­­­

1. Windows Server 2008 R2 Developer Lab Series
   1. File Classification Infrastructure with SharePoint
2. Lab version: 1.0.0
3. Last updated: 7/7/2010
4. 

Contents

Background 3

Introduction 4

Classify files on FCI 5

Adding References 11

Uploading the Files 20

Lab Summary 23

# Background

* 1. Windows Server 2008 R2 includes the File Classification Infrastructure feature set that provides insight into files to help manage data more effectively, reduce costs and mitigate risks by providing a built-in solution for file classification allowing administrators to automate manual processes with predefined policies based on the data’s business value.
  2. FCI includes file management tasks that allow IT administrators to automate tasks such as expiring stale data or migrating files from File Servers to SharePoint.
  3. By using the File Classification Infrastructure you can create properties with different types that are very familiar to developers (boolean, numeric, dates and times, among others); these properties are defined at the Operating System level. SharePoint also lets you create properties for the files in its libraries, but these properties are defined at the SharePoint level. Therefore, even if an FCI property has the exact same name as a SharePoint property, it doesn’t mean that SharePoint will relate them on its own. It becomes the choice and responsibility of the Administrators to determine how these properties should be handled.
  4. In order to synchronize these two types of properties, Microsoft provides the FCI SharePoint Upload PowerShell script. This small script is built to work in conjunction with the FCI file management tasks (Customizing File ­Management Tasks) to automatically upload files and their classification properties to SharePoint .
  5. The script supports uploading any file type. If the SharePoint site has properties with names matching the classification properties in the File Server, the script will set these properties accordingly.
  6. For example, suppose you have a TIFF file that has an FCI property called “Secrecy” and its value has been set to “High”. If the SharePoint library to which the file is uploaded contains a column for “Secrecy”, the script will set the “High” value for the “Secrecy” property in the TIFF uploaded to SharePoint.
  7. The PowerShell script does not have any special cmdlets to complete this task; it simply uses a combination of components that you could access from your .NET applications. This document is a quick guide on how to start using those components to extend your applications and work with SharePoint and FCI.

# Introduction

In this document we will review a small C# application that uploads files to a SharePoint site, and that also replicates the values of their FCI properties whenever a SharePoint property of the same name exists.

Basically this involves two different components:

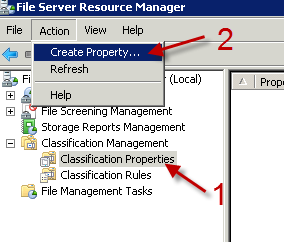
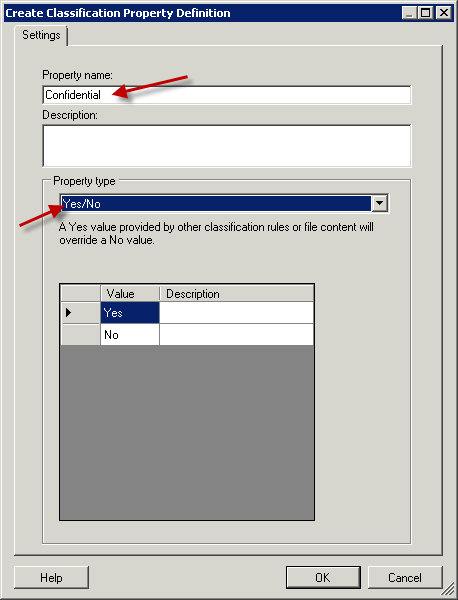
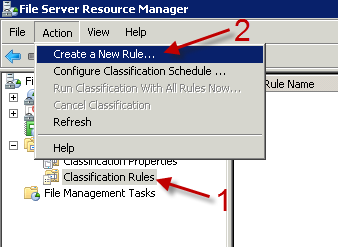
* FSRM API: The File Server Resource Manager comes with a COM API that lets you programmatically work with the classification properties in the files, among many other things.
* SharePoint Web services: SharePoint sites have multiple files supplying web services that can be accessed through many platforms and programming languages.

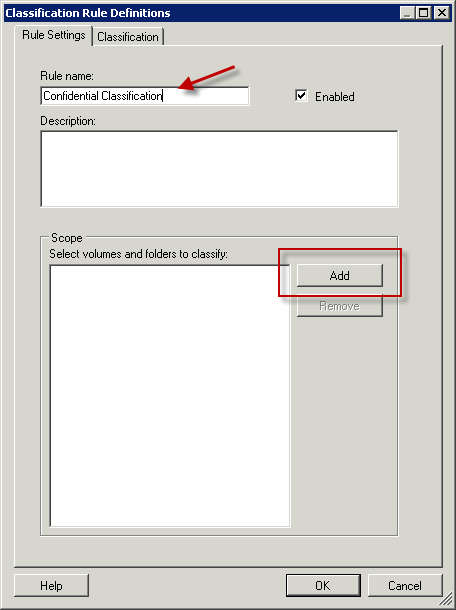
In our application we will use the FSRM API to query the classification properties in the files that we upload to the SharePoint library.

Additionally, SharePoint has a “copy” web service that allows you to programmatically upload files.

