

# Create a Workflow to Archive Documents Using Word Automation Services in SharePoint 2010

(Microsoft Corporation)

Word Automation Services enables server-side conversion of Word documents on SharePoint Server 2010. This walkthrough shows how to create a SharePoint Server 2010 Workflow that archives a Word document to an XPS file based on the status of the document.

During the walkthrough, you learn the following tasks:

1. Set up a Microsoft Visual Studio 2010 project to develop a Word Automation Services solution.
2. Design a SharePoint Server 2010 Workflow.

## Prerequisites

To successfully complete the walkthrough, you must have the following:

1. A computer running both SharePoint Server 2010 Enterprise or Standard Edition and Microsoft Visual Studio 2010.
2. Word Automation Services enabled on the SharePoint Server 2010 farm.

## Step 1: Set Up the Visual Studio 2010 Project for the Workflow Archive to XPS

This topic shows how to modify a SharePoint Server 2010 content type and set up the Microsoft Visual Studio 2010 project for the Word Automation Services solution.

## Setting Up the Project

In the first procedure of this topic, you configure a SharePoint Server 2010 document library for the input and output documents of a Word Automation Services conversion job.

### To modify the Document Content Type

1. From the SharePoint site where the Document Library is located, click **Site Actions** and then click **Site Settings**.

**Note:** This walkthrough assumes that you have created a SharePoint site that contains a Shared Documents library. This library will be referenced in the Creating the Project procedure later in this topic.

2. Under the **Galleries**, click **Site content types**.
3. Under **Document Content Types**, click **Document** to edit its content type. The Word Automation Services solution that you create in this walkthrough uses the Document content type.
4. Under **Columns**, click **Add from existing site columns**.
5. In **Available Columns**, scroll to **Status**. Select **Status** and then click **Add**.

**Note:** If **Status** does not appear in **Available Columns**, it may have already been added to the Document content type.

6. Click **OK** to add the **Status** column to the Document content type.

After modifying the Document content type, the next step is to create the Visual Studio 2010 project for the Word Automation Services solution.

# Create a Workflow to Archive Documents Using Word Automation Services in SharePoint 2010

(Microsoft Corporation)

## To create the Visual Studio 2010 Sequential Workflow project

1. Start Visual Studio 2010 as an administrator.

**Important:** You must start Visual Studio 2010 as an administrator in order to successfully publish the project to the SharePoint 2010 farm.

2. Click **File**, point to **New**, and then click **Project...**
3. In the **New Project** dialog box, under **Installed Templates**, expand **Visual C#** (if it is not already expanded), expand **SharePoint**, and then click **2010**.
4. In the list of template types, select **Sequential Workflow**.
5. Give the project a name, for example, ArchiveDocument.
6. Click **OK** to create the project.
7. On the **Specify the site and security level for debugging** page of the **SharePoint Customization Wizard** verify the SharePoint site for the solution.
8. On the **Specify the site and security level for debugging** page of the **SharePoint Customization Wizard**, select **Deploy as a farm solution**.

**Important:** You must select the option **Deploy as a farm solution** because the Word Automation Services object model cannot be accessed from partly trusted code.

9. On the **Specify the workflow name for debugging** page of the **SharePoint Customization Wizard**, give the workflow an easily recognizable name, for example, Archive Document.
10. Select **List Workflow** for the workflow template and then click **Next**.
11. On the **Select the lists you will use when you debug** page of the **SharePoint Customization Wizard**, leave the default associations and then click **Next**.

**Note:** If you want to attach the workflow to a specific list on the SharePoint site, select that list on **The library or list to associate your workflow with:**. This walkthrough associates the workflow with a Shared Documents library.

12. On the **Specify the conditions for how your workflow is started** page of the **SharePoint Customization Wizard**, leave the default settings under **How do you want the workflow to start?**.
13. Click **Finish** to create the project.

After creating the Visual Studio 2010 project, the next step is to write code to create a workflow that performs a Word Automation Services document conversion. To use the Word Automation Services object model in code, add a reference to the Word Automation Services assembly.

## To add a reference to Word Automation Services

1. In **Solution Explorer**, right-click **References** and then click **Add Reference...**

# Create a Workflow to Archive Documents Using Word Automation Services in SharePoint 2010

(Microsoft Corporation)

2. In the **Add Reference** dialog box, click the **Browse** tab. The `Microsoft.Office.Word.Server.dll` assembly is located in the SharePoint Server 2010 \ISAPI folder, which usually has the following path: `%systemdrive%:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\14\ISAPI`
3. Select **Microsoft.Office.Word.Server.dll** and then click **OK** to add the reference.

