## Automating Working with SharePoint 2013 Workflow

**Lab Time**: 60 minutes

**Lab Folder**: [[StudentFolder]]\Workflow

**Lab Overview**: This hands-on lab will walk you through the experience of creating workflows using both SharePoint Designer 2013 and Visual Studio 2012. You will get first person experience in using the new activities, loops and stages capabilities in SharePoint Designer as well as creating workflows that are used in SharePoint apps. Further, you will also see how to create workflows that communicate with OData web services.

### Exercise 1: Setup Lab Environment

In this exercise you will setup your environment.

All exercises in this lab assume you will work in a new site collection, http://workflow.wingtip.com.

1. Setup a new site collection for this lab:
   1. Ensure you are logged into the **WingtipServer** server as **WINGTIP\Administrator**.
   2. Run a PowerShell script, found in the root lab folder for this module:
      1. Right-click **SetupModule.ps1** and select **Run with PowerShell**. This file can be found in the files associated with this lab:

[..]\Workflow

* 1. When the script completes, it will launch a new browser and navigate to the lab site collection.
  2. Close the PowerShell console window.

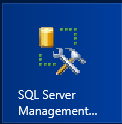
#### Setup Sample Resource Web Services

1. Ensure you are logged into the **WingtipServer** server as **WINGTIP\Administrator**.
2. Check to see if the sample services have already been deployed:
   1. Open Internet Explorer and try to navigate to the following OData services:
      1. <http://cptresources.wingtip.com:81/services/AdventureWorks2012Person.svc>
      2. <http://cptresources.wingtip.com:81/services/AdventureWorks2012Product.svc>
      3. <http://cptresources.wingtip.com:81/services/Calculator.svc>
   2. If the links above work and do not return 404 errors, you can skip to the next exercise **(Exercise 2)** because the sample site and services have already been deployed.
3. If the **Calculator.svc** service works but the two **AdventureWorks2012\*.svc** services **do not work**, this indicates the services were deployed, but two of the services are not able to connect to the local SQL Server sample **AdventureWorks2012** database. The possible issues include:
   1. A connection problem with the IIS website.
   2. The connection strings in the **cptservices** application’s **web.config**.
   3. The sample **AdventureWorks2012** database is not installed in the local SQL Server.
   4. The identity of the hosting app pool does not have rights to the **AdventureWorks 2012** database.
4. If all three of the web services return 404 errors continue on to step 6 to create these.

You must resolve any issues and ensure these web services are working before proceeding to Exercise 2.

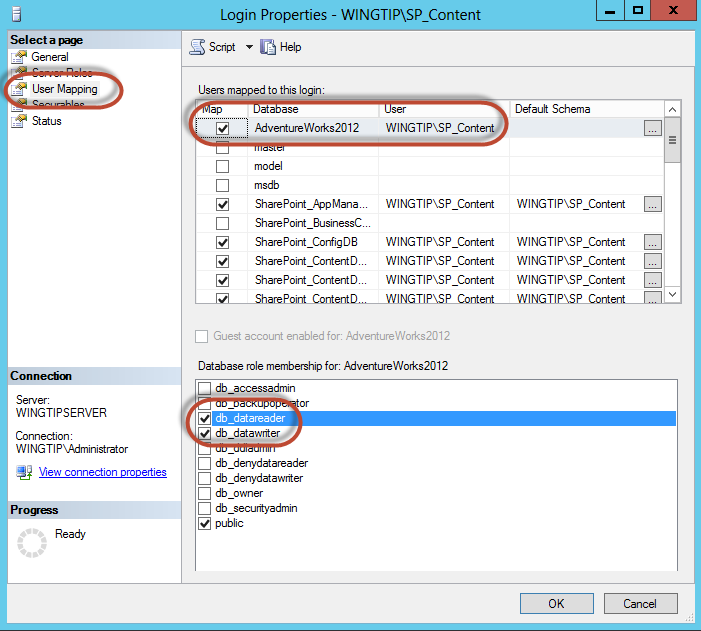
#### Update the App Pool Identity Rights in the SQL Server Adventure Works Database

1. Grant the identity of the hosting web site’s application pool access to the AdventureWorks2012 sample database:
   1. Open **SQL Server Management Studio**:
      1. **Windows Keyboard Key 🡪 SQL Server Management Studio.**



**(Note:** Alternatively after pressing the **Windows Keyboard Key** you can simply start typing the name of the program you are looking for (e.g. SQL); this will filter the results to those that match the letters typed on the keyboard)

* 1. In the **Connect to Server** dialog, enter the following and click **Connect**:
     1. **Server Type:** Database Engine
     2. **Server name:** WINGTIPSERVER
     3. **Authentication:** Windows Authentication
  2. In the **Object Explorer** tool window, expand the tree to show the contents of **WINGTIPSERVER 🡪 Security 🡪 Logins**.
  3. Right-click **WINGTIP\SP\_Content** and select **Properties**.
  4. In the **Select a page** pane on the left-hand side of the **Login Properties – WINGTIP\SP\_Content** dialog, select **User Mapping**.
  5. In the top portion of the dialog, in the **Users mapped to this login**, check the box next to **AdventureWorks2012**.
  6. In the bottom portion of this dialog, in the **Database role membership for AdventureWorks2012**, check the boxes next to:
     1. **db\_datareader**
     2. **db\_datawriter**



* 1. Click **OK** and **close** SQL Server Management Studio.

#### Create Sample Resource Website & Deploy Sample Services Project

1. Create a new IIS web site to host the sample CPT OData web services:
   1. Run a PowerShell script, found in the **ExtraStudentFiles** folder in the student folder:
      1. Right-click **CreateCptResourcesWebSite.ps1** and select **Run with PowerShell**. This file can be found in the files associated with this course:

[..]\ExtraStudentFiles\Scripts

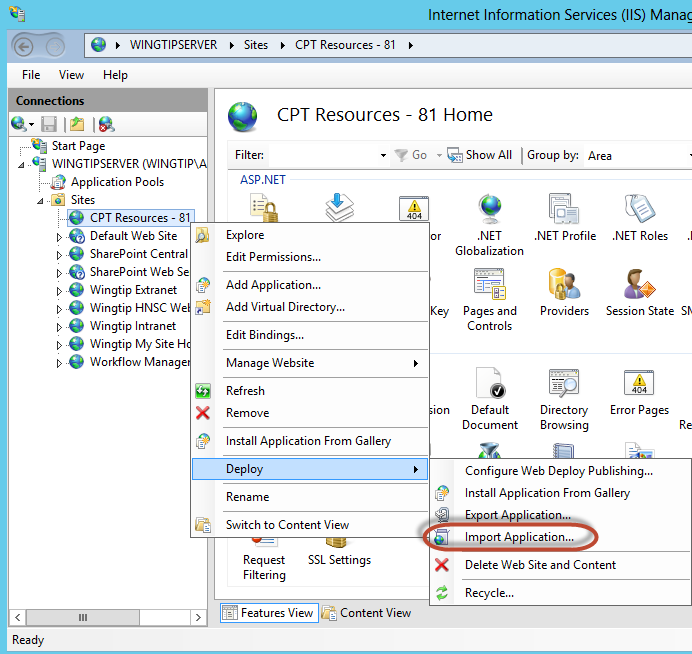
This script will create a new website in IIS named **CPT Resources – 81** with a binding of **http://cptresources.wingtip.com:81**.

* 1. Close the PowerShell console window.

