel9.msdn.com/Shows/Endpoint/endpointtv-WF4-State-Machine-Hands-On-Lab-Exercise-2) |

* 1. State Machines spend their time waiting for events to happen. When those events happen and the conditions allow it, the State Machine reacts by transitioning to other states. In this exercise, you will implement additional scenarios that will require you to implement transitions.

|  |  |  |
| --- | --- | --- |
| Given | When | Then |
| A valid ATM card | A valid card is inserted | The ATM will prompt the user to enter their PIN number |
| An invalid card | An invalid card is inserted | The ATM will prompt the user to remove their card |
| A card in the reader | A card is removed | The ATM should clear the screen and return to the Insert Card prompt |

* 1. Our ATM machine includes a card reader. When a user inserts a valid card, the reader will raise an event with data about the card including a flag that indicates if the card is valid or not.
  2. When planning your state machine it can be helpful to describe the states and transitions in a table similar to the following

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| State | Enter Actions | Exit Actions | Transition Event | Transition Enabled When | Transition Action | Destination State |
| Insert Card | Prompt “Insert Card” | Clear Screen | Card Valid |  | Turn on camera | Enter PIN |
| Card Invalid |  |  | Remove Card |
| Remove Card | Prompt “Remove Card” | Clear Screen | Card Removed |  | Turn off camera | Insert Card |

Task 0 – Open the Solution

* 1. Open the begin Solution under **(Lab Folder)\Exercise 2\Begin\AtmStateMachine.sln** or if you prefer continue with the Solution you completed at the end of Exercise 1.
  2. Build the Solution

Task 1 – Add the New States

* 1. Open **AtmActivity.xaml** from the **AtmStateMachine.Activities** project
  2. As the table above shows there are two new states we need to add to our State Machine. Drop two new state activities without auto-connecting them.
  3. Set the **DisplayName** of the first state to **Enter PIN**
  4. Double click on the **Enter PIN** state to open it
  5. Drop a **Prompt** activity on the Entry pane and set the properties

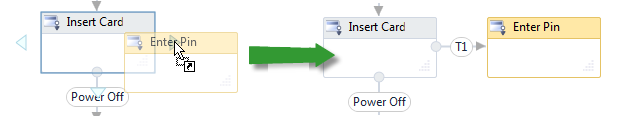
|  |  |
| --- | --- |
| Property | Value |
| Display Name | Prompt Enter PIN |
| Line | 2 |
| Text | Prompts.EnterYourPin |
| Transition | None |

* 1. Drop a **ClearView** activity in the Exit pane
  2. Navigate back up to the **ATM StateMachine**
  3. Set the **DisplayName** of the second state to **Remove Card**
  4. Double click on the **Remove Card** state to open it
  5. Drop a **Prompt** activity to the Entry pane and set the properties

|  |  |
| --- | --- |
| Property | Value |
| Display Name | Prompt Remove Card |
| Line | 2 |
| Text | Prompts.ErrRemoveCard |
| Transition | None |

* 1. Drop a **ClearView** activity in the Exit pane

Task 2- Create the Transition from Insert Card to Enter PIN

* 1. Navigate back up to the **ATM StateMachine**
  2. Connect the **Enter PIN** state to the **Insert Card** state. You can do this by dragging the **Enter PIN** state until the arrows appear and then dropping it on the arrows next to the **Insert Card** state or you can draw a transition arrow if you prefer (more on this later).
     1. 
  3. Change the **DisplayName** of the transition to **Card Valid**. You will need to move the State to see the full text in the transition arrow.
     1. ** Use the Arrow Keys**
     2. When moving states, you can select the state and use the arrow keys. This makes it easier to position the state where you want it.

Task 3 – Create the CardResult Variable

* + 1. When the user inserts a card, the card reader will notify the State Machine and pass a **CardReaderResult** object with information about the card. We need to declare a variable that can hold this result.
  1. Select the **Insert Card** state
  2. Click **Variables** to open the variables pane.
  3. Create a new variable

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Variable Type | Scope | Default |
| CardResult | AtmStateMachine.Activities.CardReaderResult | Insert Card |  |

* + 1. ** Which Scope should I use?**
    2. As with most programming languages, it is best to declare variables at the smallest scope. From the look of the diagram, you might assume that variables used in transitions must be declared at the State Machine scope but that is not the case. Transitions are children of the source state; therefore, variables declared in the source state scope are available to expressions in transitions.

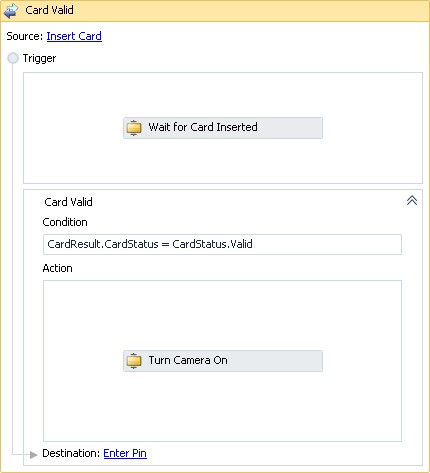
Task 4 – Implement the Card Valid trigger

* 1. Double click on the **Card Valid** transition to open it
  2. Drop the **WaitForCardReader** activity in the Trigger pane and set the properties.

|  |  |
| --- | --- |
| Property | Value |
| CardReaderEvent | CardInserted |
| DisplayName | Wait for Card Inserted |
| Result | CardResult |

* 1. Set the Condition expression to **CardResult.CardStatus = CardStatus.Valid**
     1. **** This transition will apply only when the inserted card is valid. Triggers support an optional Condition expression that determines when the transition applies.
  2. According to the scenario, when the user inserts a card you need to turn on the camera. Drop a **ControlCamera** activity in the Card Valid Action and set the properties