**Note:** If you receive a warning that `Microsoft.Office.Word.Server.dll` requires a newer version of the .NET Framework, click **Yes** to add the reference. Then, add a reference to `System.Web.DataVisualization`, which usually has the following path:

`%windir%\Assembly\GAC_MSIL\System.Web.DataVisualization\3.5.0.0__31bf3856ad364e35\`

## Step 2: Design the Workflow

This topic shows how to design a workflow that initiates a Word Automation Services conversion job.

### Designing the Workflow

The solution that you create in this walkthrough initiates a Word Automation Services conversion job based on the status of a document in a document library and then archives the converted document. When the document status is changed to **Final**, a SharePoint workflow begins that creates a Word Automation Services conversion job and then places the document that was output from the conversion into the same document library as the input document.

The workflow consists of two steps with an optional third step: the first step checks the `Status` property of the input document has been set to **Final**; if so, the second step converts the document. The third step monitors the conversion job through completion.

### Checking the Document Status Property

The first step of the workflow checks the **Status** property of the document in the input library.

#### To check the document status

1. In the Visual Studio 2010 project that you created in Step 1 of this walkthrough, add a **While** activity after the existing **onWorkflowActivated** activity. The **While** activity is under **Windows Workflow v3.0** in the **Toolbox**.
2. From the **Toolbox**, under **SharePoint Workflow**, add an **OnWorkflowItemChanged** activity inside the **While** activity.
3. Click the activity. From the **Properties** window, set the value of the **CorrelationToken** property to **workflowToken** (the only item in the drop-down list). This associates the **ItemChanged** event with the correct workflow.
4. In the **Properties** window, click the lightning bolt to switch to **Events** view, and press ENTER in the **Invoked** method to create a blank event handler.
5. In the code window, add a global variable to the class to track whether the **Status** property has been set to the desired value.

```
bool documentReady = false;
```

# Create a Workflow to Archive Documents Using Word Automation Services in SharePoint 2010

(Microsoft Corporation)

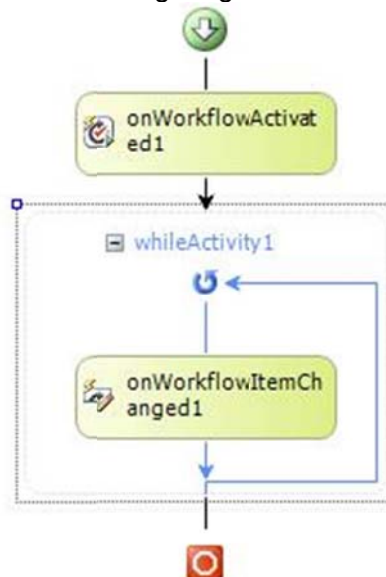
6. In the event handler for the `onWorkflowItemChanged` event, add code to check the value of the **Status** column, and set the `documentReady` variable appropriately.

```
//Check if the Status is set to "Final"; if so, the document is ready to archive if
(workflowProperties.Item["Status"] != null)
{
    documentReady = ((string)workflowProperties.Item["Status"] == "Final");
}
```

7. In the workflow designer window, click on the **While** activity. In the **Properties** window, click the drop-down next to **Condition** and then select **Declarative Rule Condition**.
8. Expand **Condition**, exposing **ConditionName** and **Expression**.
9. In the **ConditionName** field, enter a name for the condition, for example, Ready to Archive.
10. In the **Expression** field, click the ellipsis button next to **Condition Expression**.
11. In the **Rule Condition Editor** dialog, add code that sets the condition.

```
documentReady == false
```

12. Click **OK** to save the condition.
13. The resulting workflow now looks like the following diagram.



## Converting the Document

When the document **Status** property is set to Final, the second step of the workflow converts the document using Word Automation Services.

# Create a Workflow to Archive Documents Using Word Automation Services in SharePoint 2010

(Microsoft Corporation)

## To convert the document

1. In the code page for the workflow, add a using directive for the namespace that contains the Word Automation Services object model that you use to perform document conversions.

```
using Microsoft.Office.Word.Server.Conversions;
```

2. In the workflow designer page, add a **Code** activity from the Visual Studio 2010 **Toolbox** after the existing **While** activity.
3. With the **Code** activity selected in the workflow designer page, click the lightning bolt in **Properties** to switch to the Events view, and then press **ENTER** in the **ExecuteCode** field to create an empty event handler.
4. In the resulting empty event handler for the ExecuteCode event, add code to perform the document conversion.

```
// Create a Word Automation Services conversion job
ConversionJob job = new ConversionJob("Word Automation Services");

// Specify conversion settings
job.UserToken = workflowProperties.OriginatorUser.UserToken;
if (workflowProperties.Site.SiteSubscription != null)
    a. job.SubscriptionId = workflowProperties.Site.SiteSubscription.Id;
job.Settings.OutputFormat = SaveFormat.XPS;
job.Name = "Archive Document Workflow";

// Add the filestring fileUrl = workflowProperties.WebUrl + "/" + workflowProperties.ItemUrl;
job.AddFile(fileUrl, Path.ChangeExtension(fileUrl, "xps"));

// Schedule the conversion
job.Start();
```