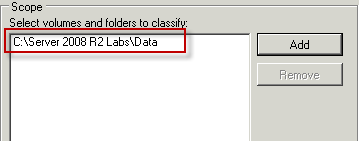
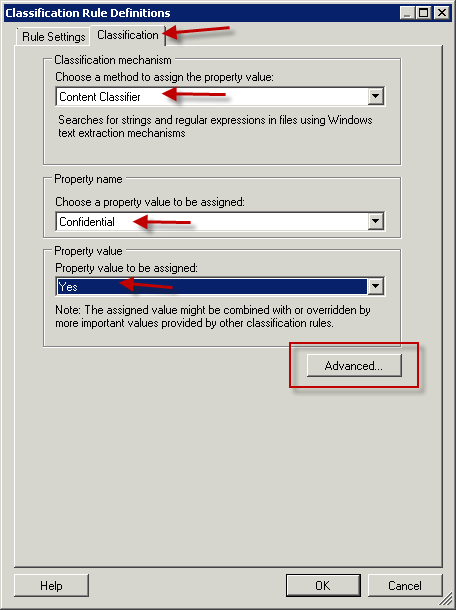
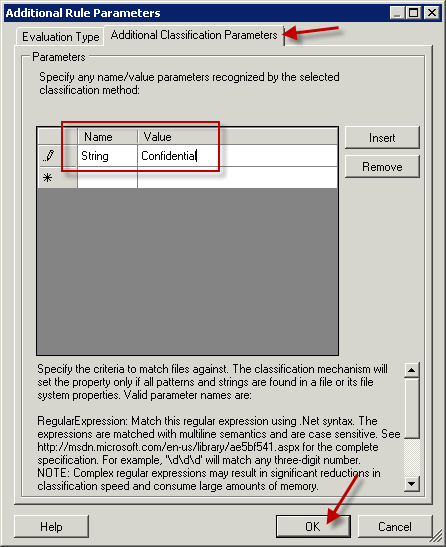
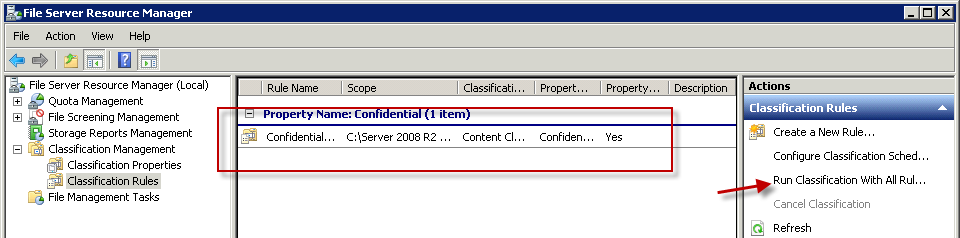
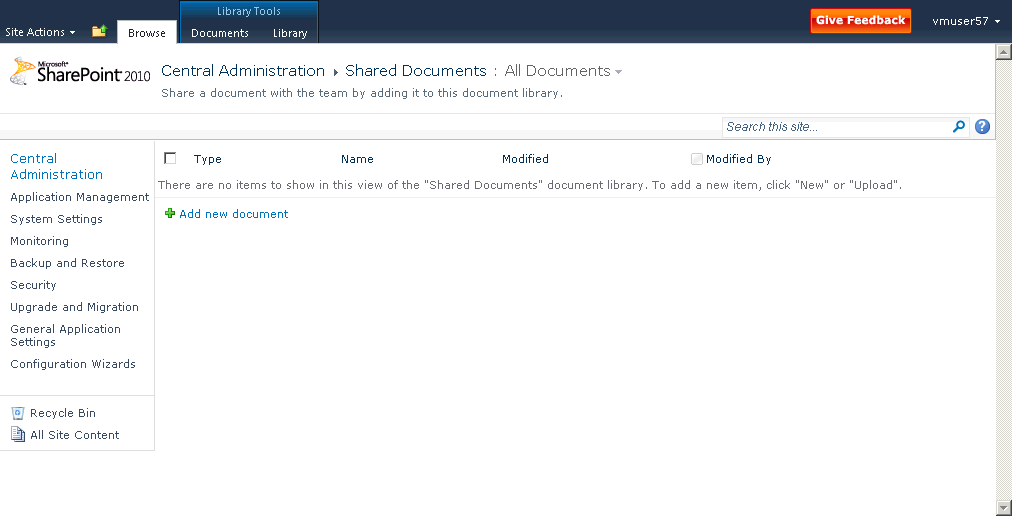
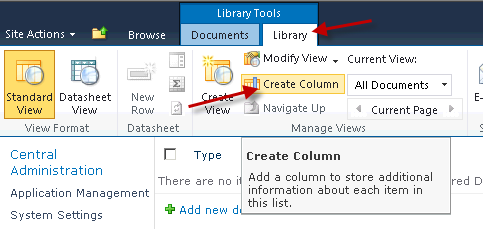
After reviewing all the necessary steps to use these components, you may choose to place this business logic in a Windows Service, Windows Forms application, Web Application, or a Web Service of your own. These implementations will be very similar in any type of .NET project.

