

Workshare Compare Server 9.7

Developer Guide

Table of Contents

Chapter 1: Introduction.....	4
Introducing Workshare Compare Server.....	5
Communicating with Workshare Compare Server	5
Chapter 2: Documentation and Examples	6
Overview	7
Types of Comparison	7
Choosing which type of comparison to run	7
Immediate Comparisons	10
GET request example	10
POST request example	13
Queued Asynchronous Comparisons	16
Making a queued comparison request.....	16
Handling the response to a queued comparison request.....	17
Data Storage on Workshare Compare Server.....	18
DVJS Comparisons	19
What is DVJS?	19
Getting started with DVJS	19
The DVJS APIs	20
Starting DVJS comparisons.....	20
Displaying a DVJS comparison	21
Customizing DVJS	22
Customizing which elements of the UI are visible	23
Customizing the language of the UI.....	23
Customizing the toolbar.....	25
Customizing the behavior of the toolbar	26
Accept change functionality.....	30
Accept/Reject Changes from the API.....	33
Merging Multiple Modified Documents	34
POST request example	35
Appendix A. Change Summary Information	38
RedlineJSON.....	39
RedlineML	53
RedlineML Schema	53

Character set values	60
ChangeT values	61
Code Samples	61
Extracting change summary information.....	61

Chapter 1: Introduction

This chapter describes the functionality provided by Workshare Compare Server and how to communicate with Compare Server.

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 .NET application that consists of Workshare's tried and tested comparison engine, and a series of RESTful APIs. It is a fast, performant and robust document comparison machine that accepts two source documents (RTF, DOC, DOCX, PDF, TXT, HTML), compares them, and provides the output as either:

1. A document, in the following formats: RTF, DOC, DOCX, PDF, TXT or track change document.
2. A URL, that loads the comparison result into Workshare's browser-based UI.
3. An JSON or XML change summary, detailing only the changes that have occurred, and associated information about them.

Compare Server's outputs are known as 'redlines', whether shown as a document or within Workshare's UI.

Using Compare Server, you can integrate Workshare's comparison technology into your application to:

- 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.

Once installed, you can launch the bundled Swagger documentation into the browser, read about each of the endpoints, and try them out. Compare Server comes bundled with sample documents to facilitate this.

Additionally, the merge feature will accept an original document and multiple modified documents and produce a merged document with all the changes shown as track changes.

Communicating with Workshare Compare Server

You can communicate with Workshare Compare Server over HTTP REST calls. To fully benefit from this guide, you should have an understanding of RESTful APIs.

Chapter 2: Documentation and Examples

This chapter provides examples of GET and POST requests and information about running different types of comparisons.

Overview

The RESTful API endpoints are documented here: [http://\[your server name\]/swagger](http://[your server name]/swagger).

Types of Comparison

- **Classic vs DVJS comparisons**
 - Classic comparisons are performed by making calls to the `api/Compare` endpoint and the result is a comparison in document format (DOCX/DOC/PDF) or in data format (JSON as RedlineJSON or XML as RedlineML). The comparison result is returned to the calling code which can take appropriate actions to use it or make it available to an end user.
 - DVJS comparisons are performed by making calls to the `api/UICompare` endpoint and the result is a URL which can be viewed in a web browser to show the comparison document and change summary information to the user.
- **POST vs GET comparisons**
 - POST comparisons are performed by sending the content of the two documents to be compared to Compare Server as part of an HTTP POST request in multipart/form-data format.
 - GET comparisons are performed by sending URLs from which the documents to be compared can be retrieved to Compare Server as parameters on an HTTP GET request.
- **Immediate vs asynchronous queued comparisons**
 - Immediate comparisons are performed by making a single HTTP request (either GET or POST). The server performs the comparison in response to receiving the request and the result of the comparison is returned as the response to the HTTP request. (Note that certain administrative configuration options may cause the request to be delayed or refused if the server is too busy.)
 - Asynchronous queued comparisons are performed by making an HTTP request (either GET or POST) to the 'Enqueue' method. For example, `/api/Compare/Enqueue` or `/api/UICompare/Enqueue`. On receipt of the request the server will place the comparison in its queue and return a URL that the client code can use to check for the completion status of the comparison. Comparisons are performed in the order that they are queued. When a comparison is complete, the response to the status request is a URL that will redirect to the location from which the result can be downloaded.

Choosing which type of comparison to run

The following questions can help you choose the most suitable way to integrate Compare Server for your use case.

Question 1: Do you have more than one modified document?

Answers	Solution
Yes, I have a single original document and multiple modified versions from different authors.	See Merging Multiple Modified Documents for further details.

Question 2: Where is the location of the documents you need to compare?

Answers	Solution
Your documents are already located on a web server or other web-based system and you can construct a URL that will allow them to be downloaded without encountering additional security prompts or requiring user interaction.	In this case you can compare documents using the HTTP GET method. You will need to construct URLs that allow each of the documents to be compared to be downloaded, and pass those URLs as parameters to the comparison request. Compare Server will download the documents using the provided URLs and then perform the comparison.
Your documents are located on a web server or web-based system, but require additional information (cookies or login) to download.	In any of these cases you will need to use the HTTP POST method when performing a comparison. Your code will need to fetch the content of the documents and include the content of both in the comparison POST request, which should be made in multipart/form-data format.
Your documents are located on the local disk or storage of the device requesting the comparison.	
Your documents are located in some other location.	

Question 3: How do you intend to use the comparison result?

Answers	Solution
You want to show the comparison document to the user requesting it in their web browser.	In this case you should make a comparison request to the UICompare endpoint (DVJS comparison) and redirect the browser location to the URL specified in the 'Location:' header of the response. This will load the comparison into the browser and allow the user to view the document and summary of changes. See DVJS Comparisons for further details on customizing the user experience.

Answers	Solution
You want to store or process the comparison result as a document.	In these cases you should make a comparison request to the Compare endpoint (classic comparison) and choose an appropriate response format ('Pdf' or 'Docx'). Your code will be able to retrieve the comparison result.
You want to allow the user to download the comparison result as a document.	
You want to process data that summarizes the changes between the two documents that are being compared.	In this case you should make a comparison request to the Compare endpoint (classic comparison) and choose the RedlineJSON (comparison data as JSON) or RedlineML (comparison data as XML) response format and then process the resulting data as appropriate.