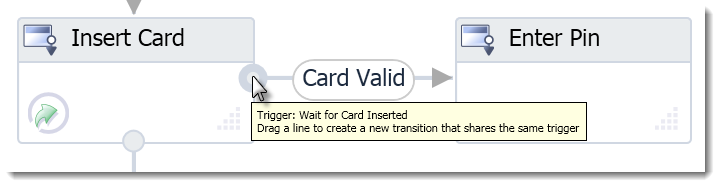
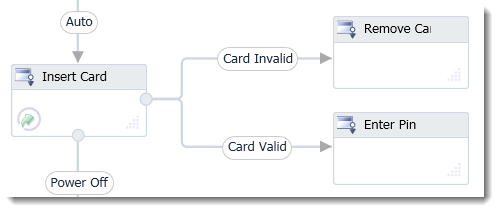
|  |  |
| --- | --- |
| Property | Value |
| DisplayName | Turn Camera On |
| Record | True (checked) |

* + 1. 

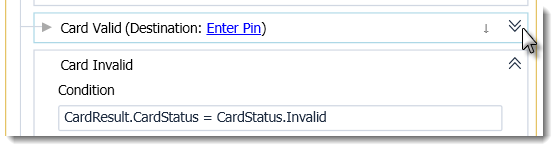
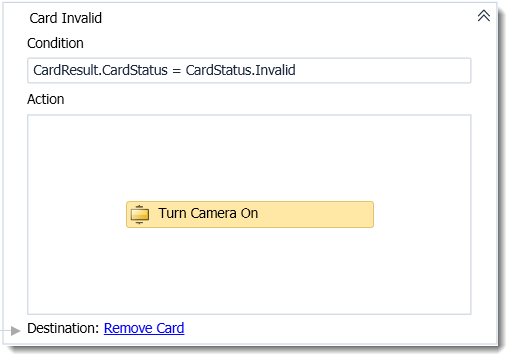
Task 5 – Create a Shared Trigger for an Invalid Card

* + 1. When the user inserts a card, you will either prompt for a PIN number or display an error message prompting the user to remove their card, depending on whether or not the card is valid. In this case, there is one event (a card inserted) with two possible state destinations. The State Machine activity supports this scenario with a ***Shared Trigger***.
  1. Navigate back up to the **ATM StateMachine**

To create a shared trigger, with the transition ***not*** selected move the mouse cursor over the source circle. If you hold the mouse still, you will see a tooltip prompting you to "Drag a line to create a transition that shares the same trigger".

* + 1. 
  1. Drag a line from the start circle of the **Card Valid** transition to the **Remove Card** state.
  2. Change the **DisplayName** of the transition to **Card Invalid**
     1. 
     2. **** Place the Remove Card state above the Enter PIN state for better diagram layout later in the lab.

Task 6 – Configure the Card Invalid Trigger

* 1. Double click the **Card Invalid** transition to open it
  2. Set the Condition expression to **CardResult.CardStatus = CardStatus.Invalid**
  3. Expand the **Card Valid** shared trigger
     1. 
     2. Click to expand the Card Valid shared trigger
  4. According to the scenario, the camera should turn on in the case of an invalid card also. You could drop another **ControlCamera** activity but in this case, it is easier to copy the T**urn Camera** **On** activity we created earlier in the **Enter PIN** transition. Right click on the **Turn Camera On** activity and select **Copy**
  5. Right click in the **Action** pane of the **Card Invalid** shared trigger and select **Paste**
     1. 

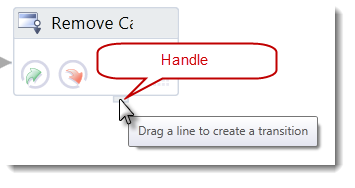
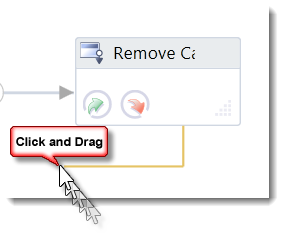
Task 7 - Configure the Power Off Transition

* + 1. In Exercise 1, you added the **Power Off** transition between the **Insert Card** and **Final State**. At that point, we did not configure the transition. The **Power Off** transition is a **Null Transition** that will automatically transition to the final state without waiting for the user to insert a card. To correct this we need to add a trigger.
    2. The ATM Hardware team has provided us with a **WaitForTransition** activity that will wait for one of a number of pre-defined transitions that the hardware will raise.
  1. Navigate back up to the **ATM StateMachine**
  2. Double click on the **Power Off** transition to open it
  3. Drop a **WaitForTransition** activity to the Trigger pane and set the properties

|  |  |
| --- | --- |
| Property | Value |
| AtmTransition | PowerOff |
| DisplayName | Wait for Power Off |
| Result |  |

* 1. ** What is the WaitForTransition Activity?**
  2. Transitions typically use an activity that waits for an event. In WF4, activities that wait for external events will create bookmarks as a mechanism to signal the host application that the workflow needs some data to continue. In our ATM simulator, we are using the **AtmTransition** enumeration to define the set of bookmarks that the State Machine and host application will use.
  3. The **WaitForTransition** activity creates a bookmark using the name of the **AtmTransition** value. The host application will see this bookmark the next time the workflow becomes idle and will enable the UI controls that will invoke that bookmark.