# Classify files on FCI

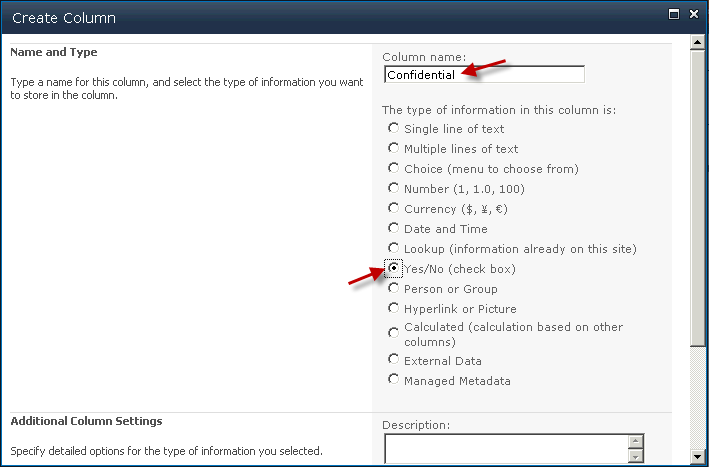
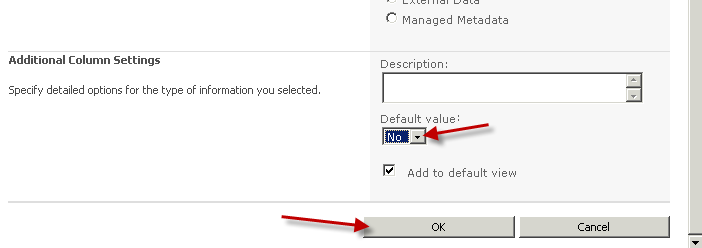
1. Open **Control Panel > Administrative Tools > File Server Resource Manager**.
2. In the **File Server Resource Manager** expand on the **Classification Management > Classification Properties** node in the panel on the left.
3. In the **Action** menu, select the **Create Property** option.
   1. 
   2. Creating a property in the File Server Resource Manager.
4. In the **Create Property** window, define a property with the following specs:
   1. Property name: Confidential.
   2. Property type: Yes/No.
   3. You can optionally add a description.
   4. 
   5. Defining the Confidential property as a Yes/No field.
5. Click **OK** to add the property to the list.
6. Next, click on the **Classification Manager > Classification Rules** node on the panel on the left.
7. In the **Action** menu, select the **Create a New Rule** option.
   1. 
   2. Creating a new rule in the File Server Resource Manager.
8. In the **Rule name** box type **Confidential Classification**.



