GUIDE

Workshare Compare Server

Developers Guide

Workshare

Company Information

Workshare Compare Server Developers Guide

Workshare Ltd. (UK)
20 Fashion Street
London
E1 6PX
UK

Workshare Inc. (USA)
208 Utah Street, Suite 350
San Francisco
CA 94103
USA

Workshare Website: www.workshare.com

Trademarks

Trademarked names appear throughout this guide as well as on other parts of the product. Instead of listing these here or inserting numerous trademark symbols, Workshare wishes to state categorically that no infringement of intellectual or other copyright is intended and that trademarks are used only for editorial purposes.

Disclaimers

The authors/publishers of the Workshare Compare Server Developers Guide and associated Help material have used their best efforts to ensure accuracy and effectiveness. Due to the continuing nature of software development, it may be necessary to distribute updated Help from time to time. The authors would like to assure users of their continued best efforts in supplying the most effective Help material possible.

The authors/publishers, however, make no warranty of any kind, expressed or implied, with regard to Workshare programs or Help material associated with them, including the Workshare Compare Server API Guide. The authors/publishers shall not be liable in the event of incidental or consequential damages in connection with, or arising out of, the programs or associated Help instructions.

Copyright

Table of Contents

Table of Figures

Chapter 1. Introduction

This chapter describes the functionality provided by Workshare Compare Server and the prerequisites for developing client applications which exploit this functionality. It includes the following sections:

- **Introducing Workshare Compare Server**, below, introduces Workshare Compare Server.
- **Prerequisites**, page 6, describes the specific technologies mentioned in this guide and the system requirements for creating client applications which are consumers of the Workshare Compare service.
- **Service Endpoints**, page 6, describes the endpoints exposed by Workshare Compare service.

***Note**: In this document, the terms Workshare Compare Server and Workshare Compare service are interchangeable. Workshare Compare Server is the name of the product but where you find references to Workshare Compare service, it is in order to be technically accurate.*

Introducing Workshare Compare Server

Workshare Compare Server is a web service that provides extremely fast and robust document comparison and returns a range of outputs including a comprehensive comparison document (RTF, DOC, DOCX or PDF) traditionally known as a 'Redline', an XML change summary and a Workshare Professional compatible WDF document. Using the Workshare Compare service, you can write applications that will:

- Provide extremely fast and robust document comparison, including change identification and extraction.
- Verify and highlight all changes and differences between drafts and versions, no matter how complex the document.
- Validate that all changes closely adhere to policies and procedures, and an approved boilerplate.

The sample applications demonstrate the use of the Workshare Compare web service to produce and display comparison outputs, given two input documents.

Communicating with Workshare Compare Server

Workshare Compare Server can be accessed via standard web service protocols and via the Microsoft WCF technology. However, the Workshare Compare Server installation includes a .NET assembly (Control.dll) which can be referenced in order to access the functionality of Workshare Compare Server without any knowledge of web services or WCF. For new integrations, this is the recommended method of communicating with Workshare Compare Server. For an example of how to use Control.dll, refer to *Chapter 5: Compare Control DLL Methods* and also the advanced web sample described on page 13.

Queued vs. Direct (Synchronous/Asynchronous) Services

There are two models for developing against the Workshare Compare service; a Queued and Direct version. Both allow two documents to be used as inputs and are capable of producing a range of outputs. They both perform all comparison processing on the server and require no client-side Workshare components.

The queued call uses document identifiers to uniquely reference documents stored on the server file system or within a Document Management System (DMS) which can be accessed by the server. The document IDs specify which files to use as inputs as well as the location (on the server or within a DMS) where the resulting output should be written. Once the queued call is posted, all further processing takes place on the server.

The direct call loads two documents at the client and sends them to the server for comparison. The resulting output is then sent back to the client once processing is complete. Direct calls may be made synchronously (the call blocks the execution on the client until the comparison document is returned) or asynchronously (the call returns immediately and the client may either poll for the result or use a callback event to receive a notification once processing is complete).

Prerequisites

To fully benefit from this guide, you should have an understanding of the Microsoft .NET Framework and have basic knowledge of how to use Service References and consume Windows Communication Foundation (WCF) based web services. The specific technologies mentioned in the guide are C#, the Microsoft Visual Studio .NET development system, the .NET Framework, WCF and XML. However, any technologies that consume standard web services can be utilized.

System Requirements

To create client applications which are consumers of the Workshare Compare service, you will need to ensure that your system meets specific development software requirements. For example:

- C# Sample Application – Synchronous
  - Microsoft Windows XP
- C# Sample Application – MSMQ
  - Microsoft Windows XP or 2008 Server
  - Microsoft Message Queuing
- Java Sample Application
  - Java Runtime Environment (JRE) 1.5

Service Endpoints

Workshare Compare Server is a web service which exposes a document comparison API enabling developers to write custom software that compares two documents and produces a Redline document that describes the differences between the two documents.

Workshare Compare Server provides two service endpoints which allow for backward compatibility with legacy clients (built against previous versions of Workshare Compare Server) as well as utilising the most up to date transport and security technologies. Importing a Service Reference creates a client proxy containing classes for both endpoints:

- **CompareWebServiceSoap** – endpoint exposing a basic Http binding configured to be fully compatible with legacy web service (.asmx) consumers using standard, plain text encoded, SOAP messages. This endpoint can accept DOC and RTF formats as input and outputs a Redline document in RTF format and a Change Summary in XML format.

  To connect to the service using the CompareWebServiceSoap endpoint you should create an instance of the LegacyComparerClient proxy class.

  When connecting to the CompareWebServiceSoap endpoint using the non-default constructor the service URL should be the default service address (e.g. http://compareserver/wcs/compareweb service.svc)

- **CompareWebServiceWCF** – endpoint exposing a wsHttp binding which utilises WCF channels for transport and security and uses Message Transmission Optimization Mechanism (MTOM) encoding to provide large document handling. This endpoint can accept DOC, RTF, PDF and HTML formats as input and outputs a Redline document in RTF, DOC, DOCX or PDF format, a Change Summary in XML format and a Deltafile (containing the source documents and Redline document) in WDF format. A WDF file can be opened in the Workshare Professional Compare module.

  To connect to the service using the CompareWebServiceWCF endpoint you should create an instance of the ComparerClient proxy class.

  When connecting to the CompareWebServiceWCF endpoint using the non-default constructor the service URI should be appended with 'Compare5' (e.g. http://compareserver/wcs/compareweb service.svc/compare5)

  Note: *You are only required to specify a service URI when using the non-default client constructor.*

Note: *When adding a Service Reference the default service URI (without 'Compare5') should always be provided, regardless of the endpoint you intend to employ. Proxy classes are automatically imported for both SOAP and WCF endpoints.*

Chapter 2. Sample Applications

This chapter describes the sample applications provided with Workshare Compare Server. It includes the following sections:

- **Introducing the Sample Applications**, below, introduces the sample applications included with Workshare Compare Server.
- **Running the Sample Applications**, page 10, describes how to run each of the three sample applications.

Introducing the Sample Applications

Workshare Compare Server installs with 6 sample applications – two C#, one Java and three ASP. These applications demonstrate different programming languages and models for connecting to and interacting with the Workshare Compare service and full source code solutions are provided.

The `{INSTALL_DIR}\Workshare\Compare Service\Samples` directory contains the C# and Java sample applications and documents:

The `Document.Services.Compare.MSMQSample` C#' application demonstrates the queued method of utilizing the Workshare Compare service via Microsoft Message Queuing.

The `Document.Services.Compare.Sample` C# application connects to the Workshare Compare service using a direct synchronous request/response method.

The `comparews` Java application has been precompiled using Java SDK1.5, and demonstrates utilizing the Workshare Compare service via the legacy Soap endpoint and Direct, synchronous, connection method.

Also contained in the `Samples` folder is the `LocalFilestoreCompareQEventHandler` folder which contains sample code demonstrating how to integrate Workshare Compare service with a server-side DMS. Refer to *Integrating Queued Comparison with a DMS*, page 20.

Test documents, rendering sets and full source code solutions for all of the sample applications are provided within the `\samples` directory.

The `{INSTALL_DIR}\Workshare\Compare Service` directory contains web sites (virtual folders) for the three ASP samples, as well as their full source code.