Question 4: How urgently do you need the result?

Answers	Solution
The comparison is being run as part of a background task and reliability is more important than turnaround time.	In this case you should use the asynchronous queued comparison option to make requests. Send requests to <code>/api/Compare/Enqueue</code> or <code>/api/UICompare/Enqueue</code> to begin a comparison and then poll the status link for completion. Note that your server must be configured by the server administrator to use an SQL database to store the queued jobs otherwise jobs may be lost if the server or IIS restarts (see <i>Compare Server Admin Guide</i>).
The comparison result is needed as soon as possible.	In this case you should use the immediate comparison options, sending requests to <code>/api/Compare</code> or <code>/api/UICompare</code> . However see also questions 4 and 5.

Question 5: Will you be comparing very large documents or large PDF documents?

Answers	Solution
Yes.	For these types of comparisons, you should consider using the asynchronous queued requests. This type of comparison may take some time to complete (from 30 seconds to many minutes) and using immediate comparison requests may lead to timeouts in web requests that are intermittent and difficult to debug. Using asynchronous queued requests avoids this problem.

Question 6: How do you expect the server to be utilized?

Answers	Solution
You expect the server to have to deal with peaks of load which exceed its capability to process comparisons. For example, you intend to perform batch processing and submit large numbers of comparisons in a short period of time.	In this case you should consider using asynchronous queued requests to ensure that peaks of load are handled well with no risk of dropped requests.

Immediate Comparisons

The immediate comparison API requests will all attempt to start the execution of the comparison immediately when the request is received and will return the result of the comparison as the response to the original HTTP request.

GET request example

In this example, the request compares two documents from source URLs and applies the specified rendering options to the resulting redline. The redline is then saved to the current directory as an RTF called **GetTest.rtf**.

Example CURL command

```
curl -X GET --header 'Accept: application/rtf' 'http://[your server name]/api/Compare?originalSourceUrl=http%3A%2F%2Finstall.workshare.com%2Fcompare%2FSampleOriginal.doc&modifiedSourceUrl=http%3A%2F%2Finstall.workshare.com%2Fcompare%2FSampleModified.doc&outputFormat=Rtf&renderingOptions=DETECT%20LIST%20NUMBERING%20CHANGES%3D1%3BCOMPARE%20HEADERS%2FFOOTERS%3D0' -o GetTest.rtf
```

Explanation of the CURL command

Snippet	Function
<code>curl</code>	Beginning the request.
<code>-X GET</code>	Type of request.
<code>--header 'Accept: application/rtf'</code>	This part of the request is not required but if used, should match the output format application. These can be found on the Swagger page.

Snippet	Function
<code>-o GetTest.rtf</code>	Tells the request to save the response as the named file in the current directory. The format should request the specified output format. For combined requests in output, this should be .zip.

Explanation of request parameters

In the example CURL command above, the request URL is as follows:

Root

`http://[your server name]/WorkshareCompareApi/api/compare`

The root will vary according to what you selected during the installation.

Rest of query:

`originalSourceUrl=http%3A%2F%2Finstall.workshare.com%2Fcompare%2FSampleOriginal.doc&modifiedSourceUrl=http%3A%2F%2Finstall.workshare.com%2Fcompare%2FSampleModified.doc&outputFormat=Rtf&renderingOptions=DETECT%20LIST%20NUMBERING%20CHANGES%3D1%3BCOMPARE%20HEADERS%2FFOOTERS%3D0'`

An explanation of each element of the request URL is in the table below.

Name	Example value	Comments
Server name	<code>http://[your server name]</code>	
Virtual Directory Path	<code>WorkshareCompareApi</code>	If you install the Workshare Compare API in the root of a new web site then this component will not be needed in the URL.
Endpoint	<code>/api/Compare</code>	

Name	Example value	Comments
<code>originalSourceUrl</code>	<code>http://install.workshare.com/compare/SampleOriginal.doc</code>	<p>The URL from which to fetch the original document. <code>http://</code> and <code>https://</code> are supported.</p> <div data-bbox="894 411 1385 506" style="border: 1px solid orange; padding: 5px;"> <p>Note 1: The files need to be available for HTTP/S requests.</p> </div> <div data-bbox="894 541 1385 800" style="border: 1px solid orange; padding: 5px;"> <p>Note 2: This parameter must be URL Encoded before it's included in your request URL. You can convert between decoded and encoded URLs at this site: http://meyerweb.com/eric/tools/dencoder/</p> </div>
<code>modifiedSourceUrl</code>	<code>http://install.workshare.com/compare/SampleModified.doc</code>	<p>The URL from which to fetch the original document. <code>http://</code> and <code>https://</code> are supported.</p> <div data-bbox="894 961 1385 1079" style="border: 1px solid orange; padding: 5px;"> <p>Note: Follow the same guidance as indicated in the notes for originalSourceUrl.</p> </div>
<code>outputFormat</code>	<p>One of: Rtf (default); Wdf; Doc; DocX; Pdf; TrackChanges; RedlineMI; RedlineMI,Rtf; RedlineMI,Wdf; RedlineMI,Doc; RedlineMI,DocX; RedlineMI,Pdf; RedlineMI,TrackChanges; RedlineJSON; RedlineJSON,Rtf; RedlineJSON,Wdf; RedlineJSON,Doc; RedlineJSON,DocX; RedlineJSON,Pdf; RedlineJSON,TrackChanges</p>	<p>The format of the returned comparison. When more than one format is specified (ie RedlineMI,DocX) the result is returned as a ZIP file containing both formats. The default output format is RTF if this option is not specified.</p> <p>WDF is the comparison format used by the Workshare Compare desktop application (comparisons created in WDF format can only be opened in the Workshare Compare desktop application).</p> <p>RedlineMI is an XML-based format and RedlineJSON is a JSON-based format that describe the changes between the two documents in an easy-to-use manner (see Appendix A: Change Summary Information).</p>

Name	Example value	Comments
renderingOptions	DETECT LIST NUMBERING CHANGES=1;COMPARE HEADERS/FOOTERS=0'	<p>Optional rendering options to customize the comparison process and the format of the comparison document.</p> <p>Rendering options are best generated by creating a rendering set using the Workshare Compare desktop application. The contents of the saved <code>.set</code> file can be passed as the value for this parameter. Semicolons can be used in place of <code>\r</code> or <code>\n</code> as line separators in this parameter.</p> <div style="border: 1px solid orange; padding: 5px; margin-top: 10px;"> <p>Note: For more information about rendering sets, see the Compare Server Rendering Set Guide.</p> </div>

POST request example

In this example, the request compares two documents from the local machine and applies the specified rendering options to the resulting redline. The redline is then saved to the current directory as a DOC called **PostTest.docx**.