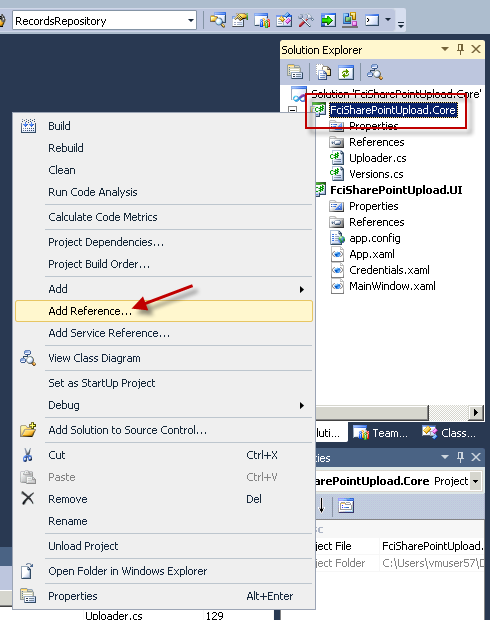
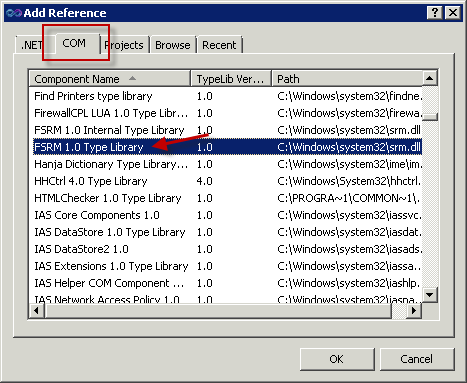
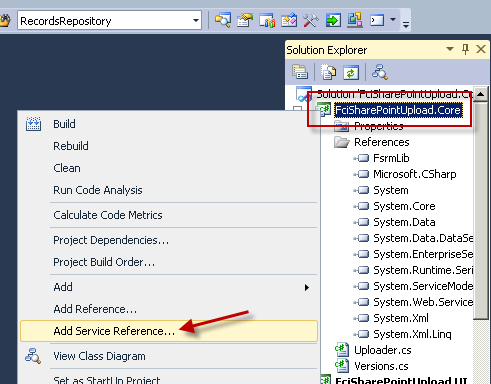
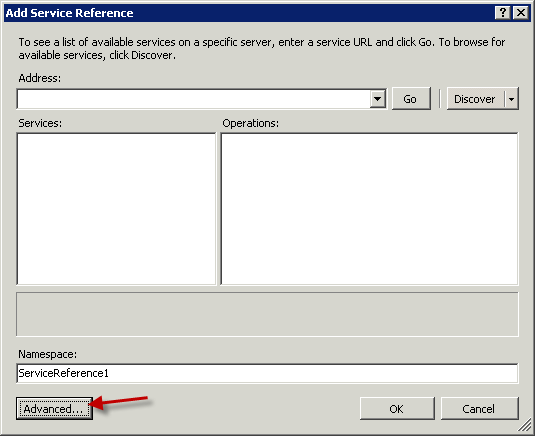
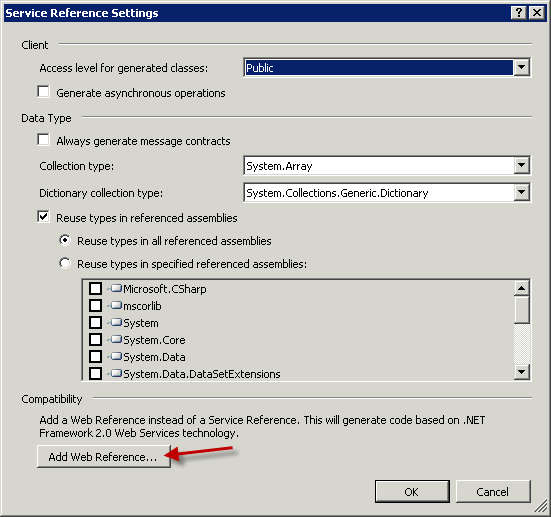
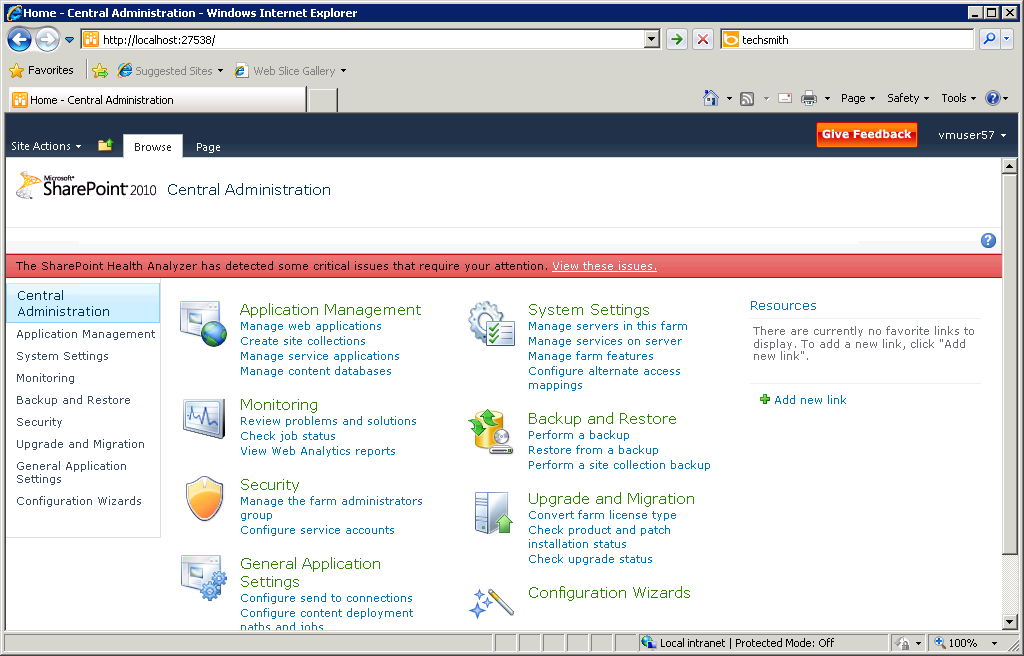
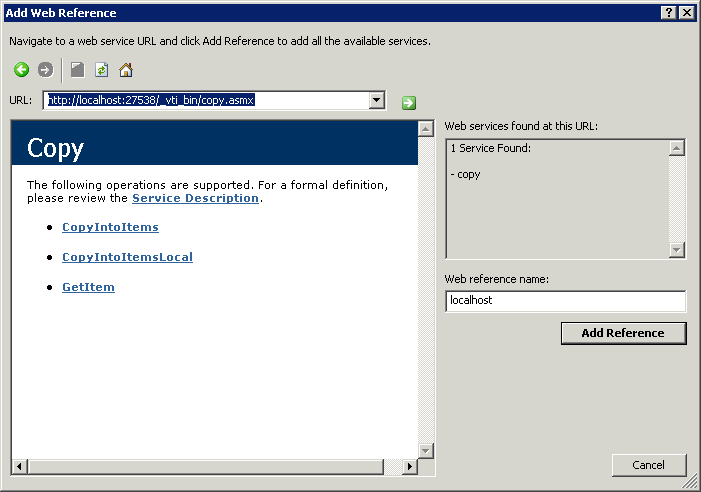
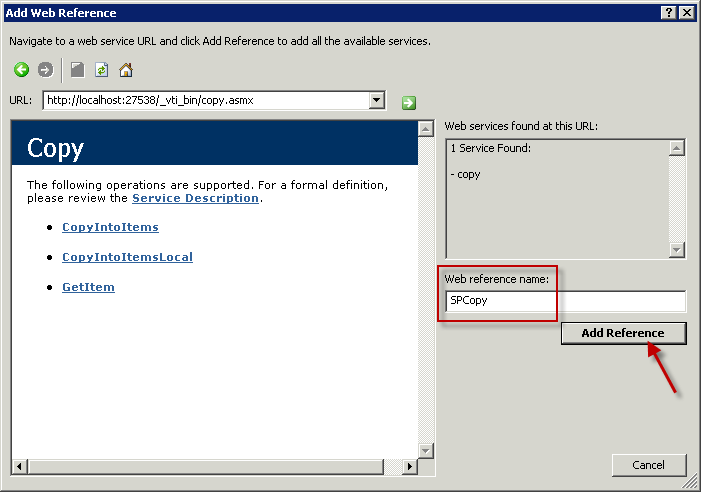