1. Install the CPT Services sample OData Web services project:
   1. Open **Internet Information Services (IIS) Manager**:
      1. **Windows Keyboard Key 🡪 Internet Information Services (IIS) Manager.**



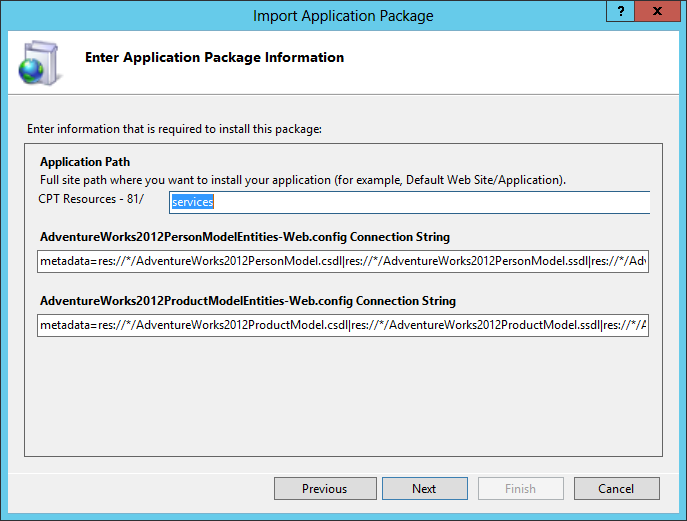
* 1. In the left-hand **Connections** pane, expand the tree to the following location:
     1. **WINGTIPSERVER 🡪 Sites 🡪 CPT Resources – 81**
  2. Right-click **CPT Resources – 81** and select **Deploy 🡪 Import Application**.



* 1. In the **Import Application Package** dialog’s **Select the Package** page, browse to the file **CptServices\_WebDeployPackage.zip** found in the files associated with this course:

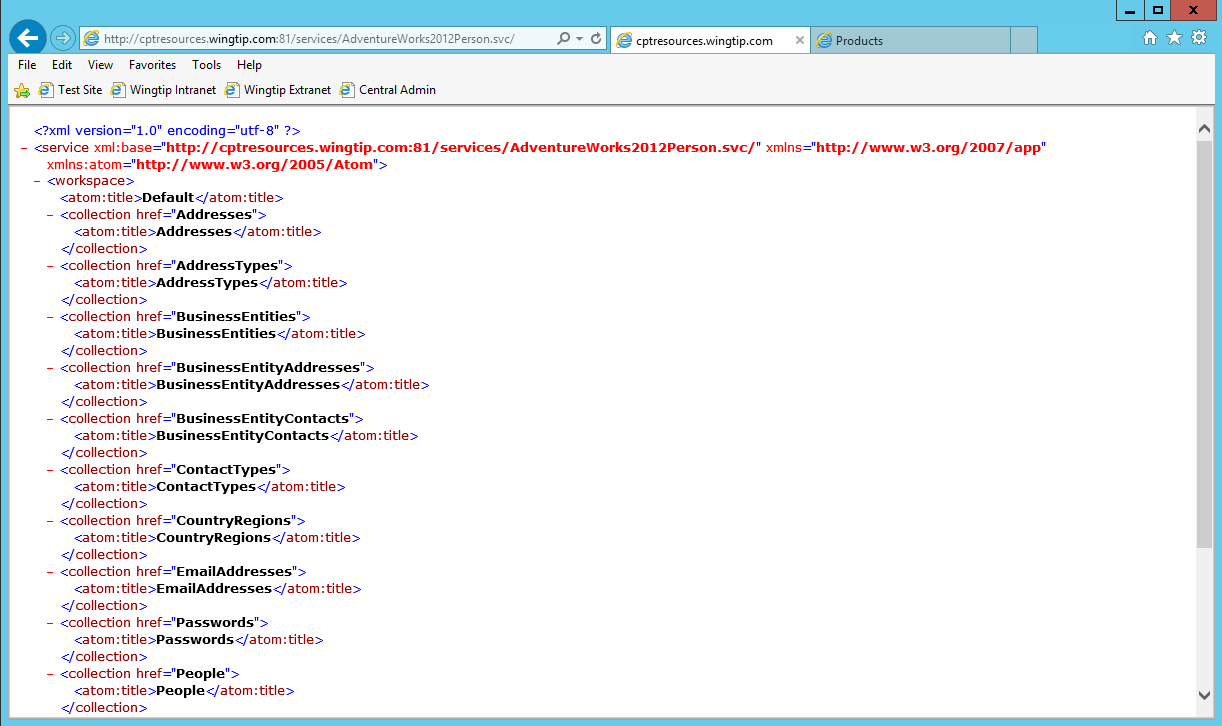
[..]\ExtraStudentFiles\Resources

* 1. Click **Open** and then click **Next**.
  2. On the **Select the Contents of the Package** page, click **Next**.
  3. On the **Enter Application Package Information** page, make sure the **Application Path** is set to **services** and click **Next.**

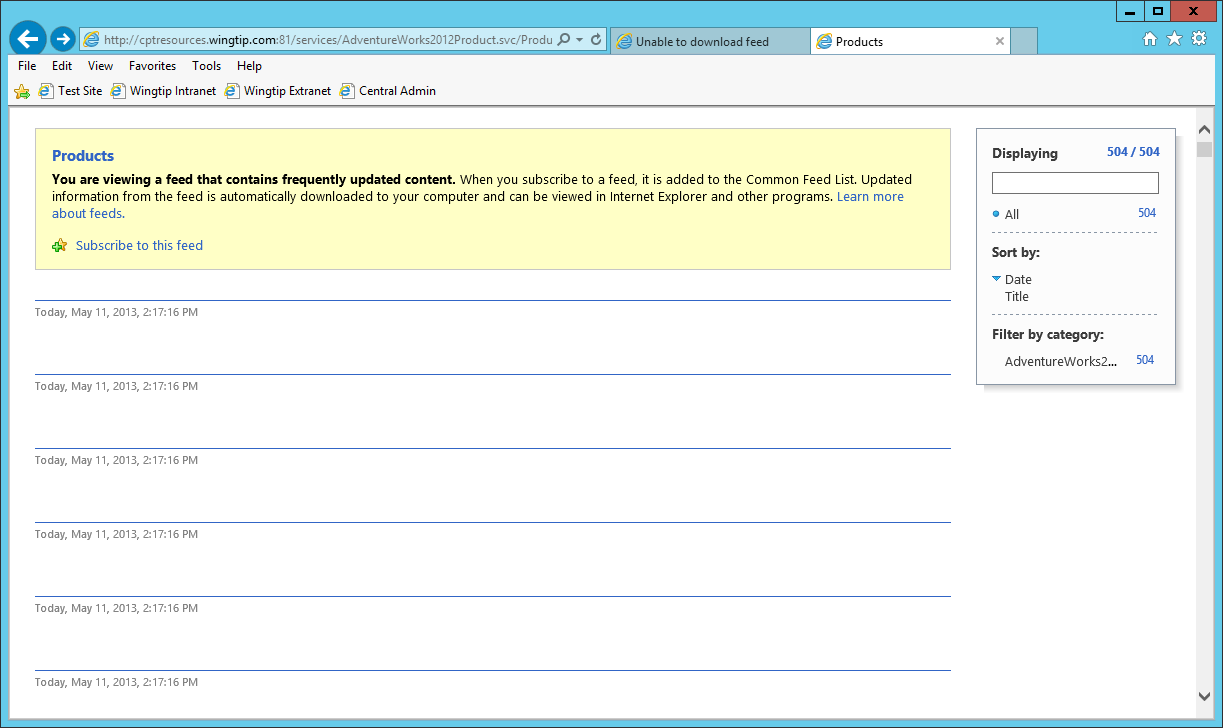


* 1. After the package is deployed, click **Finish** on the **Installation Progress and Summary** page to close the dialog.