The `AdvancedWebSample ASP` application demonstrates using the Workshare Compare service Control DLL to connect to the server and perform synchronous comparisons.

> ***Note**: Using the Control DLL is the recommended method of connecting to the Workshare Compare service. See* Chapter 5 *for more details on the methods/properties available via the Control DLL.*

The `BasicWebSample ASP` application demonstrates how to use WCF in order to connect directly to the Workshare Compare service.

The `ConfigPageSample ASP` application contains the full source code for the Administration Dashboard and demonstrates the more advanced features of the Workshare Compare service.

Running the Sample Applications

Note: *The included C# samples utilize an associated CONFIG file to provide default values to the applications.*

Running the Document.Services.Compare.MSMQSample Application

This example demonstrates using the IComparerQueued interface to make an asynchronous request to the Workshare Compare service which accepts RTF, DOC, PDF and HTML input files. The resulting response files are saved to a specified output directory for further processing.

To run the Document.Services.Compare.MSMQSample application:

1. From the Start menu, select **Programs > Workshare > Samples > MSMQ Sample**:

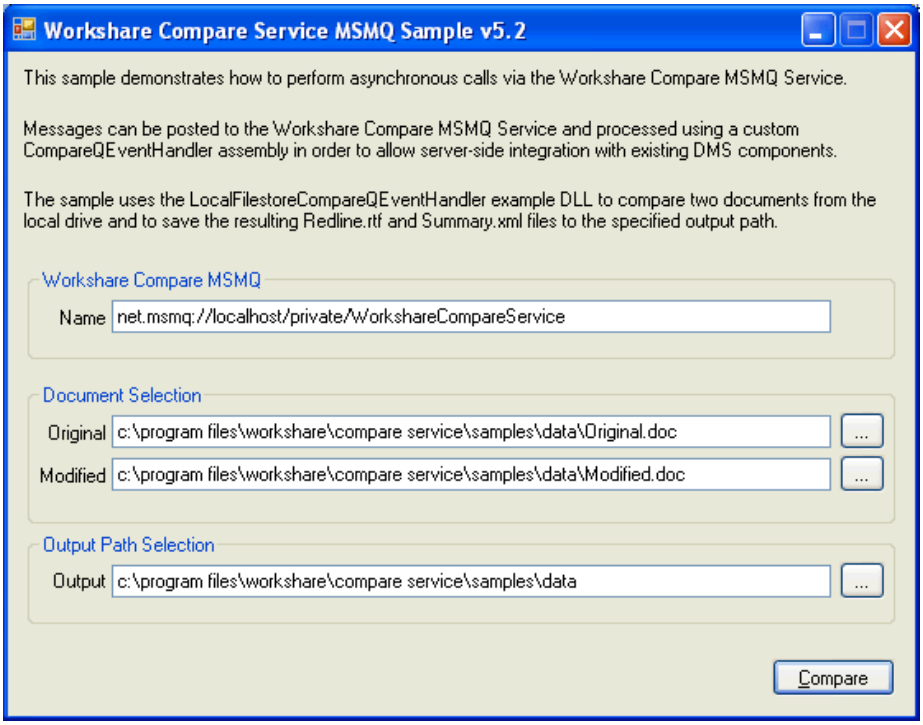

Figure 1. Document.Services.Compare.MSMQSample

2. Select the MSMQ location.
3. Select the original and modified documents to be compared.
4. Select the output path for the comparison results.
5. Select **Compare** to post the request to the specified Workshare Compare MSMQ. Upon successful execution a Redline document and summary file will be created in the specified output folder.

Running the Document.Services.Compare.Sample Application

This example demonstrates how to write a C# WinForm client in order to compare documents using the Workshare Compare service. The sample utilizes the CompareWebServiceWCF interface (as denoted by the /Compare5 host URL) and demonstrates the full feature set of the service.

To run the Document.Services.Compare.Sample application:

1. From the Start menu, select Programs > Workshare > Samples > C# Sample:

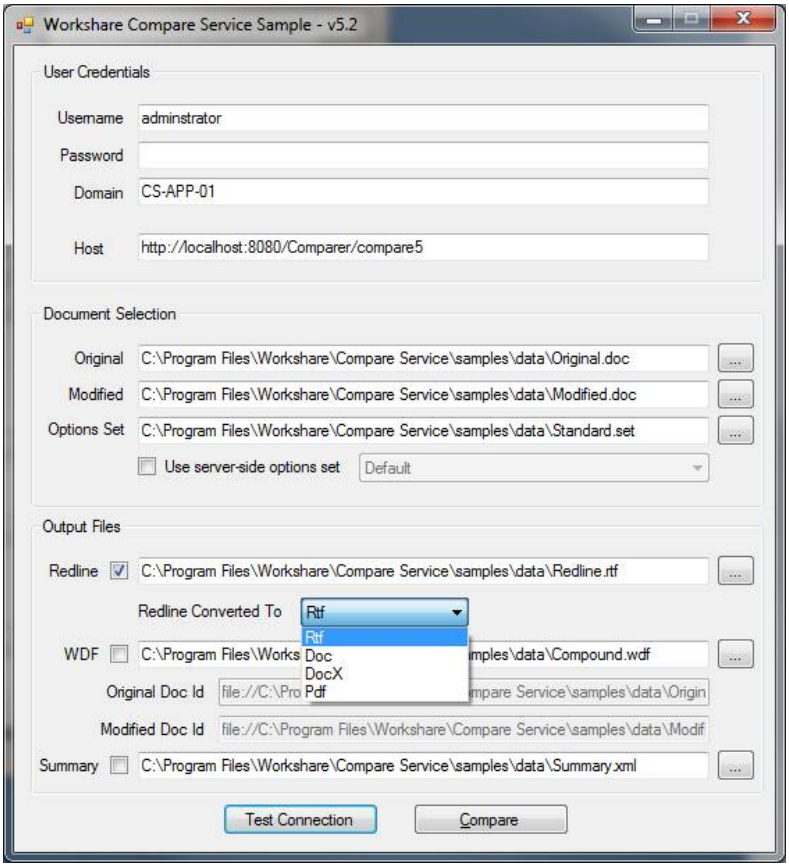

Figure 2. Document.Services.Compare.Sample

2. In the **Host** field, enter **http://localhost/wcs/comparewebservice.svc/Compare5** where Compare Service is hosted in IIS or **http://local host:8080/Comparer/Compare5** where Compare Service is hosted as a Windows service.

3. Select the original and modified documents to be compared.

> ***Note:*** *Input file formats include RTF, DOC, PDF and HTML.*

4. You can specify a custom client-side Options Set (i.e. a rendering set filename) along with the request, or use a default server-side configuration by selecting from the dropdown. Refer to the *Workshare Compare service Rendering Set Guide* for details on rendering set values.

5. Select the type of output you require. You can select one of the following:
   - **Redline**: The comparison can be in RTF, DOC, DOCX or PDF format.
   - **WDF**: The comparison is in the Workshare DeltaFile format which can be opened in the Workshare Professional Compare module.
6. If required, you can also select **Summary** to include an XML summary in the output.
7. Enter user name, password and domain details for a Windows account which is valid on the server.
8. Select **Compare**. The requested output documents are generated on the server and returned to the client.

Running the COMPAREWS Application

This example demonstrates how a Java client can be written to compare documents using the legacy SOAP interface of the Workshare Compare service.

