WORKSHARE COMPARE SERVER 8
Developer Guide

July 2016

Company Information

Workshare Compare Server Developer Guide

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

Workshare Inc. (USA)
625 Market Street, 15th Floor
San Francisco
CA 94105
USA

Workshare Website: www.workshare.com

Trademarks

Trademarked names may appear throughout this guide. 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 this guide and any 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 this 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

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**, page 5, introduces Workshare Compare Server.
- **Prerequisites**, page 5, 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.
- **Sample Applications**, page 7, explains where to download the sample applications for 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*.

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 7 or later, Microsoft Windows Server 2008 R2 or later
- 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:

- **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 RedlineML or 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.*

- **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/comparewebservice.svc)

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.*

Sample Applications

Sample Workshare Compare Server applications in C#, Java, ASP and Python can be downloaded from https://github.com/workshare/compare-service.samples.

The following samples are available:

- Document.Services.Compare.Sample: C# application connects to the Workshare Compare service using a direct synchronous request/response method.
- AdvancedWebSample: ASP application demonstrates using the Workshare Compare service Control DLL to connect to the server and perform synchronous comparisons
- BasicWebSample: ASP application demonstrates how to use WCF in order to connect directly to the Workshare Compare service.
- ConfigPageSample: ASP application containing the full source code for the Administration Dashboard and demonstrates the more advanced features of the Workshare Compare service.
- WebAdmin: ASP application that provides a simple administrative interface showing how the web client site may be monitored and controlled by a group of administrative users.
- WebClient: ASP application that provides a simple example of a user facing site that provides on-demand comparison functionality to authenticated users only
- comparews (java): 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.
- compare5Client.py (python): python script that connects to the Workshare Compare service using a direct synchronous request/response method.

Chapter 2. Sample Code

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

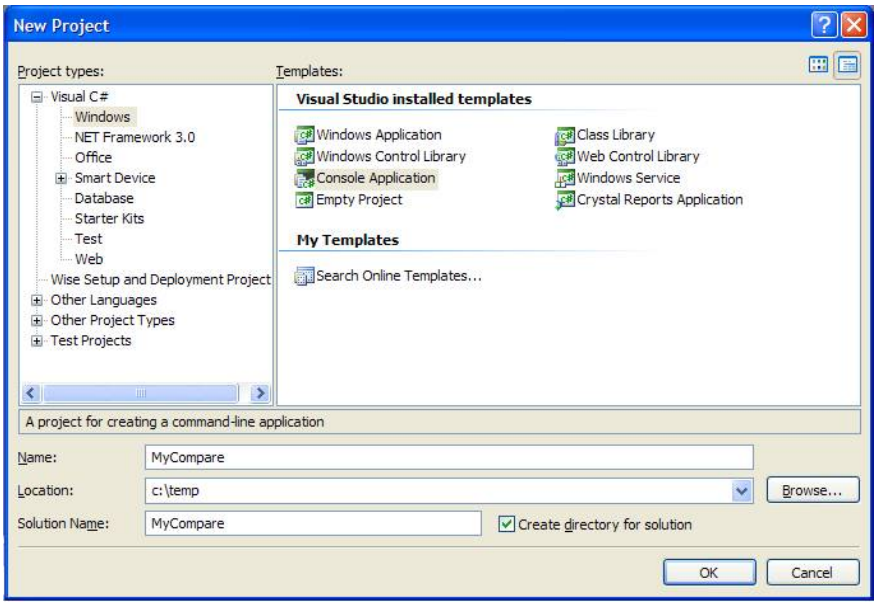
- **Creating a Console Application**, page 9, describes how to create a console application using C#.
- **Service Creation and Authentication**, page 11, provides an example of how client code can connect to Workshare Compare service.
- **Comparison**, page 11, provides an example of how client code 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:

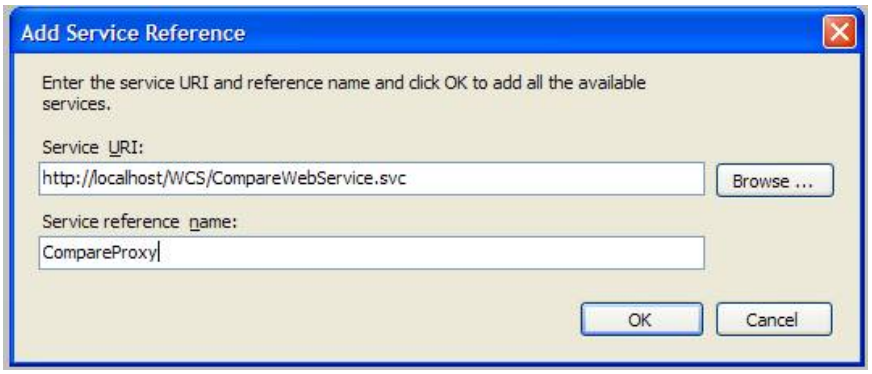
1. Create a new project in Visual Studio.NET by selecting *File* > **New** > **Project**. Enter **MyCompare** as the project name and select the location of the project.