Task 8 - Create the Card Removed transition

* + 1. After the user removes a card, the ATM will return to the **Insert Card** state waiting for the next user. When the user removes the card, the Card Reader will emit an event. We can use this event as the trigger for the **Card Removed** transition.
  1. Navigate back up to the **ATM StateMachine**
  2. Draw a transition from the **Remove Card** state to the **Insert Card** state.
     1. Move the cursor near the edge of the state until you see a handle appear
        1. 
     2. Click and drag the line to the destination state
        1. 
     3. Drag the cursor to a handle on the destination state and release the mouse button
        1. 
  3. Set the **DisplayName** of the transition to **Card Removed**
  4. Double click the **Card Removed** transition to open it
  5. Drop a **WaitForTransition** activity to the Trigger pane and set the properties

|  |  |
| --- | --- |
| Property | Value |
| AtmTransition | CardRemoved |
| DisplayName | Wait for Card Removed |
| Result |  |

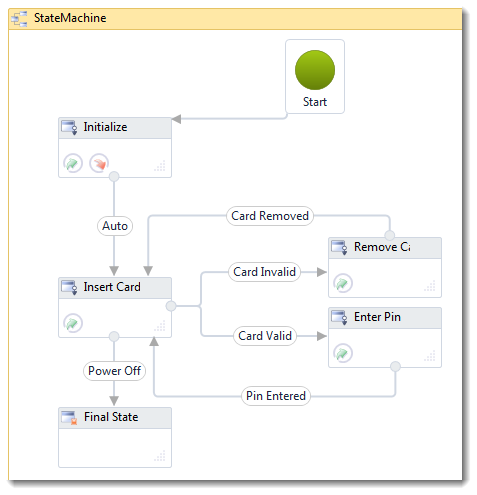
* 1. This is an unconditional transition to leave the condition blank
  2. According to the scenario, when the user removes a card we need to turn off the camera. Drop a **ControlCamera** activity to the Action pane and set the properties

|  |  |
| --- | --- |
| Property | Value |
| DisplayName | Turn Camera Off |
| Record | False (unchecked) |

Task 9 – Create the PIN Entered Transition

* + 1. You will be implementing more scenarios later in the lab but for now, after the user enters their PIN, the ATM will return to the **Insert Card** state.
  1. Navigate back up to the **ATM StateMachine**
  2. Draw a transition from **Enter PIN** to **Insert Card** and set the **DisplayName** to **PIN Entered**
  3. Double click the **PIN Entered** transition to open it
  4. Drop a **WaitForTransition** activity to the Trigger pane and set the properties

|  |  |
| --- | --- |
| Property | Value |
| AtmTransition | KeypadEnter |
| DisplayName | Wait for PIN Entered |
| Result |  |

* + 1. 
    2. **The ATM StateMachine at the end of Exercise 2**

### *Exercise 2 Verification*

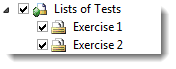
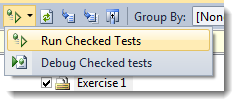
* 1. To verify that the first scenario is working correctly

#### Build the Solution

* 1. Select Build / Build Solution. It should compile without error.
  2. Correct any compile errors.

#### Unit Test Verification

The lab includes a test project that will verify that you have correctly implemented the first scenario by running the tests for Exercise 2

* 1. Open the Test List Editor. From the menu, select **Tests** / **Windows / Test List Editor.**
  2. Check the Test Lists named **Exercise 1** and **Exercise 2**
     1. 
  3. On the Test List Toolbar Run the Checked Tests
     1. 
  4. Verify that the test passed, if it did not, check the troubleshooting guide or look at the completed end solution for Exercise 2.

#### Manual Verification

The lab includes a WPF application that will provide a user interface for our ATM machine

* 1. Right click on **AtmStateMachine.WPF** and select **Set as StartUp Project**
  2. To start the application Select **Debug** / **Start Without Debugging (Ctrl + F5)**
  3. Verify the **Insert Valid Card** Scenario

|  |  |
| --- | --- |
| Action | Expected Result |
| Start the application | The ATM Simulator application will open |
| Click **Power On** | The Power On button should disable  The Power Off button will enable.  You should briefly see “ATM Machine Simulator”, "Please Wait" followed by "Please insert your ATM card" |
| Click **Valid Card** | The ATM will prompt "Enter Your PIN"  The keypad will enable the digits and Clear button.  The Camera circle should turn red |
| Click any Keypad digit  (repeat four times) | Each time you click a digit the third display row will show a "\*" for each digit  After the fourth digit the Keypad Enter button will enable |
| Click the Keypad **Enter** button | Because we have not implemented the rest of the state machine, the ATM will prompt "Please insert your ATM card" and the camera will remain on. |
| Click the **Power Off** button | The ATM will clear the screen  The Power On button will be enabled  The Power Off button will be disabled |
| Close the application | The ATM Simulator application will close |

* 1. Verify the **Insert Invalid Card** and **Remove Card** scenario

|  |  |
| --- | --- |
| Action | Expected Result |
| Start the application | The ATM Simulator application will open |
| Click **Power On** | The Power On button should disable  The Power Off button will enable.  You should briefly see “ATM Machine Simulator”, "Please Wait" followed by "Please insert your ATM card" |
| Click **Invalid Card** | The ATM will prompt "Error reading card, Please remove your card"  The Remove Card Button will enable.  The Camera circle should turn red |
| Click **Remove Card** | The ATM will prompt "Please insert your ATM card"  The Camera circle should turn black |
| Click the **Power Off** button | The ATM will clear the screen  The Power On button will be enabled  The Power Off button will be disabled |
| Close the application | The ATM Simulator application will close |

Exercise 3: PIN Entry

|  |  |
| --- | --- |
| C:\Users\rojacobs\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.IE5\DF10ZJWG\MC900431606[1].png | [Watch Exercise 3 on endpoint.tv](http://channel9.msdn.com/Shows/Endpoint/endpointtv-WF4-State-Machine-Hands-On-Lab-Exercise-3) |

In this exercise, you will implement the following scenarios related to PIN entry.

|  |  |  |
| --- | --- | --- |
| Given | When | Then |
| A valid ATM card is in the reader | The user enters a 4 digit pin and presses enter | Display main menu |
| A valid ATM card entered in the card reader and the ATM is prompting for pin | The user fails to enter their pin before the timeout | The ATM will prompt the user to remove their card |
| A valid ATM card entered in the card reader and the ATM is prompting for pin | The user presses the cancel button | The ATM will prompt the user to remove their card |