To run the COMPAREWS application:

1. From the Start menu, select Programs > Workshare > Samples > Java Sample

> ***Note:*** *The sample application is pre-compiled using Java SDK 1.5, and requires JRE version 1.5 or greater in order to run.*

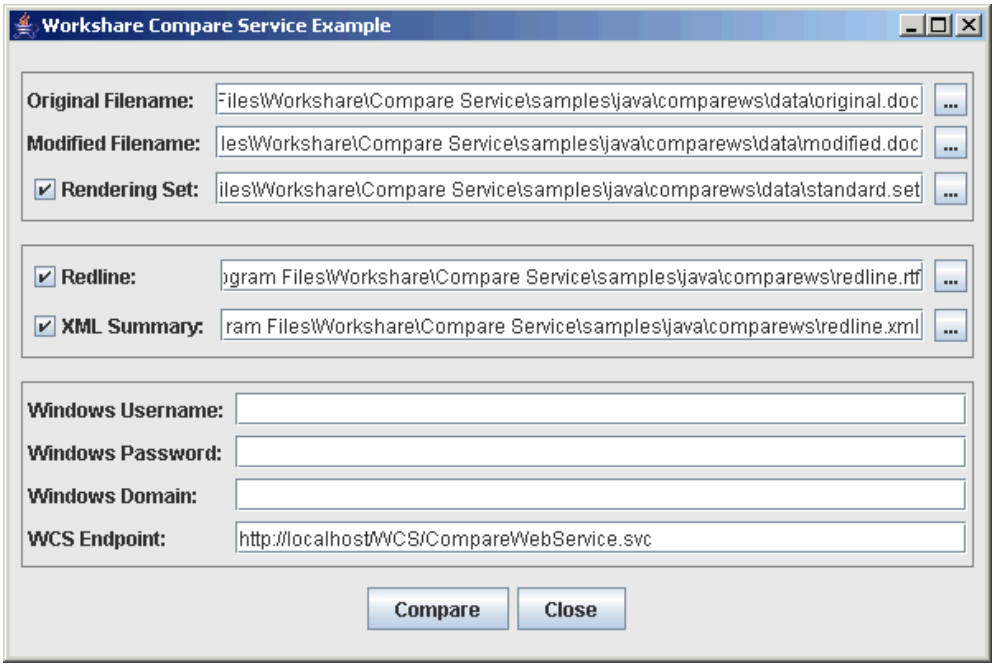

Figure 3. COMPAREWS

2. Select the original and modified documents to be compared.

> ***Note:*** *Only RTF and DOC source documents are accepted in the legacy SOAP interface.*

3. You can specify a custom rendering set to be loaded and sent along with the request. If **Rendering** Set is unchecked, the default server configuration rendering set will be used. Refer to the *Workshare Compare service Rendering Set Guide* for details on rendering set values.
4. Select the type of output you require - Redline document (in RTF file format) and an XML summary.
5. Enter user name, password and domain details for a Windows account which is valid on the server. Although these are not used by the basic HTTP binding they are required to allow for future enhancements to the service authentication process.
6. Select **Compare**. The requested output documents are generated on the server and returned to the client.

Provided are additional BATCH scripts to aid in Java development:

- compile JDK1_5.cmd – recompiles the src source code tree using JDK 1.5 (if installed)
- compile JDK1_6.cmd – recompiles the src source code tree using JDK 1.6 (if installed)
- wsdl2java.cmd – regenerates the Java source code tree using the currently published service WSDL on the local server.

Running the Advanced Web Sample

This example demonstrates how to integrate an ASP web application with the Workshare Compare service using the .NET Control DLL.

To run the advanced web sample:

1. Open a web browser and enter **http://hostname:8086** in the address bar, where hostname is the name of the machine where Workshare Compare service is installed. The login screen is displayed.

> ***Note:*** *8086 is the default port which can be changed during installation.*

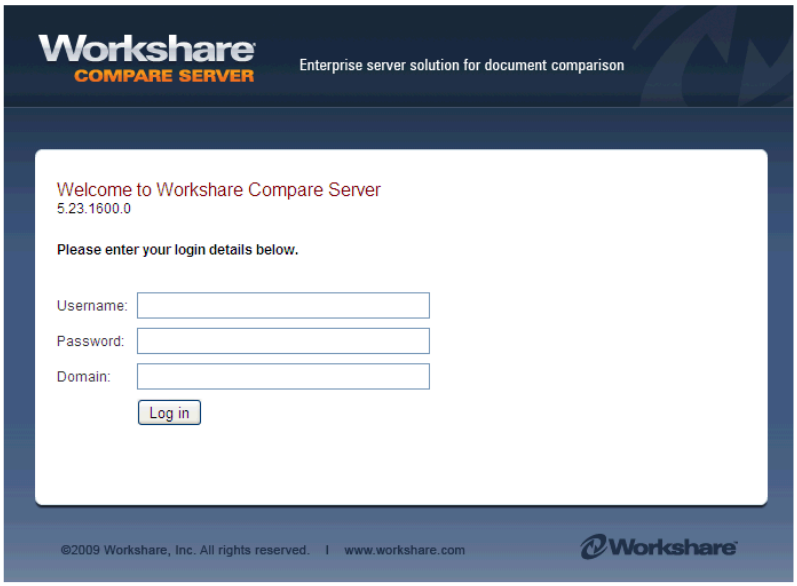

Figure 4. Advanced Web Sample - Login

2. Enter your login credentials and click **Log in**. The document selection screen is displayed.

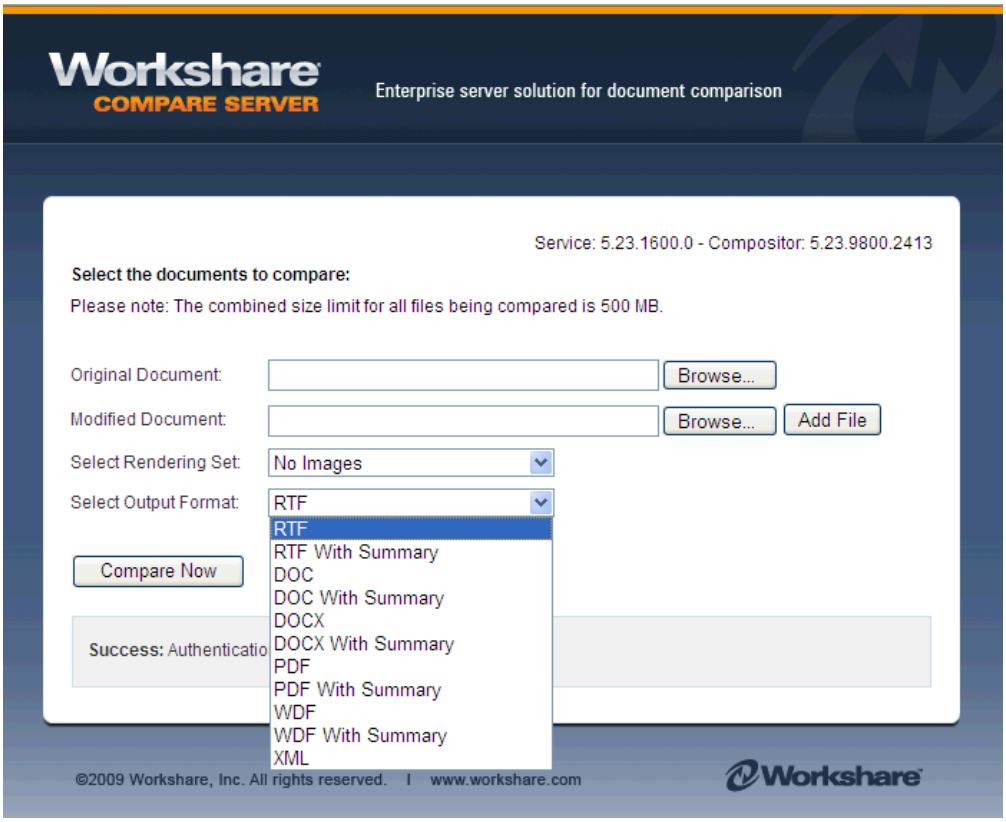

Figure 5. Advanced Web Sample

3. Select the original and modified documents to be compared.
4. If you want to compare the original document against multiple documents, click **Add File**. An additional **Modified Document** field is added.
5. From the **Select Rendering Set** dropdown list, select the rendering set you want applied to the comparison.
6. Select the type of output you require – a Redline document (in RTF, DOC, DOCX, PDF or WDF file format) with or without an XML summary or just an XML summary.