1. Verify the sample OData services are working:
   1. Open Internet Explorer and try to navigate to the following OData services:
      1. <http://cptresources.wingtip.com:81/services/AdventureWorks2012Person.svc>
      2. <http://cptresources.wingtip.com:81/services/AdventureWorks2012Product.svc>
      3. <http://cptresources.wingtip.com:81/services/Calculator.svc>
   2. You should see XML returned by all services as the following figure shows:



* 1. Open Internet Explorer and try to navigate to the following OData services:
     1. <http://cptresources.wingtip.com:81/services/AdventureWorks2012Person.svc>/People
     2. <http://cptresources.wingtip.com:81/services/AdventureWorks2012Product.svc/Products>
  2. With the **Person.svc/People** you should see a message that **Internet Explorer cannot display this feed (The size of this feed exceeds the download limit)**
  3. With the **Products.svc/Products** you should see feed information on Products similar to that shown in the image below:



You are now ready to perform the following lab exercises.

### Exercise 2: Visio 2013 & SharePoint Designer 2013 Workflows

In this exercise you will create a new workflow using Visio 2013 and SharePoint Designer 2013. In this workflow you will leverage the new support for stages.

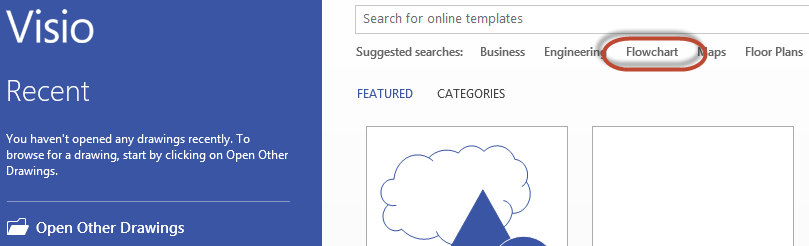
1. Ensure you are logged into the **WingtipServer** server as **WINGTIP\administrator**.

#### Model the Workflow Using Visio 2013

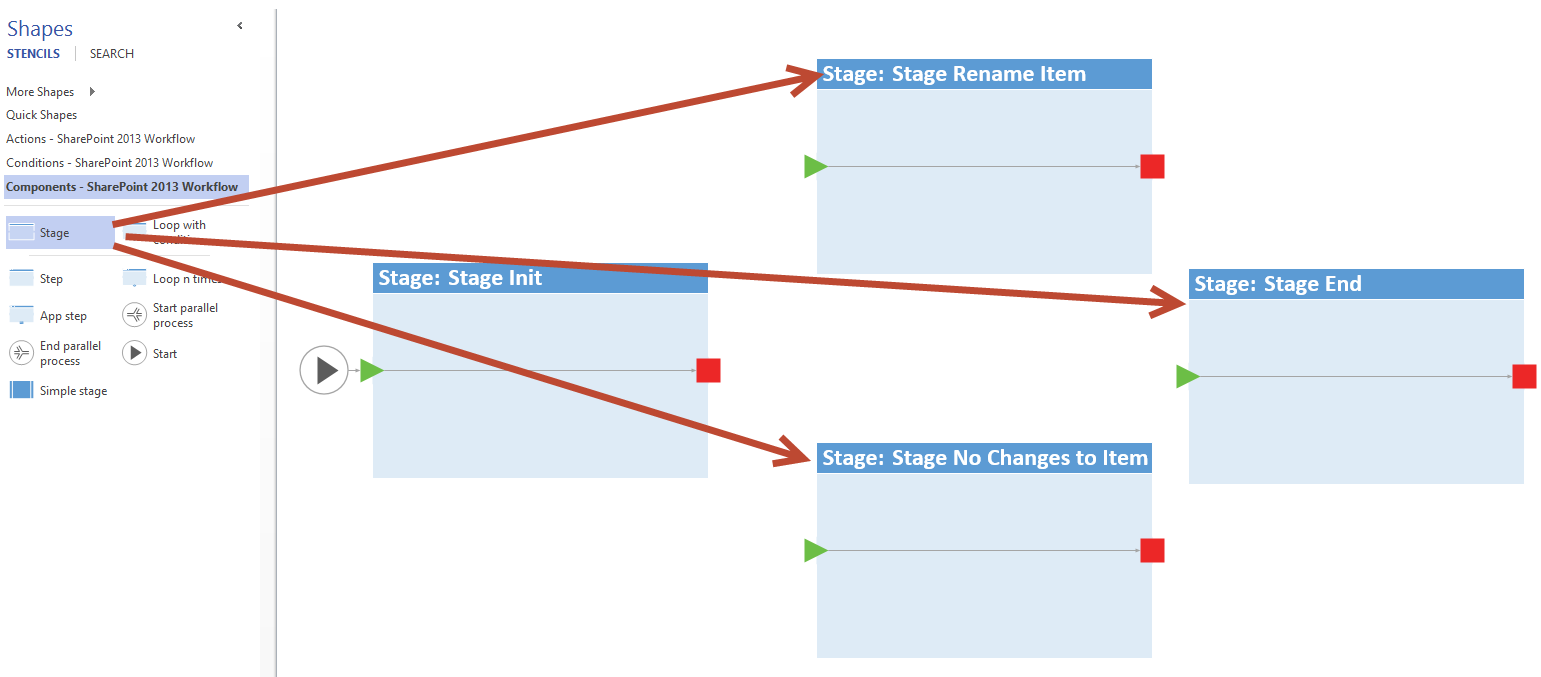
1. Open **Visio 2013**: **Windows Keyboard Key 🡪 Type “visio”** and then select the Visio 2013 application.



1. When **Visio** loads, click the **Flowchart** category under the search box:



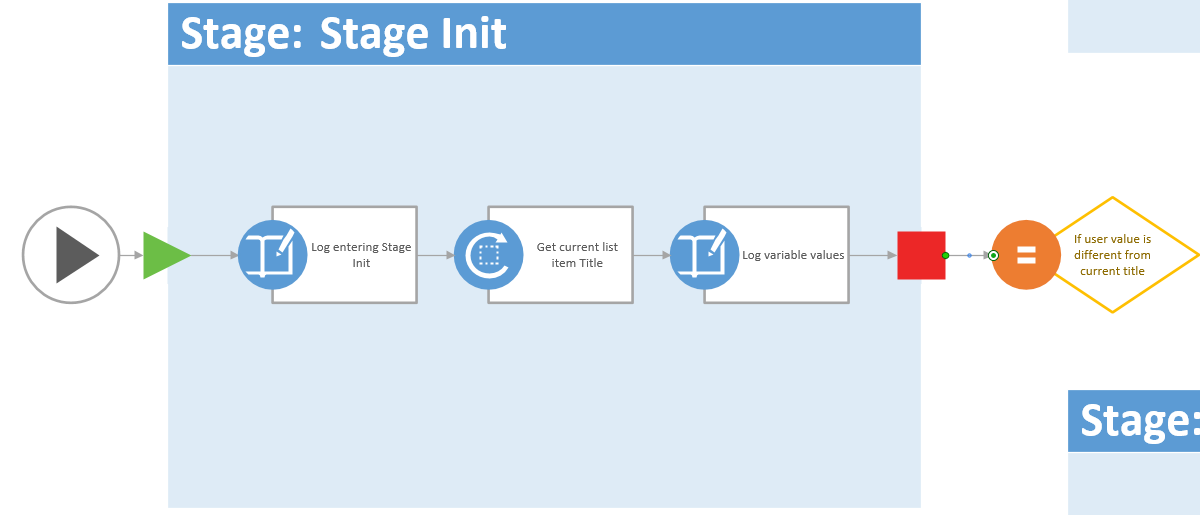
1. Select the template **Microsoft SharePoint 2013 Workflow**, followed by the **Create** button.
2. When the designer loads, right-click the dark blue heading of **Stage 1** and select **Edit Text**. Rename the stage to **Stage Init**.
3. Add three more stages to the designer by dragging the **Stage** shape from the **Shapes** task pane and rename them to the following stages as shown in the follow figure:
   1. **Stage Rename Item**
   2. **Stage No Changes to Item**
   3. **Stage End**