Once again, we have analyzed the scenarios to describe our State Machine implementation. You will add one additional state and three transitions in this exercise.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| State | Enter Actions | Exit Actions | Transition Event | Transition Enabled When | Transition Action | Destination State |
| Enter PIN | Prompt Enter PIN | Clear Screen | Keypad Enter |  |  | Main Menu |
|  |  |  | Keypad Cancel |  |  | Remove Card |
| Timeout |  |  | Remove Card |

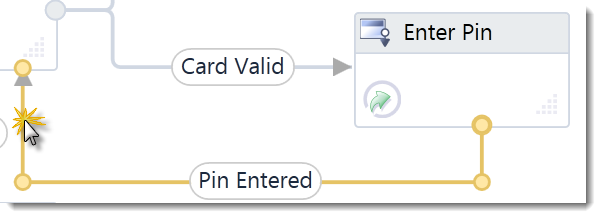
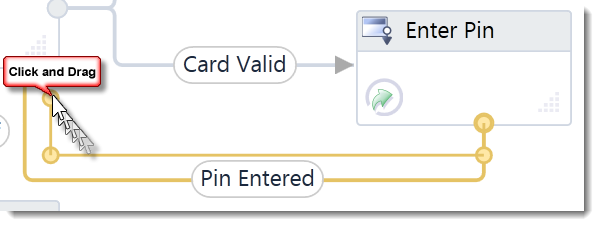
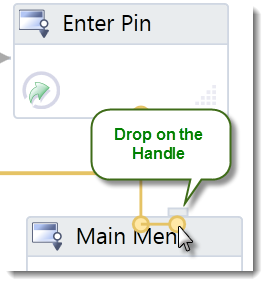
Task 0 – Open the Solution

* 1. Open the begin Solution under **(Lab Folder)\Exercise 3\Begin\AtmStateMachine.sln** or if you prefer continue with the Solution you completed at the end of Exercise 2.
  2. Build the Solution

Task 1 – Add the Main Menu State

* + 1. After entering the PIN number, the ATM will display the main menu. For now we will add the Main Menu state with no implementation
  1. Open **AtmActivity.xaml** from the **AtmStateMachine.Activities** project
  2. Drop a new **State** activity and set the **DisplayName** property to **Main Menu**
  3. The Main Menu must have at least one transition. Draw a transition from the **Main Menu** state to the **Remove Card** state.
  4. Set the **DisplayName** of the transition to **Completed**

Task 2 – Change the PIN Entered Transition

* + 1. In Exercise 2, you created the **PIN Entered** transition and connected it to the **Insert Card** state. Now that you have added the **Main Menu** state, you need to change the **PIN Entered** transition destination from **Insert Card** to **Main Menu.**
  1. Select the transition by clicking on it
     1. **Tip:** Zoom in the diagram to make it easier to select the line
     2. 
  2. Click in the destination circle and drag the destination to the new state.
     1. 
  3. As you near the new state handles will appear on the side of the state allowing you to drop the transition destination on the new state.
     1. 

Task 3 – Add Cancel Transition from Enter PIN

* + 1. You may have noticed at the end of Exercise 2 that you could enter a PIN but the user could not cancel PIN entry. To enable this scenario we need to add a transition from the PIN Entry State to the Remove Card state.
    2. **Tip**: You may want to move the Enter PIN and Main Menu states lower in the diagram to make room for the transition.
  1. Draw a transition from **Enter PIN** to **Remove Card**
  2. Set the **DisplayName** of the transition to **Cancelled**
  3. Double click on the **Cancelled** transition to open it.
  4. Drop a **WaitForTransition** activity to the Trigger pane and set the properties

|  |  |
| --- | --- |
| Property | Value |
| AtmTransition | KeypadCancel |
| DisplayName | Wait for Keypad Cancel |
| Result |  |

Task 4 – Create the Remove Prompt Variable

* + 1. In Exercise 2, you set the prompt for the Remove Card state to Prompts.ErrRemoveCard. At that point, the only scenario that prompted a user to remove their card was when the card was invalid.
    2. Now there is another scenario that also prompts the user to remove a card when they hit the Keypad Cancel button. In this scenario, the card is valid so we need to use a different prompt. To do this, you need to initialize a variable with the prompt in the Trigger action for the different transitions.
  1. Navigate back to the **ATM StateMachine**
  2. Click the **Variables** button to open the variables window
  3. Add a new variable

|  |  |  |
| --- | --- | --- |
| Name | Variable Type | Scope |
| RemovePrompt | String | ATM StateMachine |

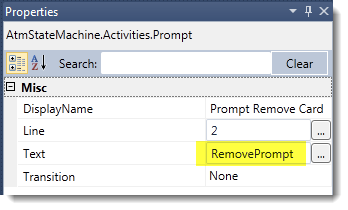
Task 5 – Initialize and Use the Remove Prompt Variable

* 1. Double click the **Card Invalid** transition to open it
  2. To have more than one action you in the Card Invalid transition action pane will need to add a Sequence activity. To do this, right click on the **Turn Camera On** activity and select **Cut**
  3. Drop a **Sequence** in the **Action**
  4. Right click inside the **Sequence** and select **Paste** to addthe **Turn Camera On** activity back to the Sequence.
  5. Drop an **Assign** activity below **Turn Camera On** and set the properties

|  |  |
| --- | --- |
| Property | Value |
| To | RemovePrompt |
| Value | Prompts.ErrRemoveCard |

* 1. Repeat the same process for the Card Valid transition but assign a different prompt