7. Click **Compare Now**. The comparison is performed and the results screen is displayed.

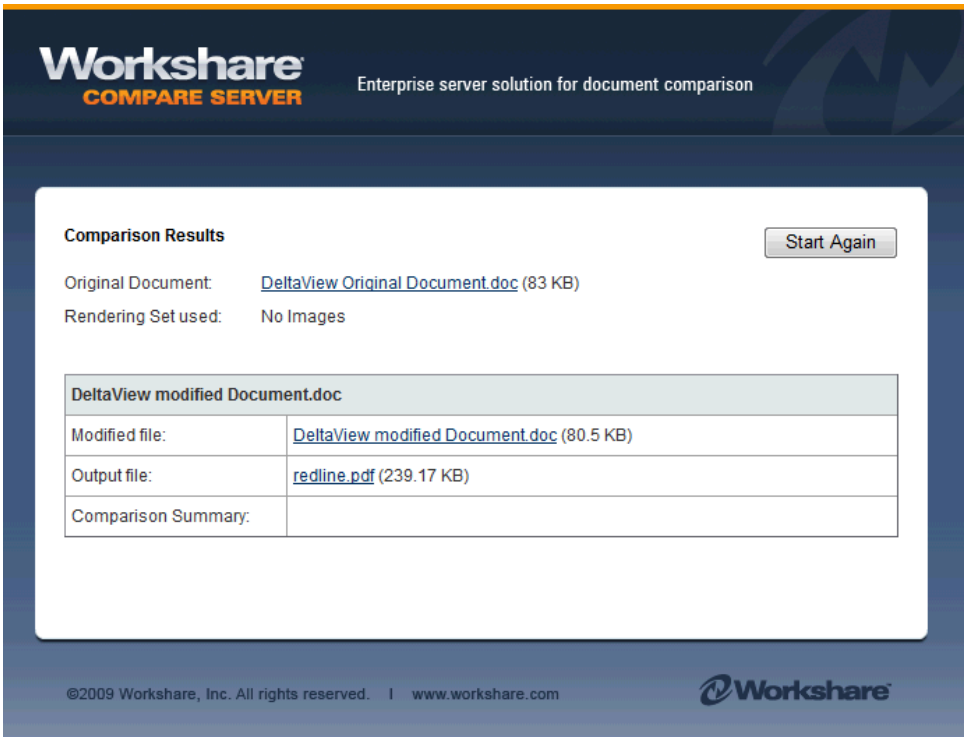

Figure 6. Advanced Web Sample - Results

You can view the comparison document by clicking on the output file link. You can also view the original and modified documents.

Running the Basic Web Sample

This example demonstrates how to integrate an ASP web application with the Workshare Compare service and make a direct connection and method calls using WCF.

To run the basic web sample:

1. Open a web browser and enter **http://hostname:8085** in the address bar, where hostname is the name of the machine where Workshare Compare service is installed. The login screen is displayed.

> ***Note:*** *8085 is the default port which can be changed during installation.*

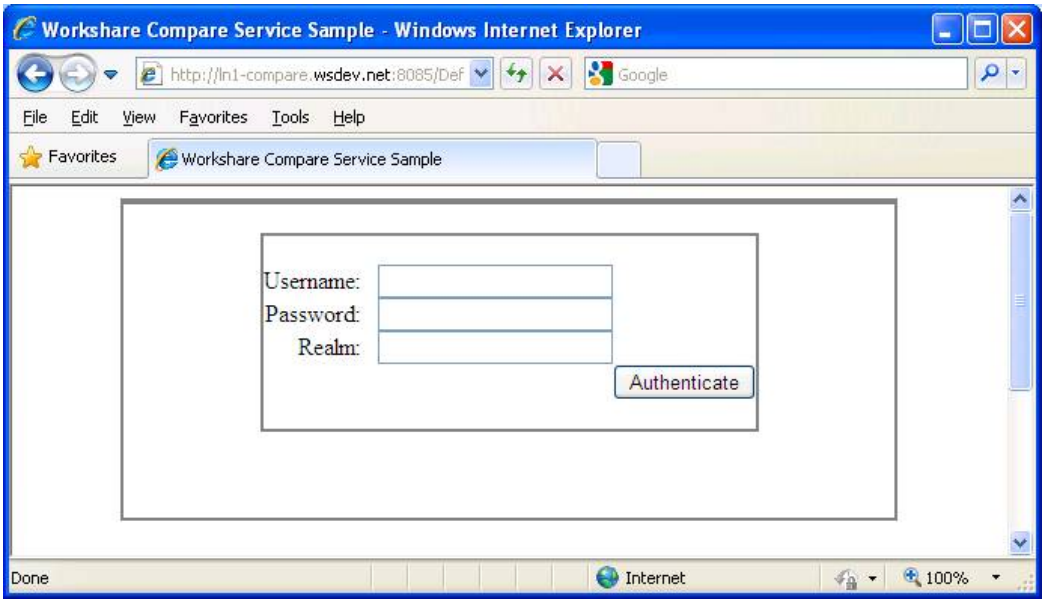

Figure 7. Basic Web Sample - Login

2. Enter your login credentials and click **Authenticate**. The document selection screen is displayed.

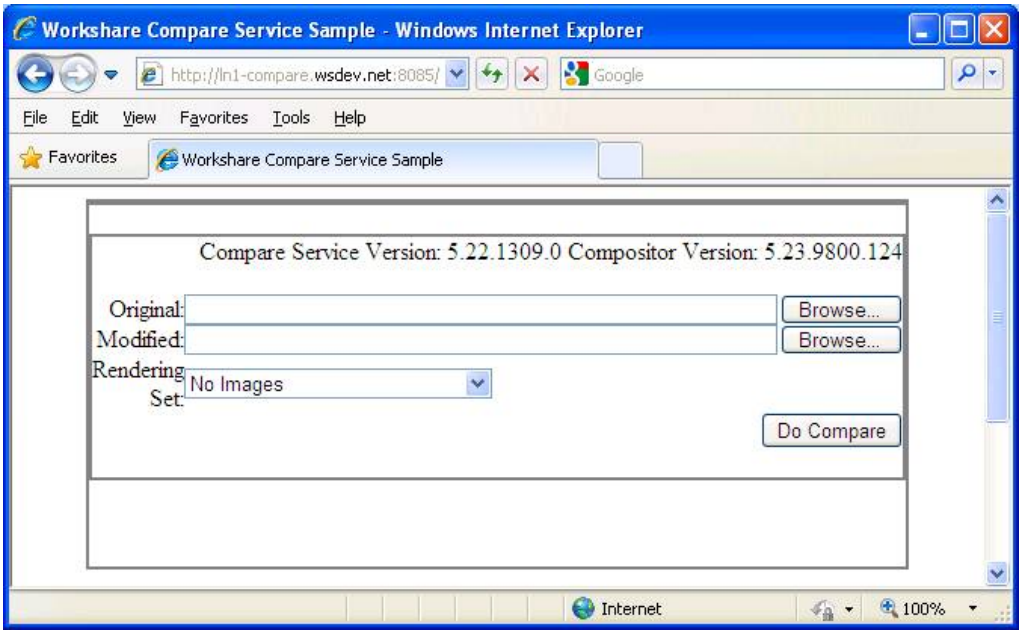

Figure 8. Basic Web Sample

3. Select the original and modified documents to be compared.
4. From the **Rendering Set** dropdown list, select the rendering set you want applied to the comparison.
5. Click **Do Compare**. The comparison is performed and the results screen is displayed.

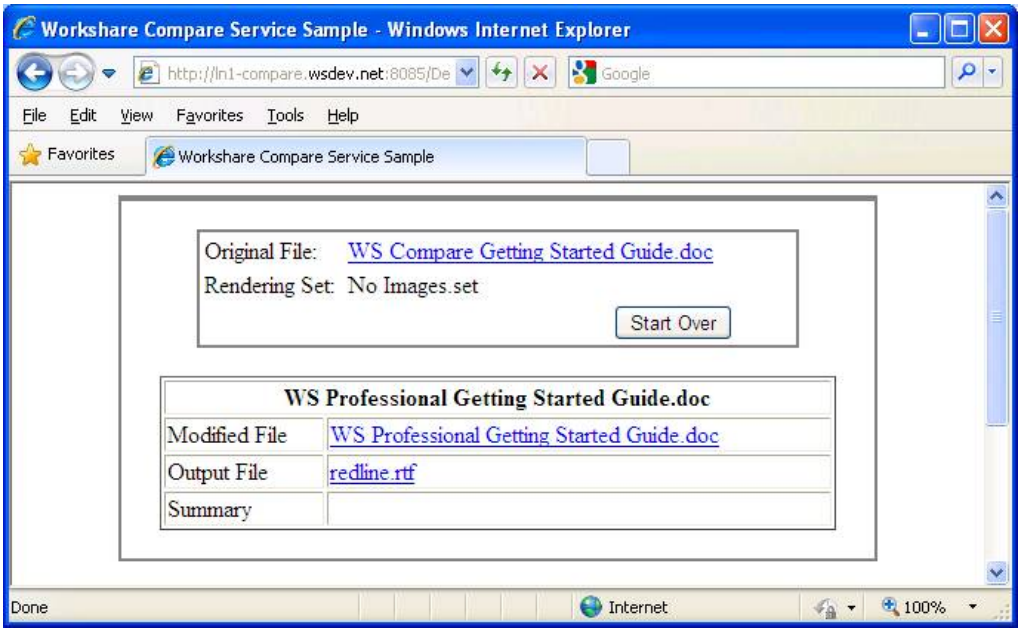

Figure 9. Basic Web Sample - Results

You can view the comparison document by clicking on the output file link. You can also view the original and modified documents.