Example CURL command

```
curl -X POST --header 'Content-Type: multipart/form-data' --header
'Accept: application/rtf' -F outputFormat=Docx -F
"file1=@documents/Comp2.docx" -F "file2=@documents/Comp1.docx" -F
'renderingOptions=Display Workshare Compare Footers=1; Inserted
Text Color=8388736 ' 'http://[your server name]/api/Compare' -o
PostTest.docx
```

Explanation of the CURL command

Snippet	Function
<code>curl</code>	Beginning the request.
<code>-X POST</code>	Type of request.

Snippet	Function
<code>--header 'Content-Type: multipart/form-data'</code>	Format of the request. Note: Must always be included for POST requests.
<code>--header 'Accept: application/rtf'</code>	This part of the request is not required but if used, should match the output format application. These can be found on the Swagger page.
<code>-o PostTest.docx</code>	Tells the request to save the response as the named file in the current directory. The format should request the specified output format. For combined requests in output, this should be .zip.

Explanation of request parameters

In the example CURL command above, the request URL is as follows:

```
outputFormat=Docx -F "file1=@documents/Comp2.docx" -F
"file2=@documents/Comp1.docx" -F 'renderingOptions=Display
Workshare Compare Footers=1; Inserted Text Color=8388736 '
'http://[your server name]/api/Compare
```

An explanation of each element of the request URL is in the table below.

Name	Example value	Comments
Server name	http://[your server name]	
Endpoint	/api/Compare	
originalDocument	file1=@documents/Comp2.docx	The location of the original document on the local system.
modifiedDocument	file2=@documents/Comp1.docx	The location of the modified document on the local system.

Name	Example value	Comments
<code>outputFormat</code>	One of: Rtf (default); Wdf; Doc; DocX; Pdf; TrackChanges; RedlineMI; RedlineMI,Rtf; RedlineMI,Wdf; RedlineMI,Doc; RedlineMI,DocX; RedlineMI,Pdf; RedlineMI,TrackChanges; RedlineJSON; RedlineJSON,Rtf; RedlineJSON,Wdf; RedlineJSON,Doc; RedlineJSON,DocX; RedlineJSON,Pdf; RedlineJSON,TrackChanges	<p>The format of the returned comparison. When more than one format is specified (ie RedlineMIAndXXX), the result is returned as a .zip file containing both formats. The default output format is RTF if this option is not specified.</p> <p>WDF is the comparison format used by the Workshare Compare desktop application (comparisons created in WDF format can only be opened in the Workshare Compare desktop application).</p> <p>RedlineMI is an XML-based format and RedlineJSON is a JSON-based format that describe the changes between the two documents in an easy-to-use manner (see Appendix A: Change Summary Information).</p>
<code>renderingOptions</code>	Display Workshare Compare Footers=1; Inserted Text Color=8388736	<p>Optional rendering options to customize the comparison process and the format of the comparison document.</p> <p>Rendering options are best generated by creating a rendering set using the Workshare Compare desktop application. The contents of the saved .set file can be passed as the value for this parameter. Semicolons can be used in place of \r or \n as line separators in this parameter.</p> <div style="border: 1px solid orange; padding: 5px; margin-top: 10px;"> <p>Note: For more information about rendering sets, see the Compare Server Rendering Set Guide.</p> </div>

Queued Asynchronous Comparisons

The immediate comparison API requests described above will all attempt to start the execution of the comparison immediately when the request is received and will return the result of the comparison as the response to the original HTTP request. Depending on the client being used to send the HTTP requests, timeouts may occur if the files to be compared are large or the server is busy. In some cases, it may be difficult to adjust the timeout limits of the HTTP requests being made, particularly if intermediate servers such as a proxy server are involved.

The queued, asynchronous comparison API addresses the issues with HTTP timeouts and also helps to manage server load when comparisons are submitted in batches.

Requests to the queued comparison endpoint are constructed identically to the GET and POST requests that are sent to the immediate comparison endpoints described above. The only differences are the URL to send the request to - which must have '/enqueue' appended to it - and the response to the initial request - which is no longer the comparison result.

Making a queued comparison request

To make a queued comparison request, follow the instructions in [Immediate Comparisons](#) to make either a POST or GET comparison request, but send the request to:

```
http://[your server name]/WorkshareCompareApi/api/compare/enqueue
```

If your request is valid then you will immediately receive a response with an HTTP response status code **202 Accepted** status code, indicating that the comparison that you have submitted has been allocated a place in the queue. If the request for a comparison is obviously invalid in some way (for instance only one document is included) then you will receive an appropriate error response to your request.

***Note:** Other errors (for example, the supplied URLs not being correct or the supplied documents not being in a supported format) will not generate an error when the comparison is queued as they only become apparent when the server attempts to run the comparison. In these cases, the details of the error will be retrieved via the status polling described below.*

If you submit a queued GET comparison request then the URLs that specify the location of the source documents will only be fetched once the comparison reaches the head of the queue and is ready to execute. You should therefore ensure that if these URLs are time-limited, they will not expire before the comparison reaches the head of the comparison queue.

Handling the response to a queued comparison request

The response to the queued comparison request includes a status URL.