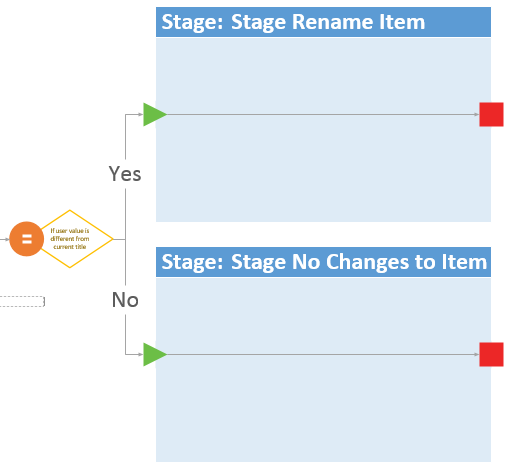
1. Add activities to **Stage Init**:
   1. Using the **Shapes** pane, select the **Actions – SharePoint 2013 Workflow** category and add the following actions to **Stage Init**:
      1. **Log to history list**
      2. **Set workflow variable**
      3. **Log to history list**

When you are dragging-and-dropping actions from the Visual Studio toolbox on the workflow designer, make sure you are dropping these actions on the connector (i.e the black line connecting the green triangle and red square) within the stage. If you don’t do this, you will have to fix up the connections before saving the workflow.

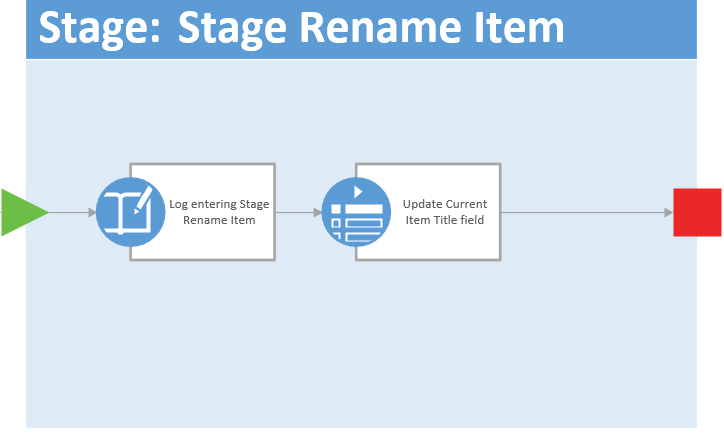
1. Rename the actions by right-clicking each one and selecting **Edit Text**. Give each action a descriptive name as follows:
   1. **Log to history list** = Log entering Stage Init
   2. **Set workflow variable** = Get current list item Title
   3. **Log to history list** = Log variable values
2. Using the **Shapes** pane, select the **Conditions – SharePoint 2013 Workflow** category and add an **If any value equals value** condition to the workflow immediately to the right of **Stage Init** (i.e. to the right of the red square).
   1. Rename this condition to **If user value is different from current title**.
3. Using the **Connector** tool, found in the **Home** tab of the ribbon in the **Tools** group, connect the red box from **Stage Init** to the condition you just added.
   1. To Connect the red box with the orange circle, click on the **Connector** tool in on the Home tab and then place your **mouse pointer near the right hand edge of the red box** (you should see a **small green square** show up
   2. Click on the small green square on the right edge of the red box and drag towards the orange circle until you see the small green square appear on the left edge of the orange circle.
   3. Release the left mouse button.
4. The **Stage Init** should look like the following figure:



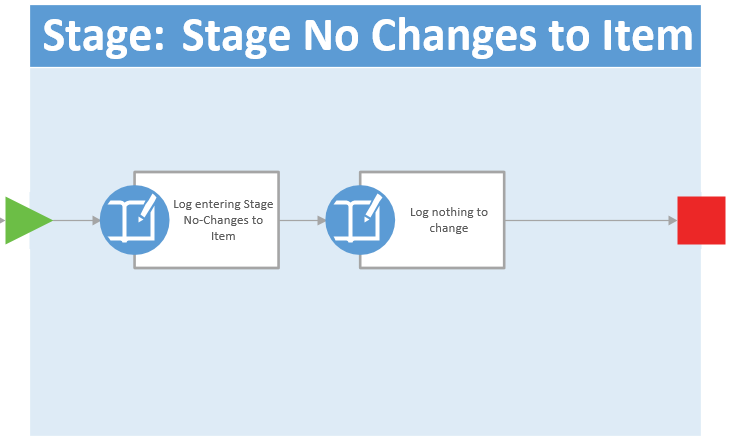
1. Using the same technique with the Connector tool, draw two connections from the condition, one to the start arrow of **Stage Rename Item** and one to the **Stage No Changes to Item**.
2. Right-click the connector to **Stage Rename Item** and select **Yes**.
3. Right-click the connector to **Stage No Changes to Item** and select **No**.
4. The condition connections should look like the following figure:



1. Use the same techniques you just applied, update **Stage Rename Item** by adding the following actions and renaming them it so it looks like the following figure:
   1. **Log to history list** = Log entering Stage Rename Item
   2. **Set field in current item** = Update current item Title field



1. Update the Stage No Changes to Item by adding the following actions so it looks like the following figure:
   1. **Log to history list** = Log entering Stage No Changes to Item
   2. **Log to history list** = Log nothing to change



1. Using the **Connector** tool, connect the terminating red square on the right of **Stage Rename Item** to the green arrow on **Stage End**.
2. Repeat the previous step connecting the **Stage No Changes to Item** to **Stage End**.
3. Add a **Log to history list** action to the **Stage End** stage.
   1. Rename it to **Log Stage End**
4. Save your changes selecting **File 🡪 Save**.
   1. When prompted, save the file to the **desktop** with a filename of **RenameItemWorkflow.vsdx**.

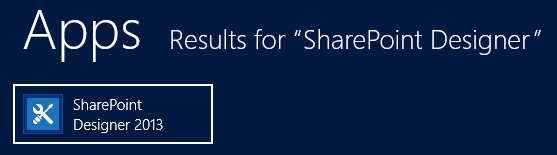
A copy of the drawing can be found in the **[..]\Exercises\Ex2\_Completed** folder in this lab’s files

1. Close **Visio 2013**.

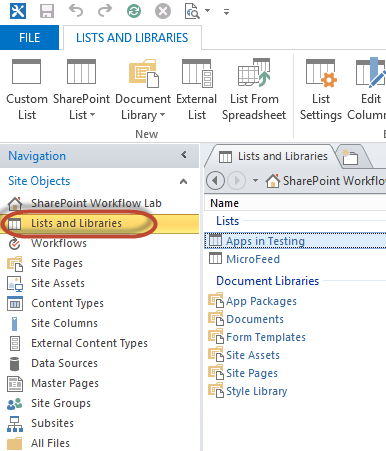
At this point you have designed your workflow using Visio 2013. This simulates the process of creating a workflow with a business user without access to SharePoint Designer 2013 or a connection to SharePoint 2013.

#### Complete the Workflow in SharePoint Designer 2013

1. Open SharePoint Designer 2013: Windows Keyboard Key 🡪 Type “SharePoint Designer” and then select the SharePoint Designer 2013 application.



1. Open the lab site collection:
   1. Click the **Open Site** button.
   2. In the **Open Site** dialog, enter <http://workflow.wingtip.com> in the **Site Name** box and click **Open**.
   3. If prompted to login, use the credentials for **WINGTIP\administrator**.
2. When SharePoint Designer loads, create a new announcement list:
   1. Select **Lists and Libraries** in the **Navigation** pane.