Chapter 3. Sample Code

This chapter provides sample code. It includes the following sections:

- **Creating a Console Application**, below, describes how to create a console application using C#.
- **Integrating Queued Comparison with a DMS**, page 20, describes how to create a custom C# module which provides server-side integration between the Workshare Compare service and an existing DMS or similar server-based storage architecture.
- **Service Creation and Authentication**, page 24, provides an example of how client code can connect to Workshare Compare service.
- **Comparison**, page 24, provides an example of how client code can perform a comparison.

Creating a Console Application

The purpose of this section is to demonstrate the potential of Workshare Compare Server by describing a potential use of this API - creating a console application. The following procedure describes how to create a console application using C# which provides direct document comparison using the Workshare Compare service.

To create a console application:

1. Create a new project in Visual Studio.NET by selecting *File* > **New** > **Project**. Enter **MyCompare** as the project name and select the location of the project.

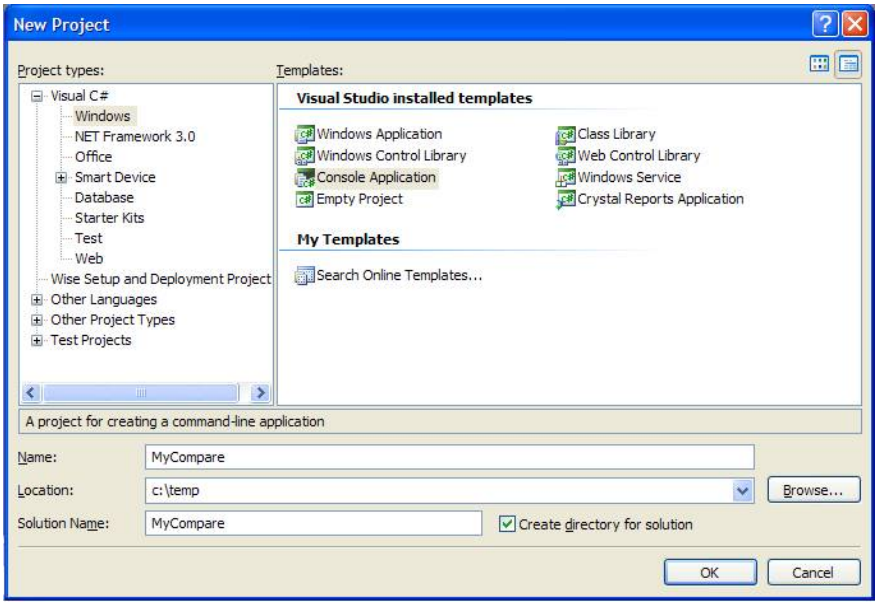

Figure 10. New Project Dialog Box

2. Click **OK** to continue.

Note: *Direct calls can be made using either Synchronous or Asynchronous methods. The examples in this document demonstrate how to create and use a proxy capable of synchronous calls. To allow asynchronous calls you should create and use a similar proxy generated by using the ServiceModel Metadata Utility Tool (svcutil.exe) with the /async option.*

3. Add a service reference by right-clicking the **MyCompare** project in the solution viewer, and selecting **Add Service Reference**.

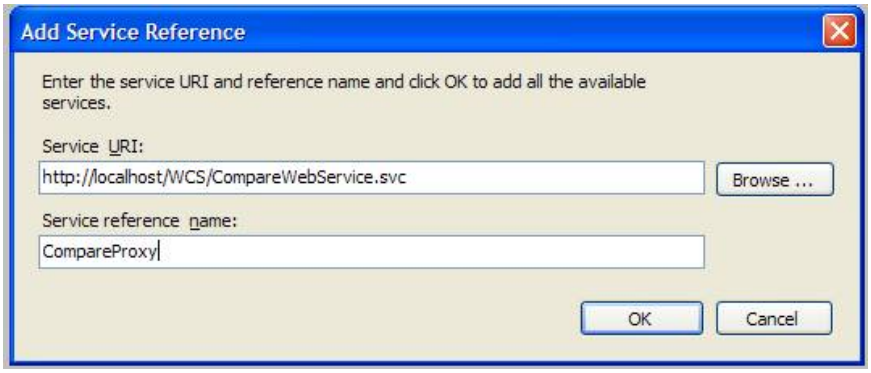

Figure 11. Add Service Reference Dialog Box

4. Ensure that the **Service URI** is pointing to your installed version of the Workshare Compare service and click **OK**.

5. Add the following code to your console application:

```
static void Main(string[] args)
{
  CompareProxy.ComparerClient cp = new CompareProxy.ComparerClient();

  if( cp.Authenticate("TestRealm", "TestUser", "TestPass") )
  {
   byte[] original = File.ReadAllBytes(@"c:\original.doc");
   byte[] modified = File.ReadAllBytes(@"c:\modified.doc");
   string sOptionsSet = File.ReadAllText(@"c:\standard.set");

   CompareProxy.CompareResults rs = cp.Execute(original, modified,
                                        CompareProxy.ResponseOptions.Rtf,
                                        sOptionsSet);

   File.WriteAllBytes(@"c:\redline.rtf", rs.Redline);
  }
}
```

6. Run the console application and if everything is configured correctly, a Redline.rtf file is generated.

Integrating Queued Comparison with a DMS

Workshare Compare Server exposes a netMsmqBinding endpoint, hosted via the 'Workshare Compare MSMQ Service', which allows clients to make asynchronous calls to compare documents within a server-side DMS or other custom storage architecture.

The Workshare Compare MSMQ Service runs as a standard Windows Service on the server and processes requests as they appear on the WorkshareCompareService message queue. If the service is stopped then requests are persisted to the server-side queue and processed when the service is restarted.

Messages relating to queued requests do not contain any document data or resulting Redline information and merely provide document references which allow a CompareQEventHandler to extract and process the relevant information on the server. Using the queued service, documents can be extracted, compared and the resulting Redline stored without the need to transmit sensitive information to the client.

In order to expose the netMsmqBinding endpoint a Service Reference can be added to the client application using port 8080 on the server. For example, **`http://CompareServer:8080/`**.

The WSDL file for the queued service can also be accessed via the metadata exchange endpoint on port 8008. For example, **`http://CompareServer:8008/mex`**.

Adding a Service Reference to the client application creates a local service proxy and allows the client to call methods on the queued Workshare Compare service using the same syntax as for standard synchronous requests. (See the MSMQSample code for a complete example of using a local proxy to make asynchronous queued calls.)

> ***Note:*** *You can check that the service is installed and running correctly using Control Panel > Administrative Tools > Services.*

Configuring and Calling the Workshare Compare MSMQ Service

The netMsmqBinding endpoint exposes the IComparerQueued interface (this generates a ComparerQueuedClient local proxy when imported as a service reference) which provides two methods for making queued requests from the client:

```
void ExecuteQueued(Dictionary<string, string> qReq);
void ExecuteQueuedCustom(Dictionary<string, string> qReq, string sHandler);
```

Both methods use the 'qReq' Dictionary parameter to compose a message containing key/value string pairs and to post this message to the server-side WorkshareCompareService message queue.

For example:

```
ComparerQueuedClient qComparer = new ComparerQueuedClient();
Dictionary<string, string> qParams = new Dictionary<string, string>();
qParams.Add("OriginalDocument", "c:\\original.doc");
qParams.Add("ModifiedDocument", "c:\\modified.doc");
qParams.Add("RenderingSet", "");
qParams.Add("RedlineDocument", "c:\\redline.rtf");
qParams.Add("RedlineXML", "c:\\summary.xml");
qComparer.ExecuteQueued(qParams);
```

The Workshare Compare MSMQ Service reads from the server-side queue and plays back the message to the Workshare Compare service. The contents of the key/value pairs within the message are then passed by the Workshare Compare service to a CompareQEventHandler assembly which determines how the parameters are interpreted on the server and allows custom handling to be implemented. You can implement multiple CompareQEventHandlers in order to provide a range of server-side integration, or use key/value pairs to specify processing options for a single, default, handler.