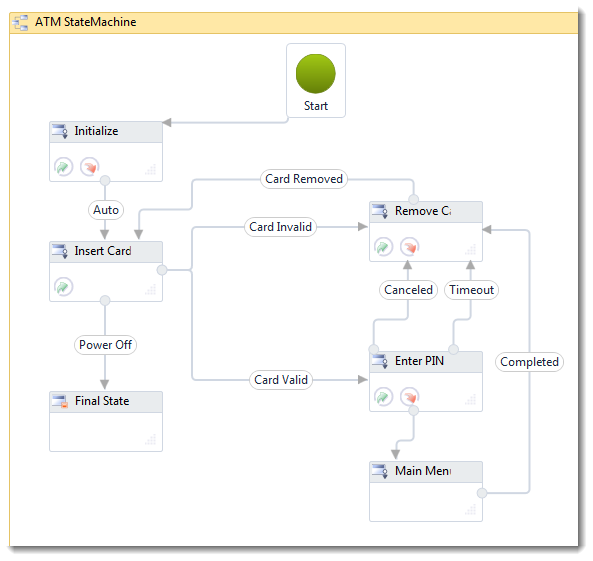
|  |  |
| --- | --- |
| Property | Value |
| To | RemovePrompt |
| Value | Prompts.RemoveCard |

* 1. Open the **Remove Card** State
     1. **Tip:** You can quickly navigate to a state from a transition by clicking the Destination hyperlink
     2. C:\Users\rojacobs\AppData\Local\Temp\SNAGHTML3773812.PNG
  2. Change the **Text** property of the **Prompt Remove Card** activity to use the variable **RemovePrompt**
     1. 

Task 6 – Add Timeout Transition from Enter PIN

* + 1. The ATM scenario says that if the user does not enter a PIN before the time out the ATM is to transition to the Remove Card state.
  1. Navigate up to the **ATM StateMachine**
  2. Draw another transition from **Enter PIN** to **Remove Card**
  3. Set the **DisplayName** of the transition to **Timeout**
  4. Double click on the **Timeout** transition to open it.
  5. Drop a **Delay** activity on the Trigger pane and set the properties

|  |  |
| --- | --- |
| Property | Value |
| DisplayName | Wait for Timeout |
| Duration | Settings.Timeout |

* + 1. **Why use Settings.Timeout?**
    2. You could provide an expression like TimeSpan.FromSeconds(30) but this makes it more difficult to change the timeout duration. By using a Settings argument, you allow the host application to pass a timeout value that will be consistent throughout your workflow.
    3. The other benefit is that when unit testing the workflow you do not want to have to wait the full 30 seconds to test the timeout scenario. The test host will be able to pass a shorter timeout, which will allow your unit tests to run faster.
  1. 
  2. The ATM StateMachine at the end of Exercise 3

### *Exercise 3 Verification*

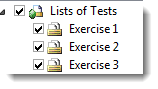
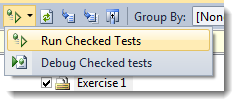
* 1. To verify that the scenarios implemented in this exercise are working correctly

#### Build the Solution

* 1. Select Build / Build Solution. It should compile without error.
  2. Correct any compile errors.

#### Unit Test Verification

The lab includes a test project that will verify that you have correctly implemented the scenarios by running the tests for Exercise 3

* 1. Open the Test List Editor. From the menu, select **Tests** / **Windows / Test List Editor.**
  2. Check the Test Lists for **Exercise 1, Exercise 2 and Exercise 3**
     1. 
  3. On the Test List Toolbar Run the Checked Tests
     1. 
  4. Verify that the tests pass, if they did not, check the troubleshooting guide

#### Manual Verification

The lab includes a WPF application that will provide a user interface for our ATM machine

* 1. Right click on **AtmStateMachine.WPF** and select **Set as StartUp Project**
  2. Select **Debug** / **Start Without Debugging**
  3. Verify the **Keypad Enter to Main Menu** Scenario

|  |  |
| --- | --- |
| Action | Expected Result |
| Click **Power On** | The Power On button should disable  The Power Off button will enable.  You should briefly see “ATM Machine Simulator”, "Please Wait" followed by "Please insert your ATM card" |
| Click **Valid Card** | The ATM will prompt "Enter Your PIN"  The keypad will enable the digits and Clear button.  The Camera circle should turn red |
| Click any Keypad digit  (repeat four times) | Each time you click a digit the third display row will show a "\*" for each digit  After the fourth digit the Keypad Enter button will enable |
| Click the Keypad **Enter** button | You should see the State Main Menu message appear in the event list  Because we have not implemented the main menu yet the state machine will immediately progress to the Remove Card state and prompt "Please remove your card" |
| Click the **Remove Card** button | The ATM will prompt Please insert your ATM card |
| Click the **Power Off** button | The ATM will clear the screen  The Power On button will be enabled  The Power Off button will be disabled |

* 1. Verify the **Keypad Cancel to Remove Card** scenario

|  |  |
| --- | --- |
| Action | Expected Result |
| Click **Power On** | The Power On button should disable  The Power Off button will enable.  You should briefly see "Please Wait" followed by "Please insert your ATM card" |
| Click **Valid Card** | The ATM will prompt "Enter Your PIN"  The keypad will enable the digits and Clear button.  The Camera circle should turn red |
| Click **Cancel** | The ATM will prompt "Please remove your card" |
| Click **Remove Card** | The ATM will prompt "Please insert your ATM card"  The Camera circle should turn black |
| Click the **Power Off** button | The ATM will clear the screen  The Power On button will be enabled  The Power Off button will be disabled |

* 1. Verify the **User Timeout to Remove Card** scenario

|  |  |
| --- | --- |
| Action | Expected Result |
| Click **Power On** | The Power On button should disable  The Power Off button will enable.  You should briefly see "Please Wait" followed by "Please insert your ATM card" |
| Click **Valid Card** | The ATM will prompt "Enter Your PIN"  The keypad will enable the digits and Clear button.  The Camera circle should turn red |
| Wait for 30 seconds | You will see a trace message saying "You have 30 seconds to enter a command"  After the timeout  The ATM will prompt "Please remove your card"  The **Remove Card** button will be enabled |
| Click **Remove Card** | The ATM will prompt "Please insert your ATM card"  The Camera circle should turn black |
| Click the **Power Off** button | The ATM will clear the screen  The Power On button will be enabled  The Power Off button will be disabled |

Exercise 4: Implement the Main Menu

|  |  |
| --- | --- |
| C:\Users\rojacobs\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.IE5\DF10ZJWG\MC900431606[1].png | [Watch Exercise 4 on endpoint.tv](http://channel9.msdn.com/Shows/Endpoint/endpointtv-WF4-State-Machine-Hands-On-Lab-Exercise-4) |

* 1. In this exercise, you will implement the main menu. You may be thinking, if we implement the entire set of ATM scenarios, our diagram is going to be very busy. To avoid this, you will implement a state machine within a state machine to keep the diagram manageable.