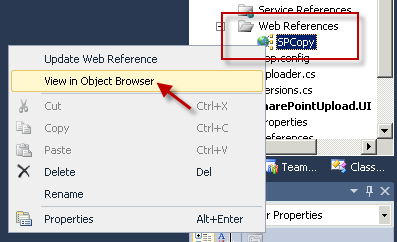
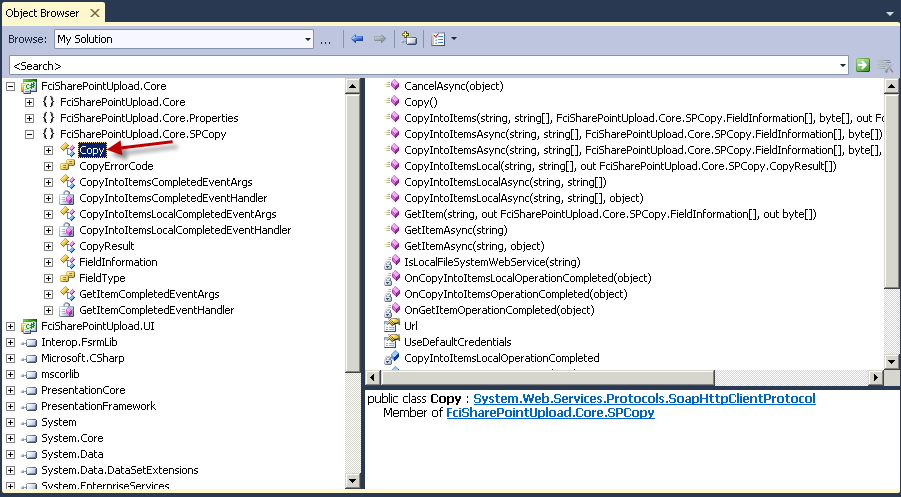
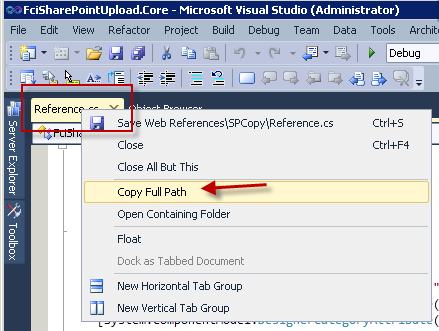
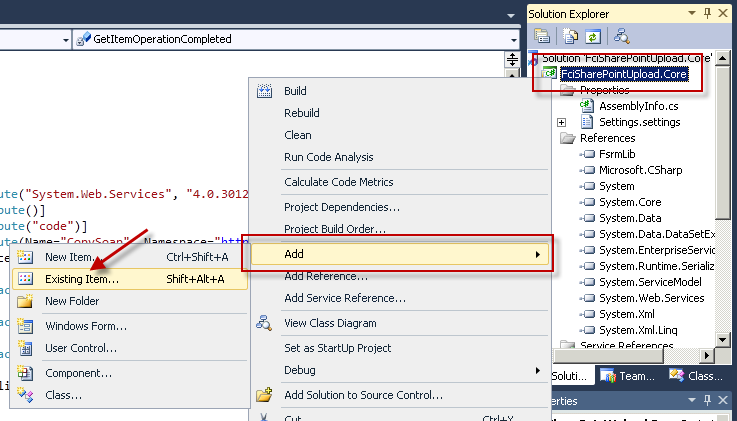
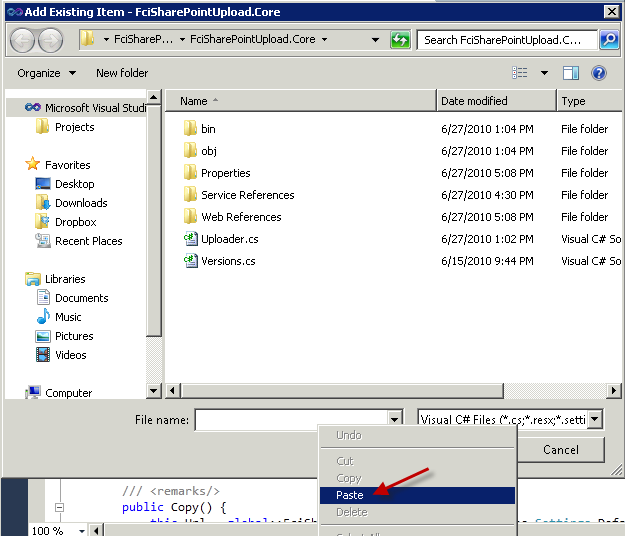
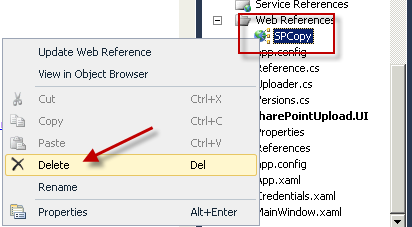
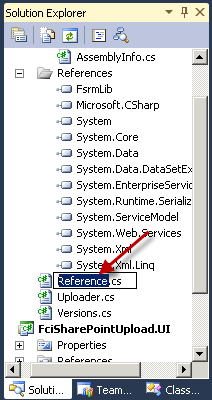
* 1. Defining the Classify Windows Sounds rule.