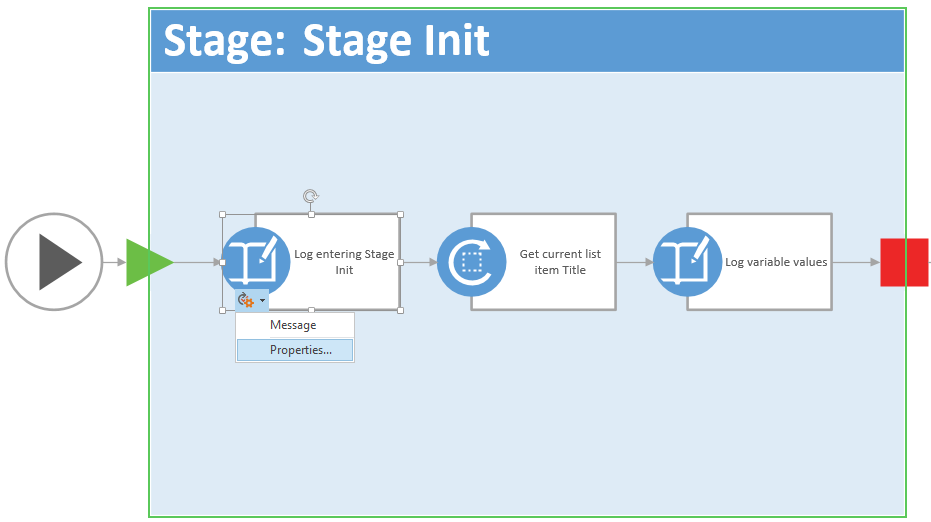


* 1. In the ribbon, select **SharePoint List 🡪 Announcements**.
  2. In the **Create list or document library** dialog, set the **Name** to **Announcements** and click **OK**.

1. Import the workflow created with Visio 2013:
   1. Select **Workflows** in the **Navigation** pane.
   2. In the ribbon’s **Manage** group, select **Import from Visio 🡪 Import Visio 2013 Diagram**.
   3. Find the **RenameItemWorkflow.vsdx** drawing you created on your desktop using Visio 2013 and click **Open**.
   4. In the **Create Workflow** dialog, set the following values and click **OK**:
      1. **Name:** Rename Item Workflow
      2. **Workflow Type:** List Workflow
      3. **SharePoint List:** Announcements

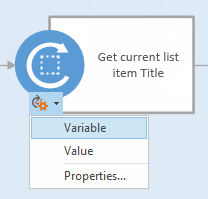
When the workflow loads you’ll notice it looks identical to the drawing in Visio. In fact, it almost seems like it opened Visio, but if you look at the very top of the application, you’ll see you still have the <http://workflow.wingtip.com> site open in SharePoint Designer 2013.

1. Before proceeding create a new variable that will trigger a form to appear when the user starts the workflow. This will prompt the user to optionally enter a new value for the title of the announcement.
   1. In the ribbon, click the **Initiation Form Parameters** button in the **Variables** group.
   2. In the **Association and Initiation Form Parameters** dialog, click **Add**.
   3. Enter the following values and click **Next**:
      1. **Field Name:** New Announcement Title
      2. **Description:** Do you want to change the title of the announcement?
   4. Click **Next** and then click **Finish**.
   5. Click **OK**
2. Select the first action **Log entering Stage Init** in **Stage Init**. Notice a little tile appears in the lower left corner. Click it and select **Properties**:

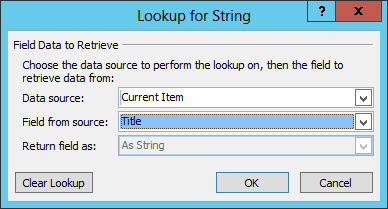


* 1. In the **Log to History List Properties** dialog, set the **Message** to **Entering Stage Init…** by clicking the **…** button, entering the text and clicking **OK** twice to accept your changes.

1. Open the **Get Current List Item Title** action’s **Properties** dialog using the same process in the last step to get the tile to appear, except don’t select **Properties…** instead select **Variable**.

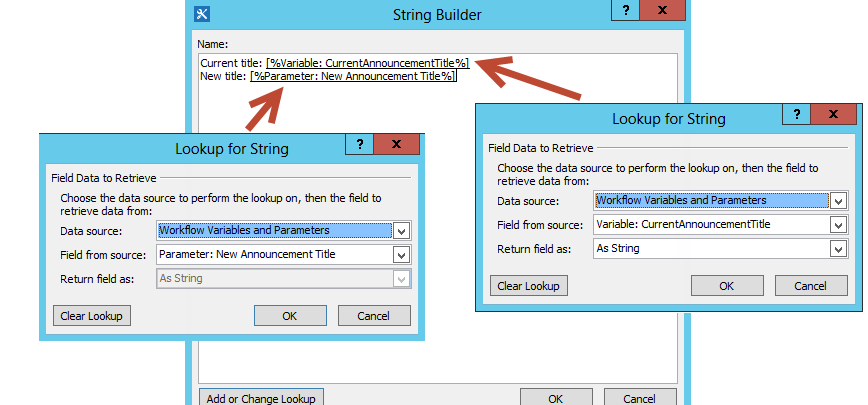


* 1. For the **Variable** property, click the drop down and select **Create a new variable**.
     1. **Name:** CurrentAnnouncementTitle
     2. **Type:** String
     3. Click **OK**
  2. Click into the **Value** property and click the **fx** button and select the following options:



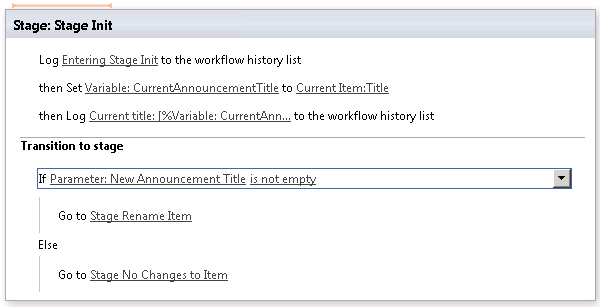
* 1. Click **OK** twice

1. Open the **Log variable values** action’s **Properties** dialog using the same process as before.
   1. Click into the **Message** field and click the builder button.
   2. Enter the following message, using the **Add or Change Lookup** button to add the values shown in the following figure:
      1. Type in the **Current title:** and **New title:** and then click the **Add or Change Lookup** button with the cursor positioned to the right of each entry to add the **Lookup for String** entries as shown below.

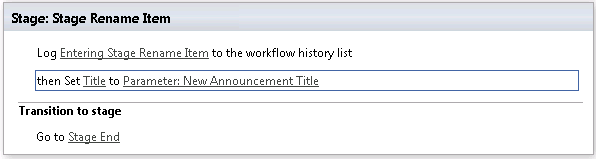


* 1. Click **OK** twice

1. Switch from the Visual Designer view to the **Text-Based Designer** by clicking the **Views** button in the ribbon in the **Manage** group.
2. Locate the **Transition to stage** section within **Stage Init**. Update the path the workflow should take depending on the values selected:
   1. In the first part of the **If** statement, select the following linked phrases and update their values:
      1. **value** (the first one): Parameter: New Announcement Title  
         (Hint: use the **fx** button to add this and select **Workflow Variables and Parameters** as your Data source).
      2. **equals:** Change to **is not empty**

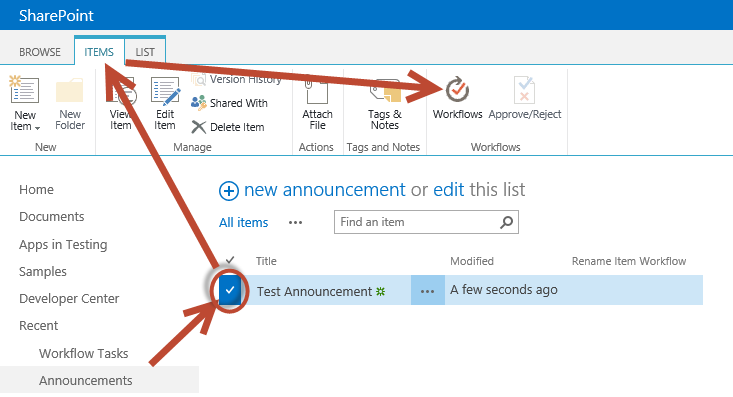


1. Update the **Stage Rename Item** to reflect what you see in the following figure using the techniques you learned in the previous steps:



1. Update the **Stage: No Changes to Item**
   1. Set the **message** for the first **Log to history list** action to **Entering Stage No Title Changes Specified**.
   2. Set the **message** for the second **Log to history list** action to **Nothing to change**.
2. Finally, in the **Log to history list** action within the **Stage End**, click the **message** link and enter **Workflow complete**.
3. Click the **Save** button in the ribbon group **Save** to save your workflow.