Your code must inspect the 'location:' header of the response to recover the status URL that your client code can use to monitor the progress of the queued comparison. The location header will take the form:

```
location: http://[your server name]/WorkshareCompareApi
/api/Compare/<comparison id>/status
```

For example:

```
location: http://localhost/WorkshareCompareApi/api/Compare/
4c9ec86ec8edbadf5db41da5dd7850c695db9b2538f06216
/status
```

The status URL returned in the location header may then be polled intermittently by your code to determine when the comparison is completed. You should avoid polling continuously to avoid unnecessary load on Compare Server, although the server itself will delay responding to polling requests where the comparison is not yet complete to limit polling frequency.

Depending on the state of the comparison, you may receive one of the following responses from the comparison status request:

Response	Explanation
404 Not Found	The comparison ID in the status URL is invalid.
404 Not Found	The comparison referenced in the status URL has already completed and been downloaded.
202 Accepted	The comparison is still in the queue. The response will contain the following custom headers: <ul style="list-style-type: none"> • X-queued-for: <seconds in queue so far> • X-queue-length: <number of comparisons in queue>
202 Accepted	The comparison is being executed. The response will contain the following custom header: <ul style="list-style-type: none"> • X-processing-for: <seconds processing so far>
302 Found	The comparison has completed. The location header of the response will contain the URL from which the comparison result can be downloaded.
An error (4XX or 5XX) status code	The comparison failed. The status reason phrase will contain more detail on why the comparison failed.

Note: When the comparison completes, the **302 Found** response that redirects to the download link for the comparison result may be handled automatically by your HTTP client software. This means that your code may actually receive a **200 OK** response with the comparison result as the response body. You should either write code to cope with this eventuality or disable automatically following redirects when making HTTP requests to the status URL. You will need to check your HTTP library documentation to determine how to disable automatically following redirects.

Once a queued comparison is completed, the server will only store the comparison result for a limited period. The result is deleted after either of these two conditions is satisfied:

- One hour has passed since the comparison completed
- Fifteen minutes have passed since the first attempt to download the comparison result was initiated.

If you try to request the status or result of a comparison after it has been deleted by the server, then you will receive a **404 Not Found** response from the server.

Data Storage on Workshare Compare Server

When comparisons are performed using immediate comparison API requests, the documents to be compared and the comparison result file are not stored on the disk storage of Compare Server except (in certain use cases) as temporary files which are deleted when no longer needed.

When comparisons are performed using queued asynchronous comparison API requests, the source documents and comparison results must be stored on disk while the comparison is queued and waiting to be run and between comparison completion and download of the comparison result. The following table indicates what data is stored on disk and when it is removed.

Source documents – POST comparisons	The source documents are stored from the time that the request is submitted and are deleted when the comparison execution is finished (either successful completion or due to a failure to compare).
Source documents – GET comparison	The source documents are not stored on disk; however the URLs that are to be used to retrieve the source documents are stored in the queue database.
Comparison result – GET and POST comparisons	The comparison result is stored from the time that the comparison completes until it is deleted according to the rules above.

DVJS Comparisons

What is DVJS?

DVJS (short for DeltaView JavaScript) is a component of Workshare Compare Server that allows comparisons to be viewed and reviewed in the user's web browser. While it is simple to take a PDF format comparison document and view it in the browser, DVJS provides richer functionality such as change navigation, accept/reject changes, change categorization, print comparison and potential for customization by the integrator.

Use of DVJS only requires a modern, standards-compliant, web browser, such as Chrome, Firefox, Edge, Safari. No other software needs to be installed on the end user's PC or device.

Note: *The only version of Internet Explorer that will work correctly is version 11; older versions are not supported.*

DVJS is designed to be integrated into existing web-based applications and workflows – for instance it could be integrated into a web-based content management system or intranet portal allowing documents stored in the system to be compared without requiring them to be downloaded to a PC.

Getting started with DVJS

Compare Server ships with a very simple, sample comparison web page that allows users to select and compare two documents located on their PC or device. This also provides the best place to get started with understanding how DVJS works and how to customize and integrate it into existing systems.

To start using the sample comparison web page:

1. In your web browser, navigate to: `http://[your server name]/WorkshareCompareApi/compare`
2. Follow the instructions to select two documents and compare them.

You may, if you wish, use the sample comparison page in production, either by pointing your users to the URL given above or by taking, customizing and hosting on your own server the following files:

```
http://[your server name]/WorkshareCompareApi/Content/deltaview.js
http://[your server name]/WorkshareCompareApi/js/demo-utilities.js
http://[your server name]/WorkshareCompareApi/js/scripts.js
http://[your server name]/WorkshareCompareApi/css/demo.css
http://[your server name]/WorkshareCompareApi/css/normalize.css
```

In addition, you will need the HTML content returned from the request to:

```
http://[your server name]/WorkshareCompareApi/compare
```

You will need to adjust for any changes in the relative paths of these files and also change the 'compareServerUrl' setting in scripts.js to point to the URL of your installed Compare Server.

Note: The file 'deltaview.js' is a Workshare copyright and you must not modify it or use it except in conjunction with Workshare Compare Server's DVJS features. You may modify and use the content of the other files listed above as the basis for your own integration to the DVJS functionality.

The DVJS APIs

DVJS is based on two APIs that are implemented by Compare Server:

1. [http://\[your server name\]/WorkshareCompareApi/api/uicompare](http://[your server name]/WorkshareCompareApi/api/uicompare)

This API can be used to initiate comparisons that are to be shown in the browser using DVJS. The uicompare API has all the options of the standard compare API (GET and POST requests, immediate or queued comparisons). See below for further information about using this API.

2. [http://\[your server name\]/WorkshareCompareApi/api/comparison](http://[your server name]/WorkshareCompareApi/api/comparison)

This API provides the resources needed to render the comparison in the browser. This API is reserved for the use of the deltaview.js front end code. It is deliberately not documented and may be changed in future versions of Compare Server. You should not make calls to this API from your code. You will not be eligible for support on any issues you may encounter and future updates to Compare Server may break your implementation.

Starting DVJS comparisons

The simplest way to begin using DVJS is to perform a comparison by making a request to the uicompare API endpoint. All the options for performing a comparison against the regular compare API endpoint are available (GET and POST requests, immediate or queued comparisons). Refer to [Immediate Comparisons](#) and [Queued Asynchronous Comparisons](#) for full details on the parameters available and other details required to make a comparison request.