In this final exercise, you will implement the following scenarios

|  |  |  |
| --- | --- | --- |
| Given | When | Then |
| A user has inserted a valid card and input a 4 digit PIN | The user completes PIN entry by pressing Keypad Enter | The transaction menu should display Withdraw and Deposit commands and enable the side buttons associated with the commands |
| The transaction menu is displayed | The user fails to select a command before the timeout | The user transaction is cancelled  The ATM will prompt the user to remove their card |
| The transaction menu is displayed | The user presses the cancel button | The user transaction is cancelled  The ATM will prompt the user to remove their card |

Once again, we have analyzed the scenarios to describe our State Machine implementation. You will implement the Main Menu state by composing the ATM StateMachine with an inner state machine. The transitions and states below are for the inner state machine. For the Withdraw and Deposit states, there would be more implementation in a real system but for this lab, you will use a null transition to the Exit state.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| State | Enter Actions | Exit Actions | Transition Event | Transition Enabled When | Transition Action | Destination State |
| Transaction Menu | Display Transaction prompts | Clear Screen | Button 1 (Withdraw) |  |  | Withdraw |
|  | | | Button 3 (Deposit) |  |  | Deposit |
| Timeout |  |  | Exit |
| Keypad Cancel |  |  | Exit |
| Withdraw |  |  | Null |  |  | Exit |
| Deposit |  |  | Null |  |  | Exit |

Task 0 – Open the Solution

* 1. Open the begin Solution under **(Lab Folder)\Exercise 4\Begin\AtmStateMachine.sln** or if you prefer continue with the Solution you completed at the end of Exercise 3.
  2. Build the Solution

Task 1 – Add the Inner State Machine

* 1. Open **AtmActivity.xaml** from the **AtmStateMachine.Activities** project
  2. Double click the **Main Menu** state to open it
  3. Drop a **StateMachine** activity into the Entry pane and set the **DisplayName** to **Transaction Menu StateMachine**

Task 2 – Configure the Transaction Menu Initial State

* 1. Double click the **Transaction Menu StateMachine** to open it
  2. Change the **DisplayName** property of **State1** to **Transaction Menu**
  3. Double click **Transaction Menu** to open it
  4. Drop a **Sequence** on the **Entry** pane
  5. Drop a Prompt activity on the Sequence and set the properties

|  |  |
| --- | --- |
| Property | Value |
| DisplayName | Prompt Withdraw |
| Line | 1 |
| Text | Prompts.Withdraw |
| Transition | Withdraw |

* 1. Drop another **Prompt** activity below **Prompt Withdraw** for the Deposit prompt and set the properties

|  |  |
| --- | --- |
| Property | Value |
| DisplayName | Prompt Deposit |
| Line | 3 |
| Text | Prompts.Deposit |
| Transition | Deposit |

* 1. In the **Exit** pane Drop a **ClearView** activity

Task 3 – Add the Final State

* + 1. Our inner state machine must have a Final State. This state marks the point where our inner state machine is complete and control will return to the outer state machine
  1. Navigate back to the **Transaction Menu StateMachine**
  2. Drop a **Final State** activity and auto connect it or draw a transition from the **Transaction Menu** state to the **Final State**
  3. To make our intention clear, set the **DisplayName** of the Final State to **Exit**. This will indicate that this is not the end but simply the exit point of this menu
  4. Rename the transition **T1** to **Cancel**
  5. Double click the **Cancel** transition to open it
  6. Drop a **WaitForTransition** activity in the Trigger pane and set the properties

|  |  |
| --- | --- |
| Property | Value |
| AtmTransition | KeypadCancel |
| DisplayName | Wait For Cancel |
| Result |  |

Task 4 – Add the Timeout Transition

* 1. Navigate back to the **Transaction Menu StateMachine**
  2. Draw another transition from the **Transaction Menu** to the **Exit** state
  3. Set the **DisplayName** to **Timeout**
  4. Double click the **Timeout** transition to open it
  5. Drop a **Delay** activity in the Trigger pane and set the properties

|  |  |
| --- | --- |
| Property | Value |
| DisplayName | Wait for Timeout |
| Duration | Settings.Timeout |

Task 5 – Add the Withdraw State

* 1. Navigate back to the **Transaction Menu StateMachine**
  2. Drop a **State** activity and auto connect it or draw a transition from the **Transaction Menu** state to the new state.
  3. Set the **DisplayName** of the state to **Withdraw**
  4. Rename the transition **T1** to **Button1**
  5. Double click the **Button1** transition to open it
  6. Drop a **WaitForTransition** activity on the Trigger pane and set the properties