5. This code creates a new `ConversionJob1`, which defines a set of conversions, by calling the constructor for the `ConversionJob1` that takes the name of the service application as an argument.
6. **Note:** This example assumes the name of the service application is Word Automation Services, which is the default name when it is created using the SharePoint Server 2010 Farm Configuration Wizard. If you have used a different name for the service application, use that name as the argument for the `ConversionJob1` constructor in the preceding code example.

Next, the code sets the necessary settings for document conversion:

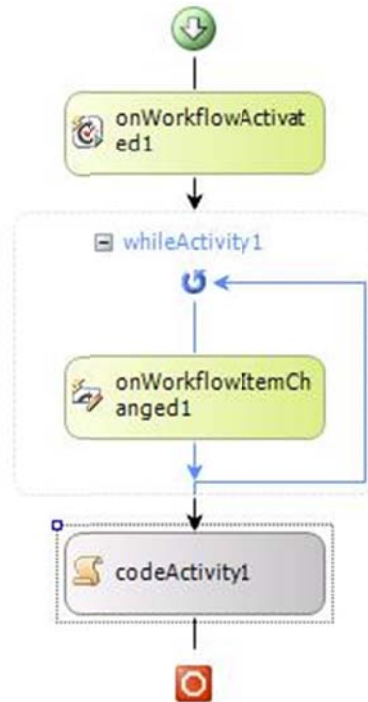
- `UserToken2` - specifies the user credentials used to read the input file and to write the output file.
- `SubscriptionId3` - specifies the site subscription ID of the site.

# Create a Workflow to Archive Documents Using Word Automation Services in SharePoint 2010

(Microsoft Corporation)

**Note:** This parameter is only required if the SharePoint farm has been configured in partitioned mode.

- OutputFormat4 - specifies the output file format for the conversion.
- In addition, the code specifies one optional setting for the conversion:
- Name5 - specifies a friendly name for the conversion.
- The resulting workflow now looks like the following diagram.



## Monitoring the Conversion

Once a document conversion is scheduled, it is processed asynchronously. Optionally, you can add a step to the solution to monitor the result of the conversion before terminating the workflow, so that the workflow completes after the conversion.

### To monitor the document conversion

1. In the workflow designer page, add a **While** activity from the Visual Studio 2010 **Toolbox** after the existing **Code** activity.
2. From the **Toolbox**, under **Windows Workflow v3.0**, add a **Sequence** activity inside the **While** activity.
3. From the **Toolbox**, under **Windows Workflow v3.0**, add a **Delay** activity inside the **Sequence** activity.
4. With the **Delay** activity selected in the workflow designer page, in the **Properties** window, click on the **TimeoutDuration** field and set it to 00:00:30 (30 seconds). This means that the workflow will be re-checked at most every 30 seconds.

# Create a Workflow to Archive Documents Using Word Automation Services in SharePoint 2010

(Microsoft Corporation)

**Note:** From the **Toolbox**, under **Windows Workflow v3.0**, add a **Code** activity immediately after the existing **Delay** activity.

5. With the new **Code** activity selected in the workflow designer page, click the lightning bolt in **Properties** to switch to the Events view, and then press **ENTER** in the **ExecuteCode** field to create an empty event handler.
6. In the code page for the workflow, add a using directive for the namespace that contains the `ReadOnlyCollection<T>`<sup>6</sup> class used later in this procedure.

```
using System.Collections.ObjectModel;
```

7. Add two global variables to the class: a Guid value to store the JobId<sup>7</sup> of the conversion job, and a Boolean value to track whether the conversion completed successfully.

```
Guid jobId;  
bool conversionComplete = false;
```

8. In the first code method, codeActivity1\_ExecuteCode, store the JobId<sup>7</sup> of the conversion job in the jobId variable by adding the following code after the Start method.

```
// Schedule the conversion  
job.Start();  
  
// Save the conversion job ID  
jobId = job.JobId;
```

9. In the new event handler created in this procedure, codeActivity2\_ExecuteCode, add code to check the status of the conversion.

```
// Monitor the conversion// Get the site subscription ID  
Guid? siteSubscription = null;  
if (workflowProperties.Site.SiteSubscription != null)  
    siteSubscription = workflowProperties.Site.SiteSubscription.Id;  
  
// Check if the conversion is complete; if it is, refresh  
ConversionJobStatus status = new ConversionJobStatus("Word Automation Services", jobId, siteSubscription);  
  
if (status.Count == status.Succeeded)  
{  
    // Success!  
    conversionComplete = true;  
}  
elseif (status.Count == status.Failed)  
{
```