#### Deploy and Test the Workflow

1. Publish the workflow to SharePoint 2013 and Workflow Manager:
   1. Click the **Publish** button in the ribbon group **Save**.
2. Create a new list item to be a test subject:
   1. Open **Internet Explorer** and browse to <http://workflow.wingtip.com>.
   2. In the **Quick Launch** to the left of the page, select **Announcements**.
   3. Create a new item by selecting the **new announcement** link or from the ribbon, click the **Items** tab and then the **New Item** button.
   4. Set the **Title** of the new item to **Test Announcement** and click **Save**.
3. Start a new instance of your workflow:
   1. In the **Announcements** list, select the **Test Announcement** item (i.e. click on the **check mark area** to the left of the **Test Announcement** Item)
   2. Within the ribbon, select the **Items** tab and then click the **Workflows** button under the **Workflows** group.  
      
   3. On the **Announcements: Workflows: Test Announcement** page, click **Rename Item Workflow** under the **Start a New Workflow** heading.
   4. A form will load prompting you to enter a new announcement title. Enter **A New Title** and click **Start**.

It will take a moment for the workflow to start and process all the way through… be patient while this happens. In the meantime continue on to the next step

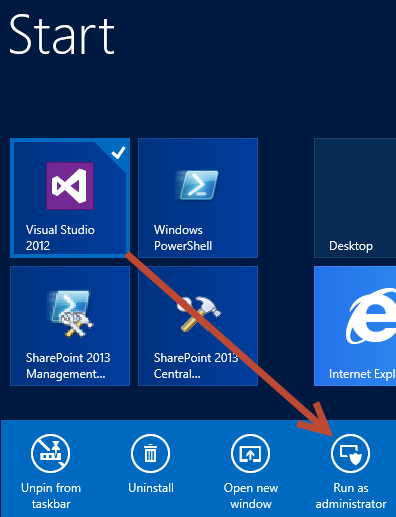
1. When SharePoint returns you back to the **Announcements** list, select the **Test Announcement** item.
2. Within the ribbon, select the **Items** tab and then click the **Workflows** button under the **Workflows** group.
3. Notice the status page will show both running and completed workflows Click the status link for the workflow you just started to see the status page with the logging information.
4. If it hasn’t completed by now, keep refreshing the page every few seconds to see if it has completed. Once it completes you should be able to see the **Internal Status** field change to **Completed**, the title updated, and the logging information for the workflow executed.

In this exercise you created a new workflow first with Visio 2013 and then completed it using SharePoint Designer 2013. You got some hands on experience with the new stages support and design tools in SharePoint 2013 and Workflow Manager.

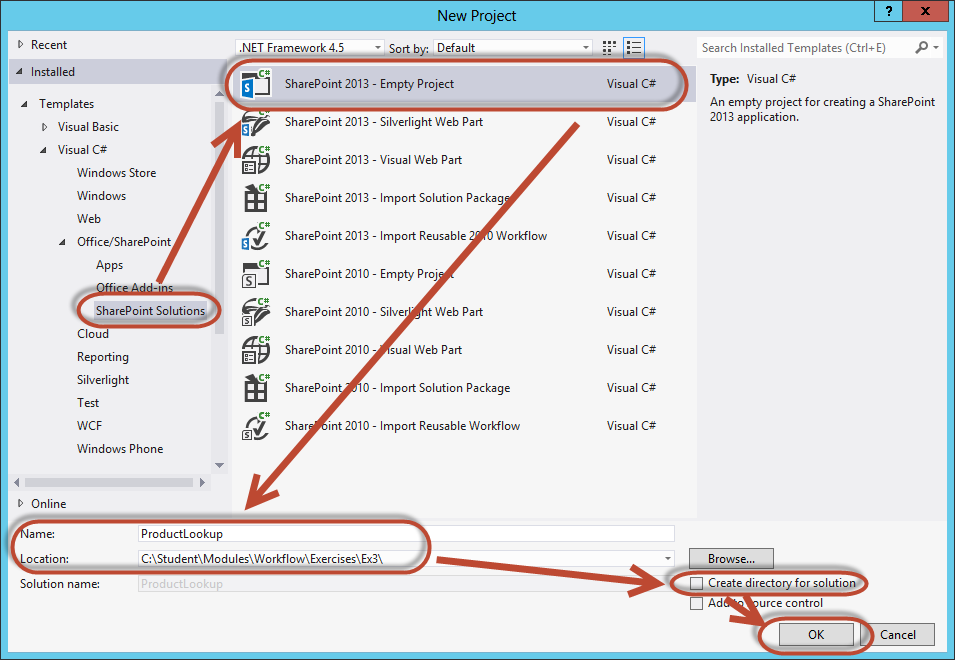
### Exercise 3: Visual Studio 2012 Workflows

In this exercise you will create a workflow using Visual Studio 2012

1. Create a new project in Visual Studio 2012:
   1. Launch **Visual Studio 2012** as administrator:
      1. **Windows Keyboard Key 🡪 Right click** on the **Visual Studio 2012** tile and select **Run as administrator**



* 1. In Visual Studio select **File 🡪 New 🡪 Project**.
  2. In the **New Project** dialog:
     1. Find the **SharePoint 2013 – Empty Project** template under the **Templates 🡪 Visual C# 🡪 Office / SharePoint 🡪 SharePoint Solutions** section.
     2. **Name**: ProductLookup
     3. **Location:** [..]\Workflow\Exercises\Ex3  
        (Where [..] represents the location of the student files (e.g. c:\student\..)
     4. **Uncheck** the **Create directory for solution** checkbox
     5. Click **OK**



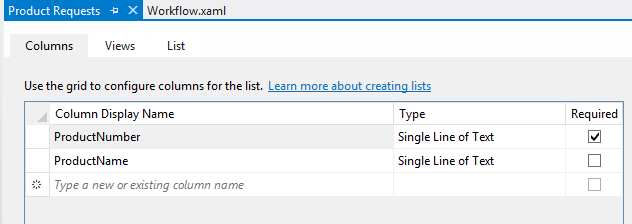
* + 1. In the **SharePoint Customization Wizard** dialog, set the following values and click **Finish**:
       1. **What SharePoint site to you want to use for debugging?** <http://workflow.wingtip.com>
       2. **What is the trust level for this SharePoint solution?** Deploy as a farm solution

#### Create a List to Store Product Requests

1. Create a new list that will store product information:
   1. Right-click the project **ProductLookup** in the **Solution Explorer** tool window and select **Add 🡪 New Item**.
   2. In the **Add New Item** dialog, select **List** from the **Office / SharePoint** category. Set the **Name** to **ProductRequests** and click **Add**.
   3. In the **SharePoint Customization Wizard** dialog, set the following values and click **Finish**:
      1. **What name do you want to display for your list?**
         1. Product Requests
      2. **Do you want to create a customizable list or a non-customizable list based on an existing list type?**
         1. **Create a customizable list template and a list instance of it:** Default (Custom List).
2. Add some columns to the new list:
   1. In the columns tab, change/add the following columns:
      1. Change **Title** column name to **ProductNumber** (single line of text) – required

Note: This should replace the existing Title field.