The response to a valid comparison request to the uicompare endpoint will have a **201 Created** response status code and will contain a 'location' header giving the URL at which the comparison result can be viewed.

In the case of queued requests to the uicompare endpoint, you must follow the procedure described in [Queued Asynchronous Comparisons](#) to check for the status of the comparison. Once the comparison is complete, the status request will return with a **302 Found** response status code and location header containing the URL at which the comparison result can be viewed.

Displaying a DVJS comparison

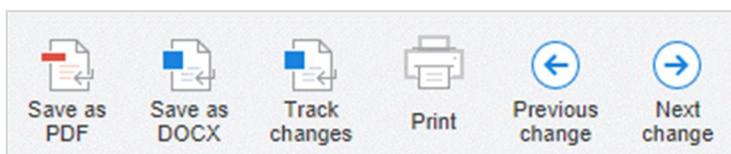
To display the comparison result to the user, the simplest approach is to browse to the URL that was obtained by performing the comparison. This will launch the DVJS redline viewer inside the browser window with no customization. The user will be able to explore the changes in the browser and download the resulting comparison in a variety of formats.

Note: The DVJS redline viewer runs in a browser. It is recommended to access it from a desktop or laptop rather than a mobile device.

The screenshot displays the DVJS redline viewer interface. At the top, there is a toolbar with icons for 'Save as PDF', 'Save as DOCX', 'Track changes', 'Print', 'Previous change', and 'Next change'. Below the toolbar is a 'Change summary' panel on the left, which lists categories of changes: CONTENT (28), NUMBERING (11), CAPITALIZATION (1), PUNCTUATION (3), FOOTNOTE (28), and MATCHING (7). A table below this lists specific changes on pages 1, 2, and 3, with checkboxes and a blue 'Accept selected changes' button at the bottom. The main area shows the original document text with redlines indicating modifications. The document title is 'DRAFT ARRANGEMENT LETTER FOR STAND-ALONE BONDS'. The text includes 'The Directors Quality Bonds Plc Issuer Plc', 'The Lead Manager and the other Managers (as defined in paragraph 2 below)', '[Date]', 'Dear Sirs/Sir/Madam.', and 'PROPOSED QUALITY BONDS PLC ISSUE BY QUALITY BONDS PLC ISSUER PLC ("the Issuer")'. An 'Introduction' section follows, starting with '1. This arrangement letter sets out the scope and limitations of the work to be performed by us in related with the above transaction, namely the proposed issue of [] ("the issue") which will involve the preparation by the issuer, and for which the issuer will be solely responsible of an offering circular in accordance with the listing rules of the'.

Note: There is no option to send the comparison via email using a desktop email client (for example Microsoft Outlook). This is not present because there is no widely supported mechanism for a web page to create an email with an attachment or content using a 'mailto:' link.

The toolbar provides the following options by default:



Customizing DVJS

For more control over the options available to the user when viewing the comparison, it is possible to create a web page which contains and customizes the DVJS viewer.

Customizations can include

- Removing options from the toolbar in order to prevent certain user actions such as downloading the comparison result.
- Adding options to the toolbar in order to allow the user to take actions with the comparison that are specific to the integration. For example:
 - Saving the comparison back into a CMS or Intranet portal system
 - Emailing the comparison via a suitable webmail platform

The code below shows a minimal HTML page that will show the DVJS UI for a comparison, which can form a starting point for customization.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8"><meta name="viewport"
content="width=device-width,initial-scale=1">

  <title>DeltaView Document Comparison</title>
  <link href="path/to/deltaview.css" rel="stylesheet">
  <script type="text/javascript"
src="path/to/deltaview.js"></script>
  <script type="text/javascript">
      document.addEventListener("DOMContentLoaded",
function(event) {
    var options = {
      deltaviewId: 'deltaview-goes-here'
    };
    window.deltaview.init(options);
    // you must get the comparisonURL to this code - maybe in
a parameter of the URL
    window.deltaview.render(comparisonURL);
  });
</script>
</head>
```

```
<body>
  <div id="deltaview-goes-here"></div>
</body>
</html>
```

In order to turn the code above into a working sample, the following changes must be made:

- Ensure that the paths to the `deltaview.js` and `deltaview.css` files are set correctly.
- Determine and implement a way for the `comparisonURL` variable to be set with the URL returned from the request to the `uicompare` endpoint. For testing purposes you may wish to perform a call to `uicompare` using the 'swagger' API test web page and then hard-wire the `comparisonURL` variable to the result.

Customizing which elements of the UI are visible

The visible elements of the UI can be customized by setting display options before calling `window.deltaview.init`. For example:

```
var options = {
  deltaviewId: 'deltaview-goes-here',
  display: {
    toolbar: true,
    statistics: false,
    redline: true,
    changeSummary: false,
  }
};
```

Customizing the language of the UI

The UI language can be changed when calling `window.deltaview.init` by passing a second parameter. For example:

```
window.deltaview.init(options, window.deltaview.languages.italian);
```

Available predefined languages are currently: english, italian, french, spanish, mandarin. You can also define your own language.

If you want to add a custom language then you can define it in an object with the following structure and pass it as the second parameter to `deltaview.init()`:

```
window.dvlanguage = {
  toolbar: {
    pdf: 'Save as PDF',
    docx: 'Save as DOCX',
    tracked: 'Tracked Changes',
    print: 'Print',
    summary: 'Change Summary',
    previous: 'Previous Change',
    next: 'Next Change'
  },
  changeList: {
    zeroState: 'Your documents have no changes.',
    totalChanges: 'total changes',
    originalDocument: 'Original',
    modifiedDocument: 'Modified',
    page: 'Page',
    changedContent: 'Changed content',
    categories: 'Categories'
  },
  changeListCategories: {
    footnote: 'footnote',
    endnote: 'endnote',
    header: 'header',
    footer: 'footer',
    //matching: 'matching', // not currently displayed
    suggested: 'suggested',
    //categorized: 'categorized', // not displayed - all
changes are children of this
    numbering: 'numbering',
    boilerplate: 'boilerplate',
    capitalization: 'capitalization',
```