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**.



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.

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 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 3. 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**, page 14, describes the public constructors of Workshare Compare service.
- **Public Methods**, page 14, 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 tables 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 4. 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**, page 18, describes the methods used to create an object implementing the ICompareService interface.
- **Public Methods and Properties**, page 20, 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: `CompareEx`

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

```
public CompareResults Compare(Stream original, Stream modified,
DocumentInfo originalDocumentInfo, DocumentInfo modifiedDocumentInfo)
```

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.

`originalDocumentInfo`
Metadata regarding the original document which can be included in a redline summary.

`modifiedDocumentInfo`
Metadata regarding the modified document which can be included in a redline summary.

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 5. Change Summary Information

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

- **RedlineML**, page 25, describes the RedlineML change summary schema.
- **XML**, page 31, describes the old XML change summary schema.

RedlineML

RedlineML is now the preferred format for extracting change summary information. It contains the entire content of the Redline document in an easy-to-work-with XML format.

RedlineML Schema

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema targetNamespace="http://workshare.com/2010/RedlineML"
    elementFormDefault="qualified"
    xmlns="http://workshare.com/2010/RedlineML"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
>

  <xs:element name="document" type="documentT"/>

  <xs:simpleType name="cellStatusT" final="restriction" >
    <xs:restriction base="xs:string">
      <xs:enumeration value="normal" />
      <xs:enumeration value="inserted" />
      <xs:enumeration value="deleted" />
      <xs:enumeration value="moveSource" />
      <xs:enumeration value="moveDestinate"/>
      <xs:enumeration value="dead"/>
      <xs:enumeration value="padding"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:group name="content">
    <xs:sequence>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="paraMarker"/>
        <xs:element ref="change"/>
        <xs:element ref="field"/>
        <xs:element ref="bkmk"/>
        <xs:element name="table" type="tableT"/>
```

```
        <xs:element name="shape" type="shapeT"/>
        <xs:element name="blob" type="blobT"/>
        <xs:element name="pict" type="blobT"/>
        <xs:element name="run" type="runT"/>
        <xs:element ref="endNote"/>
        <xs:element ref="footNote"/>
        <xs:element ref="textbox"/>
        <xs:element ref="comment"/>
      </xs:choice>
    </xs:sequence>
  </xs:group>

  <xs:attributeGroup name="insertedDeletedAttrs">
    <xs:attribute name="isInserted" type="xs:boolean" use="optional"
default="false"/>
    <xs:attribute name="isDeleted" type="xs:boolean" use="optional"
default="false"/>
  </xs:attributeGroup>

  <xs:group name="section">
    <xs:sequence>
      <xs:element type="sectionMarkerT" name="sectionMarker"/>
      <xs:group ref="content"/>
    </xs:sequence>
  </xs:group>

  <xs:group name="changeContent">
    <xs:sequence>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="paraMarker"/>
        <xs:element ref="field"/>
        <xs:element ref="bkmk"/>
        <xs:element name="shape" type="shapeT"/>
        <xs:element name="blob" type="blobT"/>
        <xs:element name="pict" type="blobT"/>
        <xs:element name="run" type="runT"/>
```

```
        <xs:element ref="endNote"/>
        <xs:element ref="footNote"/>
        <xs:element ref="textbox"/>
        <xs:element ref="comment"/>
      </xs:choice>
    </xs:sequence>
  </xs:group>

  <xs:complexType name="shapeT">
    <xs:group ref="content"/>
    <xs:attributeGroup ref="insertedDeletedAttrs"/>
  </xs:complexType>

  <xs:complexType name="blobT">
    <xs:attributeGroup ref="insertedDeletedAttrs"/>
  </xs:complexType>

  <xs:element name="change" type="changeT"/>
  <xs:complexType name="changeT" >
    <xs:group ref="changeContent"/>
    <xs:attribute name="number" type="xs:integer" use="required"/>
    <xs:attribute name="type" type="xs:integer" use="required"/>
    <xs:attribute name="crossref" type="xs:integer" use="optional"/>
  </xs:complexType>

  <xs:complexType name="documentT">
    <xs:sequence>
      <xs:group ref="content"/>
      <!--sadly the compositor can spit out content before its first
section marker-->
    <xs:sequence minOccurs="1" maxOccurs="unbounded">
      <xs:group ref="section"/>
    </xs:sequence>
    </xs:sequence>
    <xs:anyAttribute/>
  </xs:complexType>
```

```
  <xs:complexType name="sectionMarkerT">
    <xs:sequence>
      <xs:sequence minOccurs="0" maxOccurs="unbounded">
        <xs:choice>
        <xs:element name="header" type="headerfooterT"/>
        <xs:element name="footer" type="headerfooterT"/>
        </xs:choice>
      </xs:sequence>
    </xs:sequence>
    <xs:attributeGroup ref="insertedDeletedAttrs"/>
  </xs:complexType>

  <xs:complexType name="tableT">
    <xs:sequence minOccurs="1" maxOccurs="unbounded">
      <xs:element name="row" type="rowT"/>
    </xs:sequence>
    <xs:attributeGroup ref="insertedDeletedAttrs"/>
  </xs:complexType>