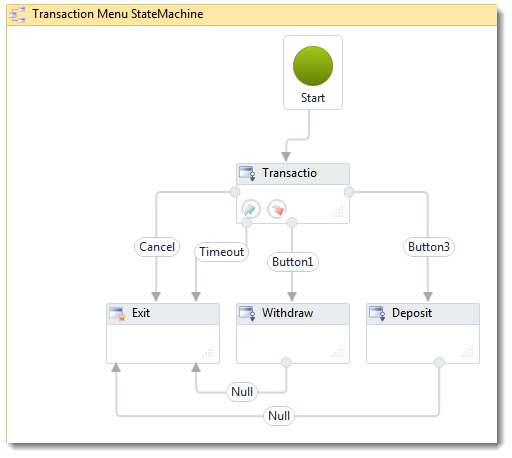
|  |  |
| --- | --- |
| Property | Value |
| AtmTransition | Withdraw |
| DisplayName | Wait For Button1 |
| Result |  |

Task 6 – Add the Deposit State

* 1. Navigate back to the **Transaction Menu StateMachine**
  2. Drop a **State** activity and auto connect it or draw a transition from the Transaction Menu state to the new state.
  3. Set the **DisplayName** of the state to **Deposit**
  4. Rename the transition **T1** to **Button3**
  5. Double click the **Button3** transition to open it
  6. Drop a **WaitForTransition** activity on the Trigger pane and set the properties

|  |  |
| --- | --- |
| Property | Value |
| AtmTransition | Deposit |
| DisplayName | Wait For Button3 |
| Result |  |

Task 7 – Add Null Transitions from Withdraw and Deposit to Exit

* + 1. For the state machine to be valid, the Withdraw and Deposit states must have at least one transition.
  1. Navigate back to the **Transaction Menu StateMachine**
  2. Draw a transition from **Withdraw** to **Exit** and set the **DisplayName** to **Null**
  3. Draw a transition from **Deposit** to **Exit** and set the **DisplayName** to **Null**
  4. 
  5. **The completed inner state machine**

### *Exercise 4 Verification*

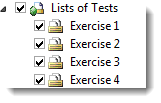
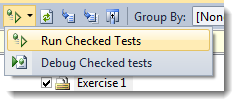
* 1. To verify that the scenarios implemented in this exercise are working correctly

#### Build the Solution

* 1. Select Build / Build Solution. It should compile without error.
  2. Correct any compile errors.

#### Unit Test Verification

The lab includes a test project that will verify that you have correctly implemented the scenarios by running the tests for Exercise 4

* 1. Open the Test List Editor. From the menu, select **Tests** / **Windows / Test List Editor.**
  2. Check the Test Lists for **Exercise 1, Exercise 2, Exercise 3 and Exercise 4**
     1. 
  3. On the Test List Toolbar Run the Checked Tests
     1. 
  4. Verify that the tests pass, if they did not, check the troubleshooting guide

#### Manual Verification

The lab includes a WPF application that will provide a user interface for our ATM machine

* 1. Right click on **AtmStateMachine.WPF** and select **Set as StartUp Project**
  2. Select **Debug** / **Start Without Debugging**
  3. Verify the **Transaction Menu Cancel** Scenario

|  |  |
| --- | --- |
| Action | Expected Result |
| **Power On**  **Insert Valid Card**  **Enter PIN** | The ATM will be at the Transaction Menu  Withdraw will be next to Button 1  Deposit will be next to Button 3 |
| Click **Keypad Cancel** | The ATM will prompt "Please remove your card" |
| **Remove Card**  **Power Off** | The ATM will prompt Please insert your ATM card  The ATM will power off |

* 1. Verify the **Withdraw** scenario

|  |  |
| --- | --- |
| Action | Expected Result |
| **Power On**  **Insert Valid Card**  **Enter PIN** | The ATM will be at the Transaction Menu  Withdraw will be next to Button 1  Deposit will be next to Button 3 |
| Click **Button 1** | The State Machine will pass through the Withdraw state and then  The ATM will prompt "Please remove your card" |
| **Remove Card**  **Power Off** | The ATM will prompt Please insert your ATM card  The ATM will power off |

* 1. Verify the **Deposit** scenario

|  |  |
| --- | --- |
| Action | Expected Result |
| **Power On**  **Insert Valid Card**  **Enter PIN** | The ATM will be at the Transaction Menu  Withdraw will be next to Button 1  Deposit will be next to Button 3 |
| Click **Button 3** | The State Machine will pass through the Deposit state and then  The ATM will prompt "Please remove your card" |
| **Remove Card**  **Power Off** | The ATM will prompt Please insert your ATM card  The ATM will power off |

* 1. Verify the **Timeout** scenario

|  |  |
| --- | --- |
| Action | Expected Result |
| **Power On**  **Insert Valid Card**  **Enter PIN** | The ATM will be at the Transaction Menu  Withdraw will be next to Button 1  Deposit will be next to Button 3 |
| Wait for 30 seconds | The ATM will prompt "Please remove your card" |
| **Remove Card**  **Power Off** | The ATM will prompt Please insert your ATM card  The ATM will power off |

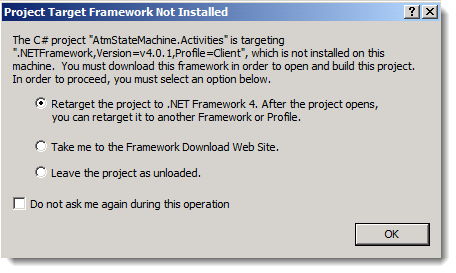
Summary

* 1. In this lab you have seen how the State Machine activity works. In this case we used it to implement a UI scenario but there are many other scenarios where the state machine would be useful. For more information on Workflow and State Machines see
  2. [MSDN Workflow Developer Center](http://msdn.microsoft.com/wf)
  3. [endpoint.tv](http://endpoint.tv/)

[The endpoint blog](http://blogs.msdn.com/endpoint)

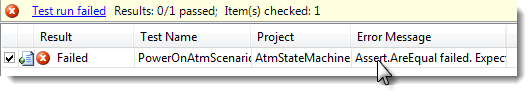
# Troubleshooting

#### Problem – Project Target Framework Is Not Installed

* 1. 