1. Click on the **Add** button in the **Scope** section and browse for the folder **C:\Server 2008 R2 Labs\****FCI-SharePoint Lab\Data**.
   1. 
   2. Selecting the root folder where the rule will be applied.
2. Switch to the **Classification** tab in the **Classification Rule Definitions** window.
3. Choose the following options:
   1. Method to assign the property value: Content Classifier.
   2. Property name: Confidential.
   3. Property value to be assigned: Yes.
   4. 
   5. Defining the classification settings for the rule.
4. Click on the **Advanced** button.
5. Go to the **Additional Classification Parameters** tab.
6. Under the Name column type **String**, under value type **Confidential** and click **OK**.
   1. 
   2. Makes the rule look for the word “Confidential” in the content of the documents.
7. Click **OK** again to commit the changes on this new rule.
8. Click on the **Run Classification With All Rules Now** button on the right.
   1. 
   2. Shows how the rule was added and how to run it now.
9. Open up your **SharePoint library** in Internet Explorer (should be http://fci-sp-test:20310/fci/Library/Forms/AllItems.aspx)
   1. 
   2. Sample SharePoint Library
10. Click on the **Library** tab in the **Library Tools** section.
11. Click on the **Create Column** button and wait while the content shows. This may take several seconds.
    1. 

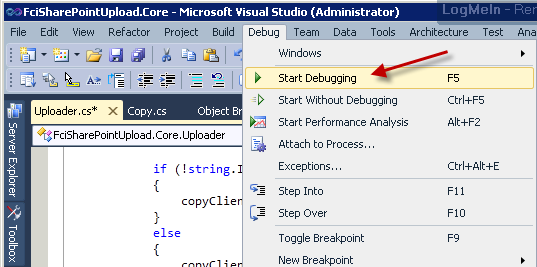
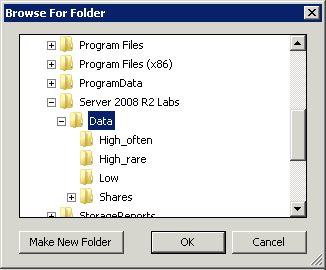
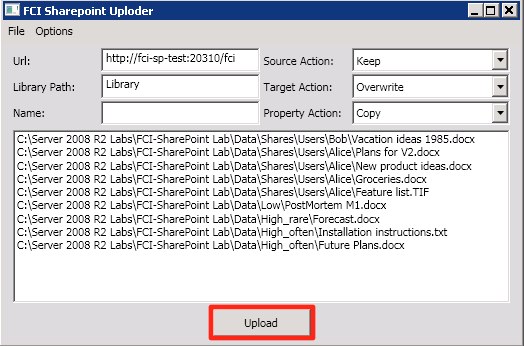
Creating a column in a SharePoint library.

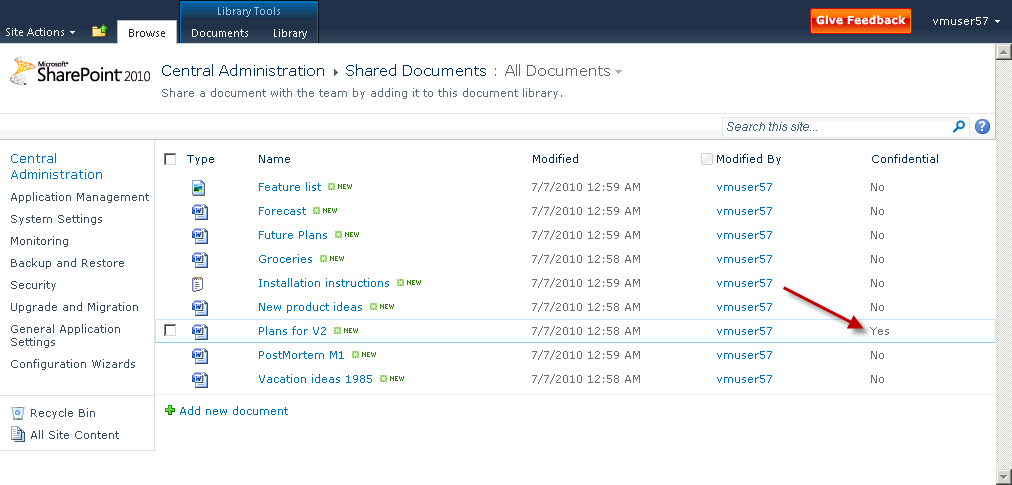
1. In the **Column name** field you will need to use the exact same name of the property in FSRM. In this case you will need to type **Confidential**.
2. Select the **Yes/No** option as the type of information in this column.
   1. 
   2. Creating a column in SharePoint that matches the property in FCI.
3. Scroll down and select **No** as the default value for any files that have not been classified.
4. Click **OK** to add the new column to the library.
   1. 
   2. Setting **No** as the default value for the column in SharePoint.