  <xs:complexType name="rowT">
    <xs:sequence minOccurs="1" maxOccurs="unbounded">
      <xs:element name="cell" type="cellT"/>
    </xs:sequence>
    <xs:attributeGroup ref="insertedDeletedAttrs"/>
  </xs:complexType>

  <xs:complexType name="cellT">
    <xs:group ref="content"/>
    <xs:attribute name="cellStatus" type="cellStatusT" use="optional"
default="normal"/>
    <xs:attribute name="isInsertedColumn" type="xs:boolean"
use="optional" default="false"/>
    <xs:attribute name="isDeletedColumn" type="xs:boolean"
use="optional" default="false"/>
    <xs:attribute name="column" type="xs:integer" use="optional"/>
```

```
    <xs:attribute name="spanInfoOriginal" type="xs:string"
use="optional"/>
    <xs:attribute name="spanInfoModified" type="xs:string"
use="optional"/>
    <xs:attribute name="spannedInOriginal" type="xs:boolean"
use="optional"/>
    <xs:attribute name="spannedInModified" type="xs:boolean"
use="optional"/>
    <xs:attribute name="isMerged" type="xs:boolean" use="optional"/>
  </xs:complexType>

  <xs:element name="textbox" type="textboxT"/>
  <xs:complexType name="textboxT">
    <xs:group ref="content"/>
  </xs:complexType>

  <xs:element name="paraMarker" type="paraMarkerT"/>
  <xs:complexType name="paraMarkerT">
    <xs:attribute name="listNumber" type="xs:string" use="optional"/>
    <xs:attributeGroup ref="insertedDeletedAttrs"/>
    <xs:attribute name="isInsertedListItem" type="xs:boolean"
use="optional" default="false"/>
    <xs:attribute name="listLevel" type="xs:int" use="optional"/>
    <xs:attribute name="listId" type="xs:int" use="optional"/>
  </xs:complexType>

  <xs:complexType name="runT" mixed="true">
    <xs:attribute name="font" type="xs:string" use="required"/>
    <xs:attribute name="wasListNum" type="xs:boolean" default="false"
use="optional"/>
    <xs:attribute name="wasField" type="xs:boolean" default="false"
use="optional"/>
    <xs:attribute name="rtl" type="xs:boolean" default="false"
use="optional"/>
  </xs:complexType>

  <xs:complexType name="commentT">
    <xs:group ref="content"/>
    <xs:attributeGroup ref="insertedDeletedAttrs"/>
```

```
</xs:complexType>

<xs:element name="comment" type="commentT"/>

<xs:element name="footNote" type="footEndNoteT" />
<xs:element name="endNote" type="footEndNoteT" />
<xs:complexType name="footEndNoteT">
  <xs:group ref="content"/>
  <xs:attributeGroup ref="insertedDeletedAttrs"/>
</xs:complexType>

<xs:complexType name="headerfooterT">
  <xs:group ref="content"/>
  <xs:attribute name="type" type="hdrftrType" use="required"/>
  <xs:attributeGroup ref="insertedDeletedAttrs"/>
</xs:complexType>

<xs:simpleType name="hdrftrType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="left"/>
    <xs:enumeration value="right"/>
    <xs:enumeration value="first"/>
    <xs:enumeration value="main"/>
  </xs:restriction>
</xs:simpleType>

<xs:element name="field" type="fieldT"/>
<xs:complexType name="fieldT">
  <xs:sequence>
    <xs:element type="fieldCodeT" name="fieldCode"/>
    <xs:element ref="fieldResult" minOccurs="0"/>
  </xs:sequence>
  <xs:attributeGroup ref="insertedDeletedAttrs"/>
</xs:complexType>

<xs:complexType name="fieldCodeT">
```

```
    <xs:sequence maxOccurs="unbounded" minOccurs="1">
      <xs:choice>
        <xs:element ref="field"/>
        <xs:element name="run" type="runT"/>
        <xs:element ref="paraMarker"/>
        <xs:element name="blob" type="blobT"/>
      </xs:choice>
    </xs:sequence>
  </xs:complexType>

  <xs:element name="fieldResult" type="fieldResultT"/>
  <xs:complexType name="fieldResultT">
    <xs:group ref="content"/>
  </xs:complexType>

  <xs:element name="bkmk" type="bkmkT"/>
  <xs:complexType name="bkmkT">
    <xs:attribute name="name" type="xs:string" use="required"/>
    <xs:attribute name="start" type="xs:boolean" use="required"/>
  </xs:complexType>

</xs:schema>
```

XML

The former change summary XML is deprecated and may be removed in a future version.

XML Schema

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 34:

```
<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 35:

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

```
<?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 6. Resources

The following websites may provide useful information.

Microsoft Web Services Home

Web Services Developer Center

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

Microsoft Windows Communication Foundation

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

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

Consuming Services Using a WCF Client

Microsoft ‘How To’ guide for creating a WCF client application.

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