The 'Workshare.Document.Services.Compare.MSMQHost.exe.config' file (located in the IIS the Virtual Directory folder, specified during installation) is used to specify a default CompareQEventHandler assembly as well as any optional extra handlers you may wish to implement.

For example, Default Handler:

```
<add key="DefaultCompareQEventHandlerAssembly"
value="LocalFilestoreCompareQEventHandler,

Version=5.0.1.0, Culture=neutral, PublicKeyToken=5fa10b593c538362" />

<add key="DefaultCompareQEventHandlerClass"

value="LocalFilestoreCompareQEventHandler.LocalFilestoreCompareQEvents" />
```

The ExecuteQueued method processes messages through the assembly specified in the DefaultCompareQEventHandlerAssembly key, while the sHandler string parameter of the ExecuteQueuedCustom method can be used to call the alternative assemblies specified in other configuration keys.

For example, Custom Handler:

```
<add key="DMS1CompareQEventHandlerAssembly" value="MyCompany.MyEventHandler,

PublicKeyToken=1234566789abcdef" />

<add key="DMS1CompareQEventHandlerClass" value=" MyCompany.MyEventHandler.
MyEventHandlerEvents" />
```

The sHandler string specifies the prefix used in the CompareQEventHandler keys.

For example:

```
qComparer.ExecuteQueuedCustom(qParams, "DMS1");
```

Writing a Custom compareqeventhandler

In order to create a custom CompareQEventHandler assembly, build a new .NET class library and include a class that implements the BeginExecute and EndExecute methods of the Workshare.Document. Services.Compare.QInterface.ICompareQEvents interface.

To create a .NET class library:

1. Create a new Class Library project in Visual Studio.NET by selecting **File** > **New** > **Project**. Enter **MyCompareQEventHandler** as the project name and select the location of the project.

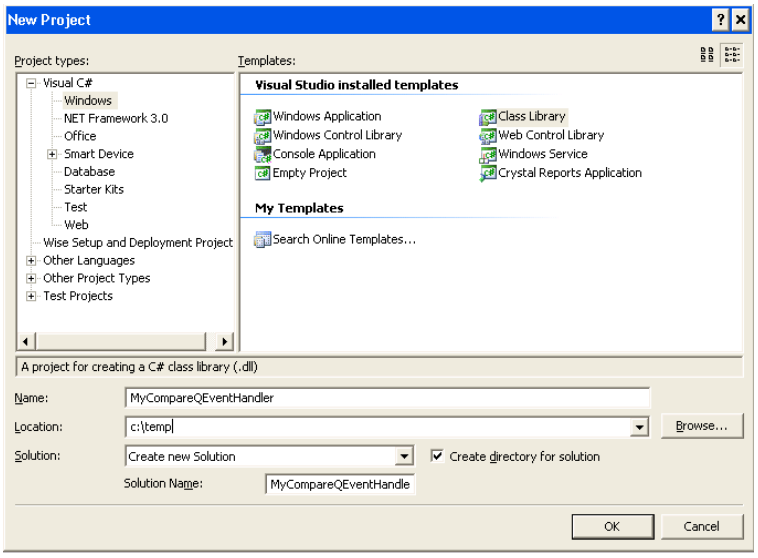

Figure 12. New Project Dialog Box

2. Click **OK** to continue.

3. Add a reference to the ICompareQEvents interface by right–clicking the **MyCompareQEventHandler** project in the solution viewer, and selecting **Add Reference**.

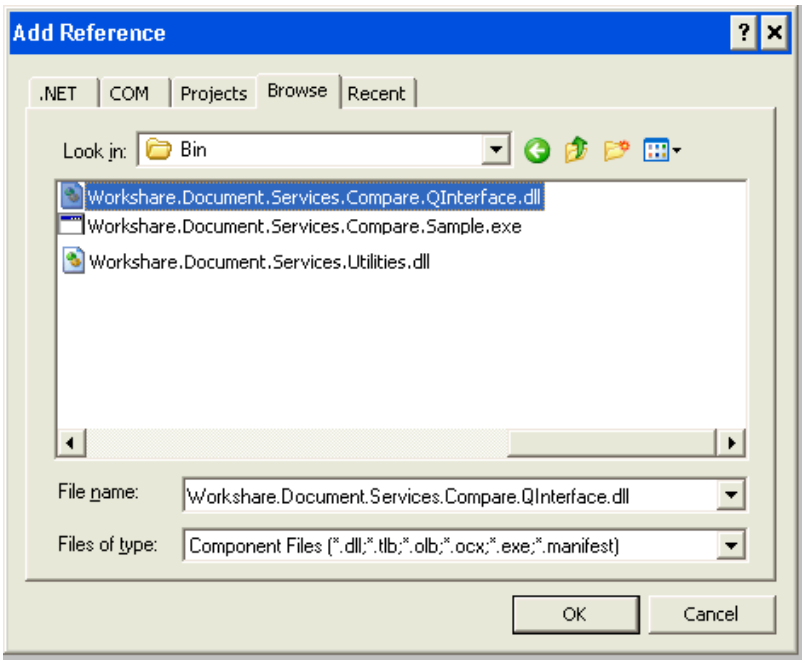

Figure 13. Add Reference Dialog Box

4. Browse to the IIS Virtual Directory folder (specified during installation) and select the Workshare.Document.Services.Compare.QInterface DLL from the **Bin** subfolder.
5. Click **OK** to continue.
6. Edit the main class of your project (defaults to class1.cs) so that the code includes the following:

```
using System;
using System.Collections.Generic;
using System.Text;
using Workshare.Document.Services.Compare.QInterface;


namespace MyCompareQEventHandler
{
    public class Class1 : ICompareQEvents
    {
        public void BeginExecute(Dictionary<string, string> Params, out
byte[] originalDocumentBuffer, out byte[] modifiedDocumentBuffer, out
string renderingset)
        {
originalDocumentBuffer = File.ReadAllBytes(Params ["OriginalDocument"]);
            modifiedDocumentBuffer = File.ReadAllBytes(Params
["ModifiedDocument"]);
renderingset = File.ReadAllText(Params ["RenderingSet"]);
        }
        public void EndExecute(Dictionary<string, string> Params, string
changeSummaryXml, byte[] redlineDocumentBuffer)
        {
File.WriteAllText(dicQueueParams["RedlineXML"], changeSummaryXml);
             File.WriteAllBytes(dicQueueParams["RedlineDocument"],
redlineDocumentBuffer);
        }
    }
}
(This example demonstrates how to use the key/value parameters to
compare two documents from the local drive and save the resulting reline
and summary. For a full example see the
LocalFilestoreCompareQEventHandler sample project.)
```

7. Build the class library and copy the DLL output next to the QInterface assembly.
8. Update the Workshare.Document.Services.Compare.MSMQHost.exe.config file to include the assembly name and class name from you project (see the example from the previous section).

When the Workshare Compare MSMQ Service is running, each message played back through the Workshare Compare service will call the BeginExecute method to get the original and modified document buffers and a string containing the rendering options. Once the comparison is complete, EndExecute is called with the resulting summary and redline buffers.