# Adding References

* 1. Run **Visual Studio 2010** and open the **FciSharePointUpload.Core.sln** solution file found under **C:\Server 2008 R2 Labs\FCI-SharePoint Lab\FciSharePointUpload.Core**.
  2. In the **Solution Explorer**, right click on the **FciSharePointUpload.Core** project and select the **Add Reference** option.
  3. 
  4. Adding reference to business logic project.
  5. In the **Add Reference** window, go to the **COM** tab.
  6. Look for the **FSRM 1.0 Type Library** in the list of references. Select it and click **OK**.
  7. 
  8. Locating the library for the File Server Resource Manager.
     1. This adds the reference to the library that will let you access the classification properties of your files.
  9. Double click on the **Uploader.cs** file under the **FciSharePointUpload.Core** project in the **Solution Explorer**.
  10. Around line 232 you should find an instantiation of class called **Copy**. Locate these lines of code:
  11. //TODO: Copy does not exit. Add a class that references the Copy web service from SharePoint.
  12. Copy copyClient = new Copy(urlCopy);
      1. The **Copy** class doesn’t exist. We will follow a series of steps to extract this class from its SharePoint Web service.
  13. Right click on the **FciSharePointUpload.Core** project and select the **Add Service Reference** option.
  14. 
  15. Adding a reference to a Web Service.
  16. Click on the **Advanced** button near the bottom left corner. You should then see the **Service Reference Settings** dialog window and it should be similar to the following.
  17. 
  18. Opening the advanced settings for a web service reference.
      1. You should see the **Add Service Reference** dialog window appear.
  19. Click on the **Add Web Reference** button near the bottom left corner.
  20. 
  21. Click on the **Add Web Reference** button.
  22. At this point the **Add Web Reference** window should be on top.
  23. Open an Internet Explorer window and type the URL to your SharePoint website. In our case it’s http://fci-sp-test:20310, and when we open it the browser displays a page similar to the following:
  24. 
  25. Sample of a SharePoint home page.
  26. Back in Visual Studio, you should still be seeing the **Add Web Reference** window. Type the URL to the home page of your SharePoint site and then append **/\_vti\_bin/copy.asmx** to that link. In our case, for instance, the URL is [**http://fci-sp-test:20310/\_vti\_bin/copy.asmx**](http://fci-sp-test:20310/_vti_bin/copy.asmx).
  27. Click on the green arrow right next to the URL box. After a few seconds on the left you should see the functions in the copy web service, and on the right you should see the option to specify how you will reference the service.
  28. 
  29. Visual Studio shows the methods in the Copy Web Service in Visual Studio.
  30. In the **Web reference name** box, type **SPCopy**. It should end up looking similar to the following:
  31. 
  32. Naming the reference and adding it to the project.
  33. Click on the **Add Reference** button.
      + 1. After a few seconds, a few elements will be added to the **FciSharePointUpload.Core** project. We will need to extract the class and delete some things that we don’t need.
  34. Right click on the **SPCopy** element in the **Web References** folder of your project, and then select the option called **View in Object Browser**.
  35. 
  36. Exploring the classes in the web reference.
  37. Expand the **FciSharePointUpload.Core.SPCopy** project on the **Object Browser**.
  38. 
  39. Locating the Copy class in the web reference through the Object Browser.
  40. Double click on the **Copy** class. Visual Studio should take you to the **Reference.cs** file of the web service.
      1. All methods of the **Copy** class and any additional classes needed by this web service have been automatically added to this **Reference.cs** file.
  41. Right click on the **Reference.cs** tab that indicates that this is the currently open file of code, and then click on the **Copy Full Path** option.
  42. 
  43. Copying the path to the file containing the web reference.
  44. Right click on the **FciSharePointUpload.Core** project in the **Solution Explorer** and select **Add > Existing Item**.
  45. 
  46. Adding the class directly to the project.
      + 1. You should then see the **Add Existing Item** dialog box.
  47. Right click on the **File name** box and click **Paste**.
  48. 
  49. Pasting the path to the References.cs class file.
  50. Click on the **Add** button.
      + 1. This creates a direct copy of the **Reference.cs** file that contains all the classes that we need from the web service. This makes the SPCopy reference no longer necessary.
  51. Right click on the **SPCopy** element in the **Solution Explorer** and select **Delete**.
  52. 
  53. Deleting the SPCopy web reference.
  54. When prompted for confirmation click **OK**.
  55. Repeat the last two steps on these elements to delete them from the project: **Settings.settings** under the **Properties** folder, the **Service References** folder, the **Web References** folder, and the **app.config** file.
  56. Right click on the **Reference.cs** file in the project and select the **Rename** option.
  57. 
  58. Renaming the Reference.cs file in the Solution Explorer.
  59. Type **Copy.cs** as the new name of the file.
  60. Double click on **Copy.cs** to edit its contents.
  61. At around line 16 you should find the namespace reference:
      + 1. namespace FciSharePointUpload.Core.SPCopy {
  62. Delete the **.SPCopy** portion at the end. It should end up looking like:
      + 1. namespace FciSharePointUpload.Core {
  63. At around line 41 you should find the constructor of the **Copy** class. It should be similar to the following:
      + 1. public Copy() {
        2. this.Url = global::FciSharePointUpload.Core.Properties.Settings.Default.FciSharePointUpload\_Core\_SPCopy\_Copy;
        3. if ((this.IsLocalFileSystemWebService(this.Url) == true)) {
        4. this.UseDefaultCredentials = true;
        5. this.useDefaultCredentialsSetExplicitly = false;
        6. }
        7. else {
        8. this.useDefaultCredentialsSetExplicitly = true;
        9. }
        10. }
        11. This default construction is trying to extract the URL of the service from the configuration file of the project. Instead of doing this we will pass the URL directly to the constructor from our User Interface later. That’s why we could delete all the configuration files from this project.
  64. Replace that entire constructor with the following simplified version:
      + 1. public Copy(string url) {
        2. this.Url = url;
        3. }