You need to install the [Microsoft .NET Framework 4 Platform Update 1 – Design-time Package for Visual Studio 2010 SP1 (KB2495593)](http://support.microsoft.com/kb/2495593)

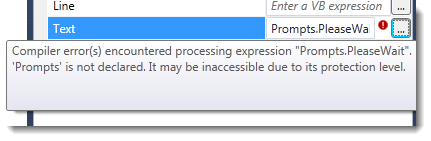
#### Problem – Unit Test Failure

* 1. If a unit test fails, in the test results view, double click on the test failure to open the log for the test
  2. 
  3. In the Additional Information area, you will see the scenario that failed.
  4. In the Error Message area, you will see the specific failure.
  5. In the Error Stack Trace area, you will find a link to the line of test code that failed.
  6. With this information, you should have an idea of where the problem is. Go back to the lab instructions and make sure you completed each step. Remember that the unit tests use state names that must match the DisplayName property of the state exactly including case.

#### Problem – Unit Test fails with an ArgumentException

* 1. System.ArgumentException: The values provided for the root activity's arguments did not satisfy the root activity's requirements:   
     'AtmActivity': The following keys from the input dictionary do not map to arguments and must be removed: Settings. Please note that argument names are case sensitive.
  2. **Solution**
  3. Make sure you added the **Settings** in Argument in Exercise 1 Task 3.

#### Problem Prompts is not declared



**Solution**

Build the solution

#### Problem State must have at least one transition

* 1. System.Activities.InvalidWorkflowException: The following errors were encountered while processing the workflow tree:
  2. 'AtmActivity': The private implementation of activity '1: AtmActivity' has the following validation error: State '<State Name>' must have at least 1 transition.

**Solution**

One of the states in the State Machine has no transitions out of it. Only Final State activities can have no transitions. Check to be sure that you added outgoing transitions for all the states except the Final State.

#### Problem Expected collection of states is larger than actual states found collection

* 1. The test looks for states that belong to a state machine by **DisplayName**. Make sure you have the correct name for the State Machine. The outer state machine (from Exercise 1) should be named “ATM StateMachine” and the inner state machine (from Exercise 4) should be named “Transaction Menu StateMachine”. Remember the state machine and state names are case sensitive.

#### Problem Assert.AreEqual failed. Expected:<Initialize>. Actual:<State1>. found at index 0 expected states <Initialize, Insert Card> found states <State1, State2, Final State>

* 1. The unit test verifies states by Display Name. The name must match exactly the names provided in the lab manual.
  2. ***Assert.IsTrue failed. Expected collection of <Prompts> is larger than actual collection, expected <Please Wait, Please insert your ATM card> found <Please Wait>***
  3. The unit test verifies the prompts that should appear during the scenario. Make sure you added all the prompts and used the correct expression.

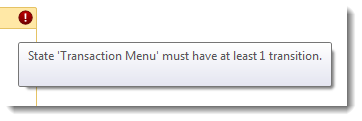
#### Assert.IsTrue failed. Verify that you added a ClearView activity to the Exit Action of the Initialize State

The unit test verifies that the ClearView activity was called the correct number of times. Make sure you add the ClearView to the Exit action of states when instructed.

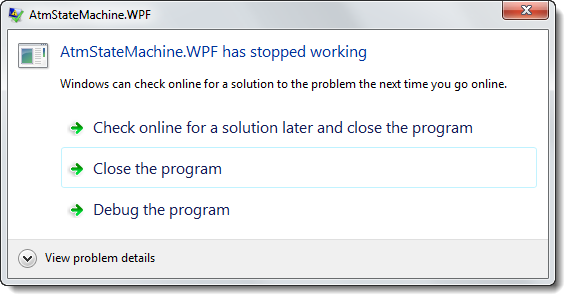
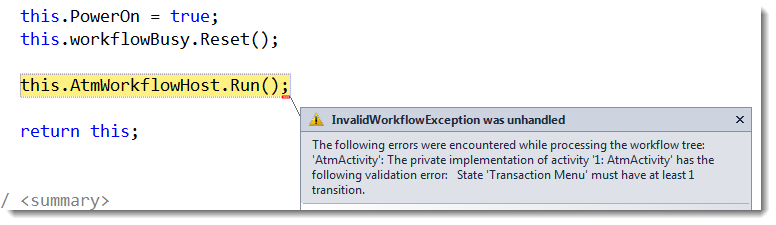
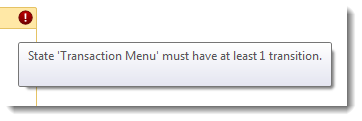
#### Assert.AreEqual failed. Expected:<Error reading card, Please remove your card>. Actual:<Please remove your card>. in collection of <Prompts> found at index 2 …

* 1. In Exercise 2 Task 1 Step 10, make sure you use Prompts.ErrRemoveCard rather than Prompts.RemoveCard.

#### State (State Name) Must have at least 1 transition

* 1. 
  2. The State Machine requires that all states have at least one transition. The application will compile but will fail to run the state machine if it is invalid. Make sure you have followed the lab instructions to draw a transition to some other state.

#### AtmStateMachine.WPF has stopped working

* 1. 
  2. This occurs when the WPF ATM simulator application has an unhandled exception. If you run with the debugger, you will see what the exception is.
  3. 
  4. Typically, the unhandled exception will be a workflow validation error such as the one shown above.
  5. **Solution**
  6. Correct the validation error. Open the workflow and look for activities marked with a validation error icon. 

#### An ATM Button Does Not Enable As Expected

* 1. The WPF UI enables and disables buttons when it detects that the state machine is waiting for a command from the UI. The WaitForTransition and WaitForCardReader activities will create the bookmarks that cause the buttons to enable.
  2. **Solution**
  3. If a button is not enabled when the verification instructions indicate that it should be check the transition activities to be sure that they are waiting for the command indicated in the lab manual.