Service Creation and Authentication

A non-default constructor is provided to allow a client to dynamically provide configuration and authentication details. The client must authenticate with the web service before calling any of the other methods. The username, password and domain provided should represent an existing Windows account which the server is capable of validating using Security Support Provider Interface (SSPI – Windows logon etc.).

```
// textHost is a TextBox control where the user can enter the service URI
// e.g. http://compareserver/wcs/comparewebservice.svc/compare5
ComparerClient svcCompare = new ComparerClient("CompareWebServiceWCF",
textHost.Text);


// textUsername, textPassword, textDomain are TextBox controls where the user
can
// can enter their Windows credentials
svcCompare.ClientCredentials.Windows.ClientCredential.UserName =
textUsername.Text;
svcCompare.ClientCredentials.Windows.ClientCredential.Password =
textPassword.Text;
svcCompare.ClientCredentials.Windows.ClientCredential.Domain = textDomain.Text;


if( svcCompare.Authenticate("DIS", "User", "Pass") )
{
…
}
```

Comparison

The two documents to be compared are both loaded into byte arrays by the client. The first parameter is treated as the original document and the second parameter is treated as the modified document. The comparison is done immediately on the server, and the call does not return until the comparison is complete. (If an error occurs an exception is thrown and event logs generated).

```
byte[] original = File.ReadAllBytes(@"c:\original.doc");
byte[] modified = File.ReadAllBytes(@"c:\modified.doc");
string sOptionsSet = File.ReadAllText(@"c:\standard.set");

CompareProxy.CompareResults rs = cp.Execute(original, modified,
                                    CompareProxy.ResponseOptions.Rtf,
                                    sOptionsSet);

File.WriteAllBytes(@"c:\redline.rtf", rs.Redline);
```

Chapter 4. Compare Web Service Methods

This chapter describes the Workshare Compare service methods exposed by an imported Service Reference in Microsoft Visual Studio. It includes the following sections:

- **Public Constructors**, below, describes the public constructors of Workshare Compare service.
- **Public Methods**, page 26, describes the public methods of Workshare Compare service.

Public Constructors

The following constructor can be used to create a **Compare** service instance.

Name: `ComparerClient` (Default)

Description: Initializes a new instance of the Compare service using configuration from the client's app.config file.

```
public ComparerClient()
```

default client constructor.

Name: `ComparerClient`

Description: Initializes a new instance of the Compare service using a specified endpoint configuration (from the client's app.config file) and URI.

```
public ComparerClient(string endpointConfigurationName, string remoteAddress)
```

Parameters:

`endpointConfigurationName`

Name of a binding specified in the app.config file under <system.serviceModel><bindings>

`remoteAddress`

Full URI of server endpoint (including 'Compare5' for WCF endpoints)

Public Methods

The following table shows the methods of the **Compare** service and a brief description of each.

Name: `Authenticate`

Description: Authorizes the user. All clients must call this method before calling any of the Execute methods.

```
public bool Authenticate(string sRealm, string sUser, string sPassword)
```

Parameters:

`sRealm`
Domain name for a Windows account which can be validated by the server.

`sUser`
Username for a Windows account which can be validated by the server.

`sPassword`
Password for the specified Windows account which can be validated by the server.

Return Value:

`true if the user account was successfully verified.`

Name: `Execute`

Description: Perform a direct comparison of two documents.

```
public CompareResults Execute(byte[] OriginalData, byte[] ModifiedData,
ResponseOptions ResponseOption, string CompareOptions)
```

Parameters:

`OriginalData`
Original document loaded into memory as a byte array.

`ModifiedData`
Modified document loaded into memory as a byte array.

`ResponseOption`
Specifies what output is generated by the comparison. Rtf, DOC, DOCX, PDF, Xml, RtfWithSummary, DocWithSummary, DocxWithSummary, PdfWithSummary, Wdf or WdfWithSummary.

`CompareOptions`
A string containing all of the desired rendering options. To use the default server-side options pass an empty (non-null) string.

Return Value:

CompareResults structure containing the RTF/DOC/DOCX/PDF/WDF redline (byte array) and XML change summary (string).

Name: **ExecuteEx**

Description: Perform a direct comparison of two documents using bundled parameters.

```
public CompareResults ExecuteEx(ExecuteParams execParams)
```

Parameters:

execParams

A structure which contains all of the comparison parameters allowing them to be maintained as a single reference. Includes Original, Modified, CompareOptions and ResponseOption fields.

Return Value:

CompareResults structure containing the RTF/DOC/DOCX/PDF/WDF redline (byte array) and XML change summary (string).

Name: **GetVersion**

Description: Retrieve the version number of the Workshare Compare service.

```
public string GetVersion()
```

Return Value:

A string representation of the current service version.

Name: **GetCompositorVersion**

Description: Retrieve the version number of the core comparison module.

```
public string GetCompositorVersion()
```

Return Value:

A string representation of the current core comparison module version.

Name: **SetOptionsSet**

Description: Set the default server-side option set.

```
public bool SetOptionsSet(string sOptionsSet)
```

Parameters:

sOptionsSet

The name of the server-side option set used to specify the default response options when an Execute call passes and empty CompareOptions string.

Return Value:

true if the named option set exists on the server.

Name: `GetOptionsSet`

Description: Retrieve the current option set.

```
public string GetOptionsSet()
```

Return Value:

The name of the server-side option set used to specify the default response options when an Execute call passes and empty CompareOptions string.

Note that Available server-side option sets are configured within the service's web.config file.

Chapter 5. Compare Control DLL Methods

This chapter describes the Workshare Compare service Control DLL methods exposed by the ICompareService interface. The ICompareService interface can be imported by adding a reference to the Document.Services.Compare.Control.Dll assembly within your Microsoft Visual Studio project. It includes the following sections:

- **Public Constructors**, below, describes the methods used to create an object implementing the ICompareService interface.
- **Public Methods and Properties**, page 30, describes the public methods of Workshare Compare service.

Public Constructors

The following static methods can be used to return an object implementing the ICompareService interface.

Name: **`CompareService::CreateHttpService`**

Description: Creates a client-side object which can call Workshare Compare service methods via an HTTP connection. (Note that this method does not take a port parameter and should only be used to connect to a service installed within IIS and using the default port 80.)

```
public ICompareService CompareService::CreateHttpService( string host )
```

Parameters:

`host`

The name of the server on which Workshare Compare service is installed.

Return Value:

An ICompareService interface providing managed, client-side, access to the service methods.

Name: **`CompareService::CreateHttpService`**

Description: Creates a client-side object which can call Workshare Compare service methods via an HTTP connection.

```
public ICompareService CompareService::CreateHttpService( string host, int port )
```

Parameters:

`host`

The name of the server on which Workshare Compare service is installed.

`port`

The port number on which Workshare Compare service is installed.

Return Value:

An ICompareService interface providing managed, client-side, access to the service methods.

Name: **`CompareService::CreateTcpService`**

Description: Creates a client-side object which can call Workshare Compare service methods via a TCP connection.

```
public ICompareService CompareService::CreateTcpService( string host, int port )
```

Parameters:

`host`
The name of the server on which Workshare Compare service is installed.

`port`
The port number on which Workshare Compare service is installed.

Return Value:

An ICompareService interface providing managed, client-side, access to the service methods.

Name: **`CompareService::CreateNamedPipeService`**

Description: Creates an object which can call Workshare Compare service methods via a Named Pipe. (Note that Named Pipes can only be used by clients which are running on the same server as the Workshare Compare service. For example, ASP web pages. Named Pipes provide significantly faster data transport than HTTP or TCP.)

```
public ICompareService CompareService::CreateNamedPipeService()
```

Return Value:

An ICompareService interface providing managed access to the service methods.

Public Methods and Properties

The following table shows the methods of the I**CompareS**ervice interface and a brief description of each.

Name: **`SetClientCredentials`**

Description: Sets the credentials used to authorize the user. All clients must call this method before calling any of the Compare, Benchmark or VerifyConnection methods.

```
public void SetClientCredentials(string user, string password, string domain)
```

Parameters:

`user`
Username for a Windows account which can be validated by the server.

`password`
Password for the specified Windows account which can be validated by the server.

`domain`
Domain name for a Windows account which can be validated by the server.

Name: **SetTimeouts**

Description: Performs a direct comparison of two documents.

```
public void SetTimeouts(int minsOpen, int minsClose, int minsSend, int minsReceive)
```

Parameters:

minsOpen

The timeout in minutes used when opening a connection to the server (default = 1).

minsClose

The timeout in minutes used when closing a connection to the server (default = 1).

minsSend

The timeout in minutes used when sending data to the server (default = 5).

minsReceive

The timeout in minutes used when the server sends data back to the client (default =5).

Name: **VerifyConnection**

Description: Tests that a successful connection can be made to the server using the current settings. Also returns version details for the Workshare Compare service and the core Workshare Comparison Engine.

```
public bool VerifyConnection(out string serviceVersion, out string compositorVersion)
```

Parameters:

serviceVersion

A string containing the version of the Workshare Compare service.

compositorVersion

A string containing the version number of the Workshare Comparison Engine.

Return Value:

A boolean value indicating whether or not it was possible to successfully connect to the server.