# Uploading the Files

1. In the **Solution Explorer**, double click on the **Uploader.cs** file under the **FciSharePointUpload.Core** project.
2. Locate the **ReadProperties** method in the **Uploader** class. It should be around line 126.
3. Near the beginning of this method, locate the first To-Do item. The lines should be similar to the following:
   1. //TODO: Load the list of FSRM properties of the file that will be uploaded
   2. FsrmLib.IFsrmCollection properties = null;
4. Change the initialization value of the **properties** collection to pull the properties out of the file that comes as a parameter of the **ReadProperties** method.
   1. FsrmLib.IFsrmCollection properties = fsrmMgr.EnumFileProperties(fileName);
5. Locate the following lines of the code around line 140:
   1. //TODO: Get the definition of the current property
   2. propDef = null;
      1. The code is looping through the properties collection and this variable should contain the definition for the current property.
6. Change the initialization value of the **propDef** variable to reflect the definition of the current property. It should resemble the following code snippet:
   1. propDef = fsrmMgr.GetPropertyDefinition(property.Name);
      1. This definition is used to obtain its SharePoint equivalent. That’s why the names of the FSRM property and the SharePoint property must match.
7. Locate the following code around line 247:
   1. if (propertyAction == PropertyAction.Copy)
   2. {
   3. fields = ReadProperties(file);
   4. }
   5. //TODO: Call Copy Web Service in SharePoint to copy the file over
   6. CopyResult[] results;
   7. uint ret = 0;
8. The **ret** variable should be used to determine the return value of a “copy” call to the SharePoint web service. Change its initialization to the following:
   1. uint ret = copyClient.CopyIntoItems(" ", dest, fields, data, out results);
9. From the **Build** menu, select **Rebuild Solution**.
10. Click on the **Debug** menu and select the **Start Debugging** option, and wait for the **FCI SharePoint Uploader** application to run.
    1. 
    2. Debugging the application with the changes.
11. In the **Url** text box type the URL of the SharePoint home page. In our case it’s **http://fci-sp-test:20310/fci**.
12. In the **Library Path** box type the name of the SharePoint library that you used on the first part of this lab. In our case it’s called **Library**.
13. In the **File** menu click on the **Open Directory** option.
14. Browse for the **C:\Server 2008 R2 Labs\FCI-SharePoint Lab\Data** folder and click **OK**.
    1. 
    2. Browsing for and selecting the sample Data directory.
       1. This adds all files in the Data folder to the list.
15. Click on the **Upload** button. This may take several seconds or even minutes.
    1. 
    2. Starting the upload of files to SharePoint.
16. After the process is complete, review the files that were uploaded to the SharePoint library and take a look at the **Confidential** column. Documents that contain the word “Confidential” should default to Yes.



# Lab Summary

1. The new set of features that have been directly built into the Windows Server 2008 operating system as the File Classification Infrastructure can be used through your custom applications, and therefore it can also be leveraged with any number Microsoft products that support API’s.
2. In this particular scenario we covered how SharePoint and FCI can be leveraged through configurations and code. The property values that can be easily customized by the File Server Resource Manager, and even by your .NET applications, can be reutilized in SharePoint through their Web Services.
3. By following the steps above you can realize that there are many applications for this. Windows Services are usually developed to process entire paths in the hard drive, locate large files, and make massive I/O operations. FCI gives you the tools to flag processed files, summarize results of those processes, all directly in the file. The file can then be transferred through the network to multiple servers and the properties would still remain. In this case we covered how to cross one more boundary and have these properties propagated to SharePoint.
4. It is advised that you review the different Web Services that SharePoint has to offer and find the multiple applications that could be implemented or enhanced by using the File Classification Infrastructure to complement them.