# Create a Workflow to Archive Documents Using Word Automation Services in SharePoint 2010

(Microsoft Corporation)

```
// Conversion failed
ReadOnlyCollection<ConversionItemInfo> failedItems = status.GetItems(
ItemTypes.Failed);
    thrownew Exception(failedItems[0].ErrorMessage);
}
elseif (status.Count == status.Canceled)
{
    // Conversion was canceledthrownew Exception("Conversion was canceled.
");
}

// Since the conversion is not yet completed, keep waiting
```

The code checks the status of the conversion using the `ConversionJobStatus8` object.

**Note:** This example assumes the name of the service application is Word Automation Services, which is the default name when it is created using the SharePoint Server 2010 Farm Configuration Wizard. If you have used a different name for the service application, use that name as the argument for the `ConversionJobStatus8` constructor in the preceding code snippet.

The code checks the conversion job status to determine whether the item has been successfully converted or not, and reacts accordingly. Otherwise, the code takes no action because the conversion is in progress.

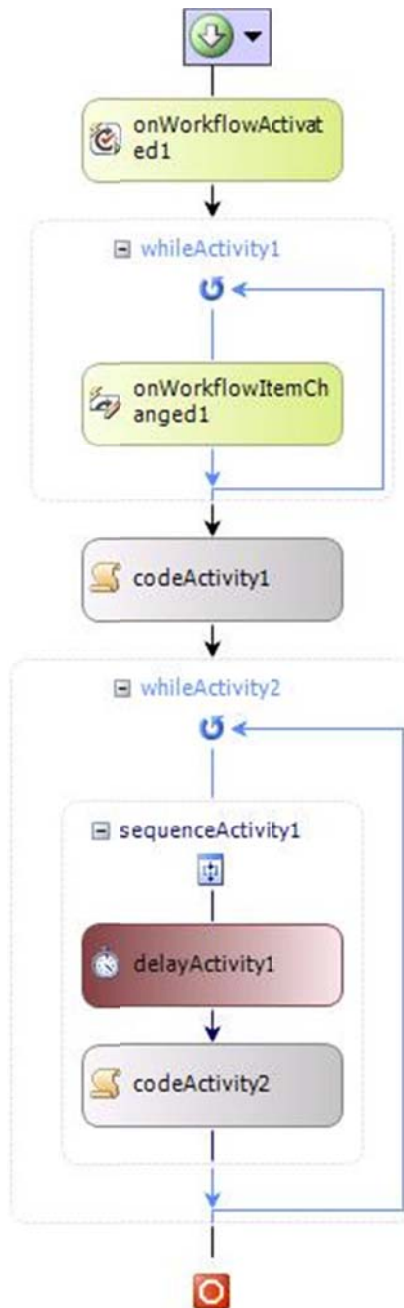
10. With the **While** activity selected in the workflow designer, in the **Properties** window, click the **Condition** field and then select **Declarative Rule Condition**.
11. Expand **Condition**, exposing **ConditionName** and **Expression**.
12. In the **ConditionName** field, enter a name, for example, Conversion Complete.
13. In the **Expression** field, click the ellipsis button next to **Condition Expression**.
14. In the **Rule Condition Editor** dialog, add code that sets the condition.

```
conversionComplete == false
```

15. Click **OK** to save the condition.
16. In the workflow designer, the final workflow should look like the following diagram.

# Create a Workflow to Archive Documents Using Word Automation Services in SharePoint 2010

(Microsoft Corporation)



## Step 3: Build and Deploy the Workflow Solution

In this topic you build and deploy a SharePoint Server 2010 solution that uses Word Automation Services.

### Building and Deploying the Solution

Once you have followed the previous steps in this walkthrough, you can build and deploy the resulting solution.

# Create a Workflow to Archive Documents Using Word Automation Services in SharePoint 2010

(Microsoft Corporation)

## To build and deploy the solution

1. With the **ArchiveDocument** solution selected in **Solution Explorer** in Visual Studio 2010, click **Build** and then click **Deploy Solution**.
2. On the SharePoint site where you deployed the solution, navigate to the source (input) list, for example, the Shared Documents library, and then upload a new Microsoft Word document to initiate the workflow.

**Note:** If you attached the workflow to a different list, navigate to that list.

3. Click the drop-down next to the new document and then click **Edit Properties**.
4. Set **Status** to **Final**.
5. Click **OK**.

When the timer job runs and the conversion is processed, an XPS file appears next to the original document (refresh the page to see the result).