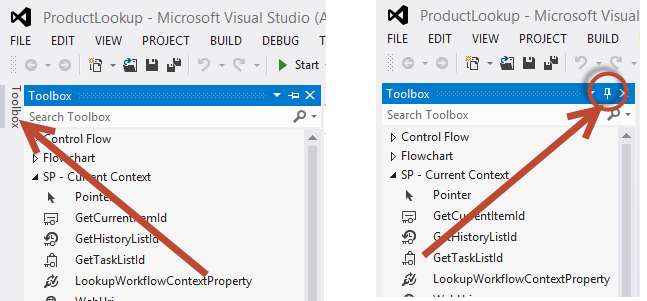
* + 1. **ProductName** (single line of text)



1. Save your changes: **File 🡪 Save All**.

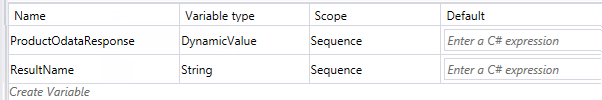
#### Create a Workflow to Retrieve Additional Data

1. Add a workflow to the project:
   1. Right-click the project **ProductLookup** in the **Solution Explorer** tool window and select **Add 🡪 New Item**.
   2. In the **Add New Item** dialog, select **Workflow** from the **Office / SharePoint** category. Set the **Name** to **UpdateProductRequest** and click **Add**.
   3. In the **SharePoint Customization Wizard** dialog, on the **Specify the workflow name for debugging**, set the following values and click **Next**:
      1. **What is the name of the workflow?** Update Product Request
      2. **What type of workflow do you want to create?** List Workflow
   4. In the **SharePoint Customization Wizard** dialog, on the **Select the lists you will use when debugging**, set the following values and click **Next**:
      1. **Would you like Visual Studio to automatically associate the workflow?** Checked
      2. **The library or list to associate your workflow with:** Product Requests
      3. **The history list to display all the events that occur while the workflow is running:** <Create New>
      4. **The task list to display the workflow tasks available to each workflow participant:** <Create New
   5. In the **SharePoint Customization Wizard** dialog, on the **Specify the conditions for how your workflow is started**, set the following values and click **Finish**:
      1. **A user manually starts the workflow**: checked
      2. **The workflow starts automatically when an item is created**: unchecked
      3. **The workflow starts automatically when an item is changed**: unchecked
2. After creating the workflow, the designer will load with an empty **Sequence** activity. Since you will be adding quite a few activities, check to see if the **Toolbox** is opened already on the left side of your screen.
   1. If it is not, make your life easier by pinning the **Toolbox** Tool Window to the environment. This is done by clicking the **Toolbox** tab in the left margin followed by the pushpin in the top-right corner of the tool window:



1. The first step is to create a handful of variables that will be used throughout the workflow to store values between activities. Add a few variables to the workflow:
   1. In the main design window, click the **Sequence** activity.
   2. At the bottom of the window, click the **Variables** tab.
   3. Add the following variables to the workflow:
      1. **ProductODataResponse**: DynamicValue \*
      2. **ResultName**: String

\*Note: Some of the variable types listed do not show up in the drop down list by default. Select the **Browse for Types…** and search for the type in these cases.



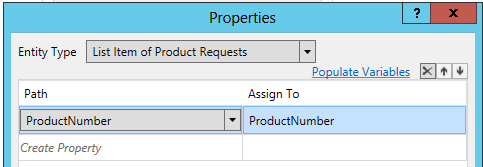
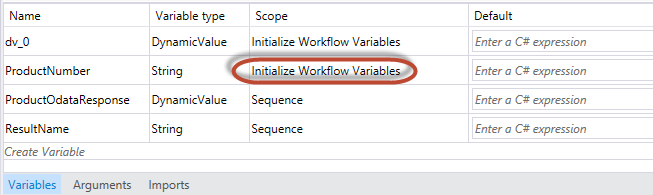
Throughout the process of creating the workflow, you may want to add multiple **WriteToHistory** activities to write log messages to the history list. This provides some contextual information to your users when the workflow is running and after it completes. The lab will not have you add any of these in the interest of time, but feel free to add as many as you like.

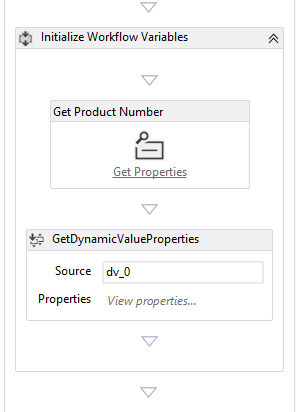
1. Next, retrieve some values from the list item that triggered the workflow:
   1. Using the **Toolbox**, add a **Sequence** activity (found in the **Control Flow** section) within the default **Sequence** activity.
      1. Select the **Sequence** activity you just added.
      2. In the **Properties Tool Window**, change the **DisplayName** property to **Initialize Workflow Variables**.

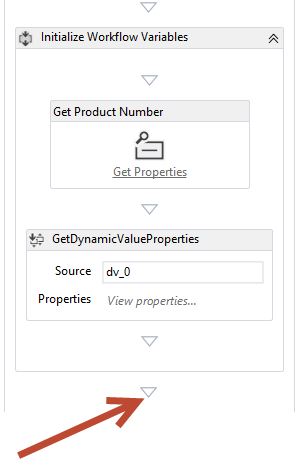
If the **Properties** Tool Window is not visible, press **[F4]**.

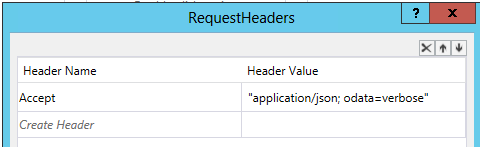
* 1. Using the **Toolbox**, find the following activity and drop it in the **Initialize Workflow Variables** Sequenceactivity:
     1. **LookupSPListItem** (found in the SP - List Item section)
  2. Next, for the **LookupSPListItem**, change its properties to the following values using the **Property** window:
     1. **LookupSPListItem**:
        1. **ItemId:** (current item)
        2. **ListId:** (current list)
        3. **DisplayName:** Get Product Number
     2. On the **LookupSPListItemProperty** click on the **Get Properties** link which will autoconfigure several items for you:
        1. This will automatically create a **DynamicValue** variable (**dv\_0**) and set the **GetProductNumber** **Result** property using this variable
        2. This will also create a **GetDynamicValueProperties** action and automatically set this action’s **Source** property to use the **dv\_0** DynamicValue variable

**Note:** this must be done to enable us to extract the **Product** Number field value from the **DynamicValue** results set generated by theLookupSPListItem action**.**

* + 1. On the **GetDynamicValueProperties** activity click on the **Define…** link
       1. Set **Entity Type**: **List Item of Product Requests**
       2. In the **Properties** dialog box double click on **Create Property** and set:
          1. **Path:** Product Number
       3. Click on the **Populate Variables** link  
          (Note: this will auto create a new variable for each row of the appropriate type and sets them in the Assign To column.)
       4. Your **Properties** dialog box should now look like this:  
          
       5. Click **OK**
  1. Back on the Workflow Design surface, at the bottom of the window, click the **Variables** tab (as needed) to show us the workflow variables:  
     
  2. You will see that two new variables were created **dv\_0** and **ProductNumber**.
  3. We need to modify the Scope for the ProductNumber variable as we will need this value later in this process.
     1. In the **Product Number Scope** column select the **dropdown** arrow and change the **Scope** to **Sequence**
  4. Your workflow should now look like the following figure:



1. With the values from the list item in the workflow, the next step is to query a web service for information about the product.
   1. Add a new **Sequence** activity to the end of the workflow, after the **Initialize Workflow Variables** sequence you just completed (but still inside of the containing **Sequence** activity as shown in the image below), and set its **DisplayName** to **Retrieve Product Details**.  
      