Name: **Compare**

Description: Compares two documents and returns a redline, WDF and/or change summary.

```
public CompareResults Compare(Stream original, Stream modified)
```

Parameters:

original

An open input stream providing access to the original copy of the document to be compared.

modified

An open input stream providing access to the modified copy of the document to be compared.

Return Value:

A CompareResults object containing a redline and/or XML change summary.

Name: **`UseChunking`**

Description: Determines whether the Control DLL should send data to the server in a single unit or should break it into chunks in order to avoid issues caused by large documents (default = true).

```
public bool UseChunking
```

Name: **`ChunkSize`**

Description: When `UseChunking` is true, ChunkSize determines the number of bytes sent to the server in each data chunk (default = 1024 x 1024 = 1Mb).

```
public int ChunkSize
```

Name: **`ComparisonOutput`**

Description: Gets/sSets the output type(s) which the server should return from a comparison.

```
public ResponseOptions ComparisonOutput
```

Parameters:

ResponseOptions can be one of the following enumerated values: Rtf, DOC, DOCX, PDF, Xml, RtfWithSummary, DocWithSummary, DocXWithSummary, PdfWithSummary, Wdf, WdfWithSummary

Name: **`CompareOptions`**

Description: Used to get/set the rendering set options for a comparison.

```
public string CompareOptions
```

Parameters:

A string containing the rendering set options. This can be loaded from one of the pre-defined set of options which are installed with the service.

Name: **`TransportProtocol`**

Description: Gets the protocol used for transporting data to/from the server.

```
public TransportProtocolEnum TransportProtocol
```

Parameters:

TransportProtocolEnum is one of the following enumerated values: Http, Tcp, NamedPipe

Chapter 6. XML Change Schema

This chapter describes the XML schema for the Change Summary produced with a comparison. It includes the following sections:

- **Introduction**, below, introduces the XML schema.
- **XML Change Schema**, below, describes the detailed format of the change schema.

Introduction

This schema is derived from an instance document of the change summary XML produced by Workshare Compare service. The XML output generated is not a full redline, but a summary of the differences between the original and modified documents submitted to the service, based on the comparison rendering set used.

A total number of differences between the two source documents will be displayed at the start of the XML:

<ChangeSet ChangeCount="**78**" xmlns="">

Each Change Group has a Character Set (CharSet) attribute, as defined on page 36:

<ChangeGroup Number="**0**" FirstChange="**1**" LastChange="**3**" ChangeHash="**-715526815**"
ContentHash="**1252279543**" HashCode="**1252279543**" TableStartPos="**-1**" UnmatchedTableNum="**-1**"
CharSet="**1**">

For each change, the Type attribute relates to the Change Type Values listed on page 37:

<Change Number="**1**" Type="**0**" HashCode="**-155916924**" PositionHash="**-155916924**" Item="**Text**">

XML Change Schema

```
<?xml version="1.0" encoding="utf-8" ?>
<xs:schema        elementFormDefault="qualified"
                  attributeFormDefault="unqualified"
                  xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:complexType name="Position">
    <xs:attribute name="CPosStart" type="xs:int" use="required" />
    <xs:attribute name="CPosParaStart" type="xs:int" use="required" />
  </xs:complexType>

  <xs:complexType name="Identity">
    <xs:sequence>
```

```
    <xs:element name="Position" type="Position" minOccurs="0"/>
    <xs:element name="PlainText" type="xs:string" />
  </xs:sequence>
  <xs:attribute name="Number" type="xs:int" use="required" />
  <xs:attribute name="HashCode" type="xs:int" use="required" />
  <xs:attribute name="Type" type="xs:int" use="required" />
</xs:complexType>

<xs:complexType name="Change">
  <xs:sequence>
    <xs:element name="Position" type="Position" minOccurs="0"/>
    <xs:element name="PlainText" type="xs:string" />
  </xs:sequence>
  <xs:attribute name="Number" type="xs:int" use="required" />
  <xs:attribute name="Type" type="xs:int" use="required" />
  <xs:attribute name="HashCode" type="xs:int" use="required" />
  <xs:attribute name="PositionHash" type="xs:int" use="required" />
  <xs:attribute name="Item" type="xs:string" use="required" />
  <xs:attribute name="MoveXRef" type="xs:int" use="optional" />
  <xs:attribute name="LastBkmk" type="xs:string" use="optional" />
  <xs:attribute name="Row" type="xs:int" use="optional" />
  <xs:attribute name="Col" type="xs:int" use="optional" />
</xs:complexType>

<xs:complexType name="ChangeGroup">
  <xs:sequence>
    <xs:element name="ParaStartText" type="xs:string" minOccurs="0"/>
    <xs:choice maxOccurs="unbounded">
      <xs:element name="Identity" type="Identity"/>
      <xs:element name="Change" type="Change"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="Number" type="xs:int" use="required" />
  <xs:attribute name="FirstChange" type="xs:int" use="required" />
  <xs:attribute name="LastChange" type="xs:int" use="required" />
  <xs:attribute name="ChangeHash" type="xs:int" use="required" />
  <xs:attribute name="ContentHash" type="xs:int" use="required" />
  <xs:attribute name="HashCode" type="xs:int" use="required" />
  <xs:attribute name="TableStartPos" type="xs:int" use="required" />
  <xs:attribute name="UnmatchedTableNum" type="xs:int" use="required" />
```

```
    <xs:attribute name="CharSet" type="xs:unsignedByte" use="required" />
  </xs:complexType>

  <xs:complexType name="Body">
    <xs:sequence>
      <xs:element maxOccurs="unbounded" name="ChangeGroup" type="ChangeGroup"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="ChangeSet">
    <xs:sequence>
      <xs:element name="Header" />
      <xs:element name="Body" type="Body"/>
    </xs:sequence>
    <xs:attribute name="ChangeCount" type="xs:int" use="required" />
  </xs:complexType>

  <xs:element name="ChangeSet" type="ChangeSet"/>
</xs:schema>
```

Character Set Values

ANSI_CHARSET	0
DEFAULT_CHARSET	1
SYMBOL_CHARSET	2
SHIFTJIS_CHARSET	128
HANGEUL_CHARSET	129
HANGUL_CHARSET	129
GB2312_CHARSET	134
CHINESEBIG5_CHARSET	136
OEM_CHARSET	255
JOHAB_CHARSET	130
HEBREW_CHARSET	177
ARABIC_CHARSET	178
GREEK_CHARSET	161
TURKISH_CHARSET	162
VIETNAMESE_CHARSET	163
THAI_CHARSET	222
EASTEUROPE_CHARSET	238
RUSSIAN_CHARSET	204
MAC_CHARSET	77
BALTIC_CHARSET	186

Change Type Values

DV_STYLE_NONE	1
DV_STYLE_DELETION	0
DV_STYLE_MOVESOURCE	1
DV_STYLE_MOVEDESTINATION	2
DV_STYLE_INSERTION	3
DV_STYLE_FORMAT_CHANGE	4
DV_STYLE_MATCH	5
DV_STYLE_CELLINSERTED	6
DV_STYLE_CELLDELETED	7
DV_STYLE_CELLMERGED	8
DV_STYLE_CELLISPADDING	9
DV_STYLE_CHANGENUMBER	10
DV_STYLE_DELIMITER	11
DV_STYLE_CHANGED_TEXT	12
DV_STYLE_MOVEDDELETION	13
DV_STYLE_STYLECHANGE_TEXT	14
DV_STYLE_STYLECHANGE_LABEL	15
DV_STYLE_COMMENT	16
DV_STYLE_INSERTEDCOMMENT	17
DV_STYLE_DELETEDCOMMENT	18
DV_STYLE_CHANGED_COMMENT	19
DV_STYLE_COUNT	20

Chapter 7. Resources

The following websites may provide useful information.

Microsoft Web Services Home

Web Services Developer Center

http://msdn.microsoft.com/webservices/default.aspx

Microsoft Windows Communication Foundation

Online MSDN documentation providing a conceptual overview, tutorials, samples and tools for WCF development.

http://msdn2.microsoft.com/en-us/library/ms735119.aspx

Documents, forums and samples relating to the WCF and related .NET 3.0 technologies.

http://wcf.netfx3.com/

Consuming Services Using a WCF Client

Microsoft 'How To' guide for creating a WCF client application.

http://msdn2.microsoft.com/en-us/library/ms734691.aspx