   2. Within the **Retrieve Product Details** sequence, add an **HttpSend** activity (found in the **Messaging** section).
   3. Next, update the properties on the web service (Http Send) activity you just added. This activity will request data from a sample OData service you deployed in a previous exercise. This service is an OData front-end to the sample AdventureWorks2012 database that contains product information.
      1. **DisplayName:** Call AW Product OData Service**.**
      2. **Method**: GET.
      3. Add a **Request Header** to specify the type of data to retrieve:
         1. Click the **…** button (ellipsis) for the **RequestHeaders** property to display the Request Headers dialog
         2. Double-click the ***Create Header*** in the **Header Name** section and select **Accept** from the dropdown
         3. Enter a value of: **"application/json; odata=verbose"**  
            (Note: make sure you have the quotes around the whole string as in the image below)



* + - 1. Click **OK** to close the Request Headers dialog box
    1. Uri:

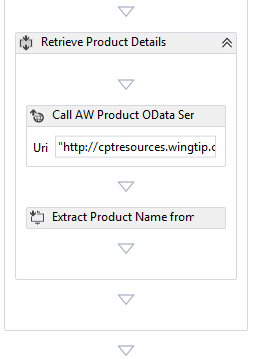
"http://cptresources.wingtip.com:81/services/AdventureWorks2012Product.svc/Products?$filter=ProductNumber eq '" + ProductNumber +"'"

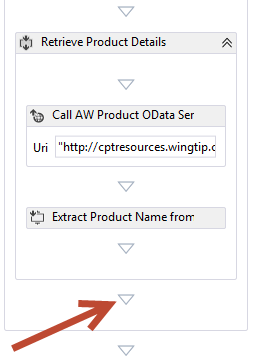
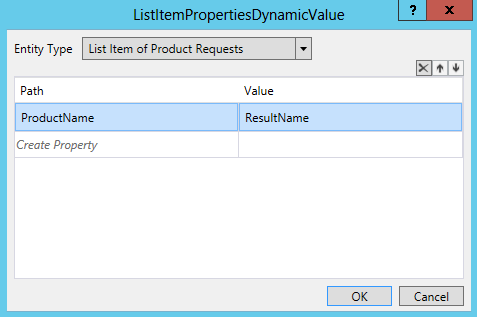
**(Note:** Make sure to keep all of the quotes exactly as displayed here, also this entry will be a single line [there is no carriage return or line feed in this expression])

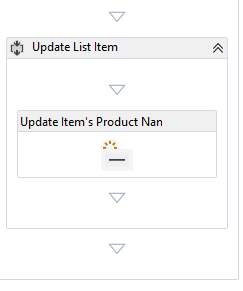
This query will issue a query to the OData service for all product information where the ProductNumber is equal to the value entered in the list item.

After calling the service, you need to extract some information from the response.

* + 1. **ResponseContent**: ProductODataResponse
  1. Add a **GetDynamicValueProperty<T>** activity (found in the **Dynamic Value** section) after the **Call AW Product OData Service** activity. When prompted for a value of **T**, select **String** and click
     1. Rename this activity to **Extract Product Name from Response**. (i.e. set the Display Name property)
     2. Set the following property values on the **Extract Product Name from Response** activity:
        1. **PropertyName**: “d/(0)/Name”  
           (Note: be sure to put this in double quotes (“) and the 0 is a Zero not a letter O)
        2. **Result**: ResultName
        3. **Source**: ProductODataResponse
  2. Your **Retrieve Product Details** sequence activity should now look like the following figure:



1. The last step in the workflow is to update the list item. Add a new Sequence activity to the end of the workflow, just after the **Retrieve Product Details** sequence you just completed, and set its **DisplayName** to **Update List Item**.  
   
   1. Add a **UpdateListItem** activity (found in the **SP –List** section) to the **Update List Item** sequence and set the following properties:
      1. **ItemId:** (current item)
      2. **ListId:** (current list)
      3. **DisplayName:** Update Item's Product Name
      4. **ListItemPropertiesDynamicValueDisplayName**: Click the build button and set:
         1. **Entity Type:** List Item of Product Requests
         2. **Path**: ProductName
         3. **Value**: ResultName  
            
         4. Click **OK**
   2. Your **Update List Item** sequence activity should now look like the following figure:



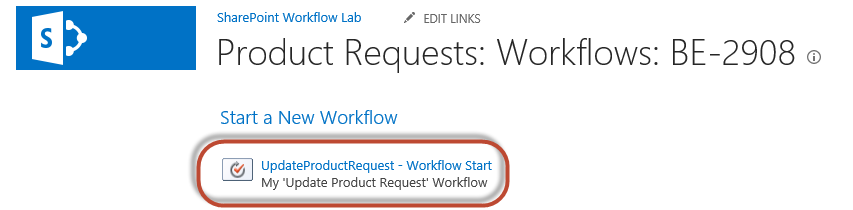
1. Save the workflow: **File 🡪 Save All**.

#### Deploy and Test the Workflow

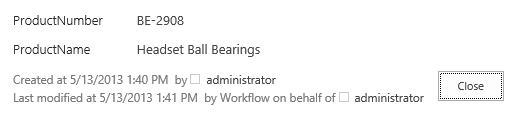
1. Deploy the app to SharePoint by pressing **[F5]** or selecting the menu **Debug 🡪 Start Debugging**.

This might take a moment… a console window will appear that starts with **Listening at http…**. Leave that window open as it is running a utility program that the workflow is using to ensure the necessary components are running on the server. You can see the status of the installation by clicking the Output tab found at the bottom of the Visual Studio 2012 window. The deployment process writes a message to the list every second it is running.

1. If you receive a **Deployment Conflicts** dialog click **Resolve Automatically**
2. If you receive an **Attach Security Warning** dialog, click **Attach**.
3. When the homepage of the workflow site loads, click **Product Requests** in the Quick Launch navigation.
4. Add a new item:
   1. Set the **Product Number** field to **BE-2908**.  
      (note the dash between BE and 2908)
   2. Leave the **Name** field blank.
   3. Click **Save**
5. After creating the item, select it to go to the detail page.
   1. Click on the Hyperlinked title: **BE-2908**
6. Within the **View** tab in the ribbon, click **Workflows**.
7. On the **Product Requests: Workflows: BE-2908**, click the **UpdateProductRequest – WorkflowStart** link.



1. When the **Product Requests** page loads, select the **BE-2908** item (i.e. click on the Hyperlinked title again).
2. Within the **View** tab in the ribbon, click **Workflows**.
3. Notice the status page will show both running and completed workflows, Click the status link for the workflow you just started to see the status page with the logging information.
4. If it hasn’t completed by now, keep refreshing the page every few seconds to see if it has completed. Once it completes you should be able to see the **Internal Status** field change to **Completed**, the ProductName has been updated, and the logging information for the workflow execution.
5. Return to the item by clicking on the **BE-2908** hyperlink to the right of **Item:** on the **Workflow Status: UpdateProductRequest – Workflow Start** screen.
6. The following figure shows the updated item:



In this exercise you created a new workflow using Visual Studio that called an OData service and updated a list item.