```

        uncategorized: 'Content', // default first category
containing all ungrouped changes
        punctuation: 'punctuation',
        style: 'style',
        font: 'font',
        //whitespace: 'whitespace', //not currently displayed
        comment: 'comment'
    },
    loading: {
        pages: 'Loading pages...',
        comparing: 'Comparing...',
        firstDelay: 'Your documents are being compared. Please
be patient - comparing large documents can take some time.',
        secondDelay: 'Sorry for the delay, your documents are
still being compared. Please be patient - comparing large
documents can take some time.',
        error: 'Your documents have not been compared. Try
refreshing your page or start your comparison again.'
    }
};

```

Customizing the toolbar

The following buttons are defined in the default toolbar and can be shown, hidden or customized by configuring the options before calling `window.deltaview.init`.

Button	Default Action
Save as PDF	Download comparison in PDF format.
Save as DOCX	Download comparison in DOCX format.
Track changes	Download comparison in DOCX format with changes shown as tracked changes (not supported for comparisons with PDF source documents).
Email	No action (placeholder to allow email integration with webmail systems).
Print	Print the comparison.
Previous change	Navigate to previous change.
Next change	Navigate to next change.

Note: *The About button is also configurable.*

For example, to hide the pdf button and show the print button:

```
var toolbar = {
  pdf: {display: false },
  print: {display: true },
};
var options = {
  deltaviewId: 'deltaview-goes-here',
  display: {
    homeButtons: toolbar,
    toolbar: true,
    redline: true,
    statistics: false,
    changeSummary: false,
    about: true
  }
};
```

Customizing the behavior of the toolbar

Deltaview.js is written so that it can be embedded in different applications, or integrations into third party systems. These applications are needed because deltaview.js cannot initiate the comparison itself. Once the comparison is rendered then there might be a requirement to handle the actions of the toolbar buttons differently to the default behavior in a browser.

What the toolbar contains

The deltaview.js toolbar contains a number of predefined buttons. The toolbar can be customized to control which buttons are displayed and the default action of some of the buttons can be overridden.

The visibility, labelling and what will happen when the buttons are clicked is defined in the "homeButtons" part of the options object passed into `deltaview.init()`.

Visibility of buttons

The most important attribute of a button configuration is whether it is displayed or not. If it is not displayed then all other attributes are irrelevant.

Each button definition has a "display" boolean value. The default actions for the button will be used if only `display:true` is specified. The default visibility of buttons is detailed in each button's description below.

PDF button

This button causes the server to generate a PDF of the comparison redline.

Default action: "download"

Default configuration

```
"pdf" : {  
  "display" : true,  
  "text" : "Save as PDF",  
  "action" : "download"  
}
```

You can override the action with a function:

```
"pdf" : {  
  "display" : true,  
  "text" : "Save as PDF",  
  "action" : function (urlForPdf) {  
    alert('PDF can be downloaded from ' + urlForPdf);  
  }  
}
```

Note: The callback is invoked after making a call to the server to generate the PDF. Only if this call succeeds will the callback be invoked.

DOCX button

This button causes the server to generate a DOCX of the comparison redline.

Default action: "download"

Default configuration:

```
"docx" : {  
  "display" : true,  
  "text" : "Save as DOCX",  
  "action" : "download"  
}
```

You can override the action with a function:

```
"docx" : {
  "display" : true,
  "text" : "Save as DOCX",
  "action" : function (urlForDocx) {
    alert('DOCX can be downloaded from ' + urlForDocx);
  }
}
```

Note: The callback is invoked after making a call to the server to generate the DOCX. Only if this call succeeds will the callback be invoked.

Tracked button

This button causes the server to generate a DOCX of the comparison redline but with all the changes applied as tracked changes in the Word document.

Default action: “download”

Default configuration:

```
"tracked" : {
  "display" : true,
  "text" : "Track changes",
  "action" : "download"
}
```

You can override the action with a function:

```
"tracked" : {
  "display" : true,
  "text" : "Track changes",
  "action" : function (urlForTrackedDocx) {
    alert('DOCX with tracked changes can be downloaded from '
+ urlForTrackedDocx);
  }
}
```

Note: The callback is invoked after making a call to the server to generate the DOCX with tracked changes. Only if this call succeeds will the callback be invoked.

Print button

This button generates a PDF of the comparison redline on the server which is returned in a web request suitable for viewing in a browser ready for printing. If possible, the browser's print capability is automated. If not, the PDF view is opened in a new tab.

You can override the action with a function:

```
"print" : {
  "display" : true,
  "text" : Print,
  "action" : function (urlForViewablePDF) {
    alert('PDF of comparison can be viewed or printed from '
+urlForViewablePDF);
  }
}
```

Note: The callback is invoked as soon as the button is clicked.

Next button

This button navigates the UI to highlight the next change or change group. It highlights the change in the change summary and the redline.

It can be hidden:

```
"next" : {
  "display" : false,
  "text" : "Next change"
}
```

Previous button

This button navigates the UI to highlight the previous change or change group. It highlights the change in the change summary and in the redline.

It can be hidden:

```
"next" : {
  "display" : false,
  "text" : "Previous change"
}
```

Example

When a comparison is performed using one of the UICompare endpoints a URL location is returned. This is not a URL for viewing the comparison - it is the root part of URLs that deltaview.js will use to display the comparison.

We have provided a sample page that will use deltaview.js to display the comparison. This HTML page is the one that is returned when a web browser tries to access that comparison ID URL.

The source of that page contains example code that shows buttons being hidden or shown.

Accept change functionality

By default, accept change functionality is turned on so the Compare web UI is enriched with:

- Checkboxes next to changes that can be applied
- An action button at the bottom of the change summary

Once the user has selected at least one change to be applied then the action button is enabled.

The screenshot displays a 'Change summary' window. At the top, there is a title 'Change summary' and a 'Categories' dropdown menu. Below the title, a list of categories is shown: 'CONTENT (28)', 'NUMBERING (11)', 'CAPITALIZATION (1)', 'PUNCTUATION (3)', 'FOOTNOTE (28)', and 'MATCHING (7)'. The main area contains a table of changes. The first row is collapsed. The second row is expanded, showing a change on page 10. The change text is 'This letter is addressed ... to as "the Managers"'. The original text is shown in green and the new text in red. A checkbox next to this change is checked. Below this, there are two more rows of changes, one of which is partially visible. At the bottom of the window, there is a blue button labeled 'Accept selected changes'.

Page	Changed content
—	conditions, the terms
	<input checked="" type="checkbox"/> of this letter
	<input type="checkbox"/> shall prevail as between the relevant parties
10	<input type="checkbox"/> This letter is addressed ... to as "the Managers".
	<input type="checkbox"/> This letter is addressed ... to as "the Managers".
	<input type="checkbox"/> to any other party to whom our <u>if this</u> letter is shown .

Accept selected changes

When the user clicks the action button, Compare Server makes a copy of the original document and applies only the selected changes. This process can take a few seconds.

When the new file is available, the success callback function (as provided in the initialization of `deltaview.js`) is invoked with a URL location from which the file can be downloaded.

The user can then further change the selection of changes and generate subsequent files from the comparison.

The text used on the additional controls for accepting changes can be customized. You can do this by specifying replacement language strings in the language object which is provided to the `deltaview.init()` function.

The text strings for accept change will appear in the language file like this:

```
var language = {
  toolbar: {
    pdf: "Save as PDF",
    docx: "Save as DOCX",
    tracked: "Track changes",
    summary: "Change summary",
    previous: "Previous change",
    next: "Next change"
  },
  acceptedChanges: {
    workflowStates_idle: "Accept selected changes",
    workflowStates_generating: "Saving selected changes to new
document...",
    workflowStates_sending: "Saving selected changes to new
document...",
    workflowStates_success: "Changes successfully applied.",
    workflowStates_error: "Unable to apply selected changes."
  },
  changeList: {
    zeroState:
      "There are no changes between the original and modified
documents.",
    totalChanges: "total changes",
    originalDocument: "Original",
    modifiedDocument: "Modified",
    page: "Page",
    changedContent: "Changed content",
    categories: "Categories"
```

```
},
changeSummary: {
  repeatedToggler: "matching changes for "
},
changeListCategories: {
  footnote: "footnote",
  endnote: "endnote",
  header: "header",
  footer: "footer",
  matching: "matching",
  suggested: "suggested",
  categorized: "categorized",
  numbering: "numbering",
  boilerplate: "boilerplate",
  capitalization: "capitalization",
  uncategorized: "content",
  punctuation: "punctuation",
  style: "style",
  font: "font",
  whitespace: "whitespace",
  comment: "comment",
  spelling_corrections: "spelling corrections",
  word_variations: "word variations"
}
};
```

Accept/Reject Changes from the API

If you don't use DVJS to display the comparison, but write your own application, you can request Compare Server to generate a new version of the original document with selected changes accepted. To do this, you need to submit the following to Compare Server:

- Source documents (original and modified)
- Comparison (redline or redlineML)
- List of changes to accept (comma separated list of change numbers)

Your application will have to enable users to select which changes they want to accept.

Example CURL command

```
curl -X POST "http://localhost:49334/api/ApplyChanges" -H "accept: application/msword" -H "Content-Type: multipart/form-data" -F "originalDocument=@Original.docx;type=application/vnd.openxmlformats-officedocument.wordprocessingml.document" -F "modifiedDocument=@Modified.docx;type=application/vnd.openxmlformats-officedocument.wordprocessingml.document" -F "redlineML=@redlineML.xml;type=text/xml" -F "changeNumbers=1,2"
```

Explanation of request parameters

The example CURL command above includes the following parameters:

Name	Type	Comments
<code>originalDocument</code>	File	The document used as the original document in a previously performed comparison.
<code>modifiedDocument</code>	File	The document used as the modified document in a previously performed comparison.
<code>redlineML</code>	File	The result of the previously performed comparison in redlineML format.
<code>ChangeNumbers</code>	String	A list of comma separated integers corresponding to the change number to be applied. <i>Note: The change numbers refer to the numbered changes listed in the redlineML comparison.</i>
<code>asTrackChanges</code>	Boolean	If true, the output should be a document with track changes

Note: The three File parameters must be identical to those used in/from a previous comparison.

Response

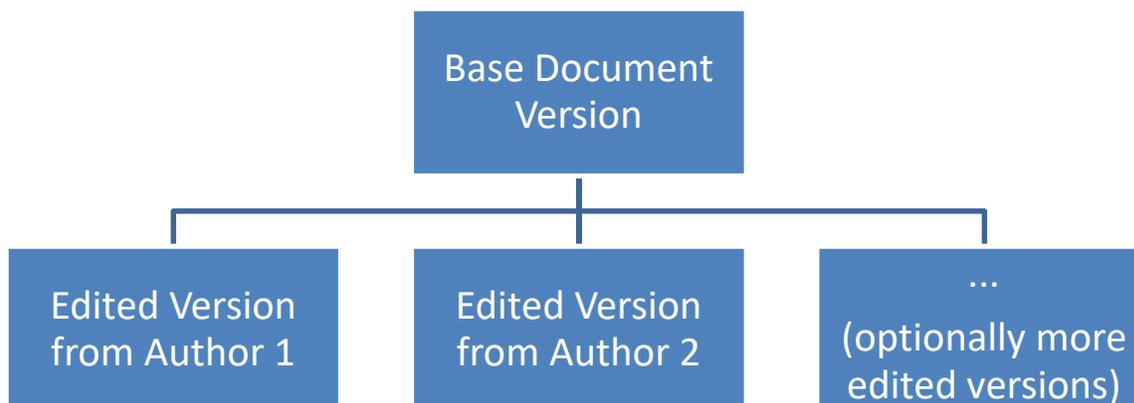
Compare Server generates a DOCX file which is a new version of the original document with the specified changes accepted. If there are any errors, they are indicated in a header:

- **changesThatFailedToApply:** If a change failed to apply, the header will include a list of change numbers that failed to apply.
- **changesThatWereOutOfRange:** If change numbers are out of range, for example, there is no change with that number, the header will include a list of change numbers that were out of range

Merging Multiple Modified Documents

The merge API provides the ability to merge changes made by multiple authors to a single version of document. Multiple modified documents are merged together into a single track-changed representation that can then be used for review or further evolution of the document.

The situation that can be solved with this API can be seen in the diagram below.



In order to use the API, the base document version and all edited versions to be merged must be posted as a single multipart/form-data request to the API endpoint. The result will be a document in .docx format that contains the changes made by the various authors as track changes.

The base document from which the edited versions were derived must be supplied to this API in docx or doc format. The edited versions must be supplied in docx, doc or RTF format. This API does not support operations on documents in PDF format or on documents that are password-protected.

The role of the posted documents in the merge process is determined by their order in the multipart/form-data request. The first document in the request is used as the base document version and all other documents are used as edited versions.

If the parts of the request containing the edited documents are given names then those names are used as the author names for the track changes in the merged document. For instance:

```
-----boundary
Content-Disposition: form-data; name="author name";
filename="edited1.docx"
Content-Type: application/vnd.openxmlformats-
officedocument.wordprocessingml.document

... contents of file goes here ...
-----boundary
```

if the author name is not specified in this way, then the author name will be extracted from the metadata of each edited document (the 'last author' metadata item will be used).

This API does not allow the output format or the rendering options to be configured by the end user.

Under some circumstances, the process may complete and give a merged result but it may have been impossible to represent all of the changes made by the editors in the merged document (perhaps because some of the changes made by one editor clashed with those made by another editor). This case can be detected by checking the response for the presence of a warning header.

```
Warning: 199 http://[your server name]/api/Merge/warnings/[long
alphanumeric id]
```

If the warning header is present then you should fetch the URL included in the warning header. The contents of this URL will describe in JSON format details of the changes that could not be represented in the merged document.

Note: *This API will function with a single baseline document and a single edited document, but in this case it will give results identical to those expected from performing a comparison of those two documents with the output format set as track changes.*

POST request example

In this example, the request merges three documents – baseline.docx, edited1.docx and edited2.docx – and outputs single document - merged-docx – showing all changes as track changes.

Example CURL command

```
curl -X POST --header 'Content-Type: multipart/form-data' --header
'Accept: application/vnd.openxmlformats-
officedocument.wordprocessingml.document' -F
"basedoc=@documents/baseline.docx" -F
"author1=@documents/edited1.docx" -F
"author2=@documents/edited2.docx" 'http://[your server
name]/api/Merge' -o Merged.docx
```

Explanation of the CURL command

Snippet	Function
<code>curl</code>	Beginning the request.
<code>-X POST</code>	Type of request.
<code>--header 'Content-Type: multipart/form-data'</code>	Format of the request. <div style="border: 1px solid orange; padding: 5px; margin-top: 10px;">Note: Must always be included for POST requests.</div>
<code>--header 'Accept: application/vnd.openxmlformats-officedocument.wordprocessingml.document'</code>	This part of the request is not required but if used, should match the output format application. These can be found on the Swagger page.
<code>-o Merged.docx</code>	Tells the request to save the response as the named file in the current directory.

Explanation of request parameters

In the example CURL command above, the request parameters and URL are as follows:

```
-F "basedoc=@documents/baseline.docx" -F
"author1=@documents/edited1.docx" -F
"author2=@documents/edited2.docx" 'http://[your server
name]/api/Merge'
```

An explanation of each element of the request URL is in the table below.

Name	Example value	Comments
Server name	http://[your server name]	
Endpoint	/api/Merge	

Name	Example value	Comments
Baseline document	basedoc=@documents/baseli ne.docx	The location of the baseline document on the local system.
First Edited Document	author1=@documents/edited 1.docx	The location of the first edited document on the local system.
Second Edited Document	author2=@documents/edited 2.docx	The location of the second edited document on the local system.
... optionally more edited documents		

Appendix A. Change Summary Information

This appendix describes the schema for the change summary produced with a comparison and the character set values.

RedlineJSON

RedlineJSON is a format for extracting change summary information. It contains the entire content of the Redline document in an easy-to-work-with JSON format.

Object Type	_type specification	Contains Content	May have IsInserted/ IsDeleted fields	Notes
Document	redlinedocument	Yes	No	Content will be an array of sections. Options contains the rendering options used
Section	section	Yes	Yes	'headersandfooters' field contains the headers and footers for the section
Paragraph	paragraph	Yes	Yes	The font listed is the most used font in the paragraph
Text	No _type field	No	No	The actual text is in the 'text' field. May contain a 'font' field if the font differs from the paragraph font.
List Number	listnumber	No	No	Contains 'text' and optional 'font' fields
Change	change	Yes	No	The 'type' and 'number' fields contain the change's type (insertion,deletion etc) and number respectively
Table	table	No	Yes	Has a 'rows' field that contains the list of rows in the table
Table Row	row	No	Yes	Cells field Contains a list of cells in the row
Table Cell	cell	Yes	No	Status field contains info on whether the cell is insert/deleted
Foot Note	footnote	Yes	Yes	
End Note	endnote	Yes	Yes	

Object Type	_type specification	Contains Content	May have IsInserted/ IsDeleted fields	Notes
Header	header	Yes	Yes	The type field of a header or footer can be 'main', 'first', 'even' or 'odd' (*check)
Footer	footer	Yes	Yes	
Bookmark	Bookmark	No	No	A bookmark object has fields 'name' and 'isstart'. When 'isstart' is false the object represents the end of the bookmark
Comment	comment	Yes	Yes	
Field	field	No	Yes	The 'instruction' and 'result' fields contain the details of the content of the document field
Field Instruction	fieldinst	Yes	No	
Field Result	fieldresults	Yes	No	This contains the content of the field that is actually displayed as part of the document
Shape	shape	Yes		This represents a drawing or text box in the document
Picture	pict	Yes		This represents an image in the document
Other Document Feature	blob	No		Any other document feature not handled above

```
{
  "_type": "redlinedocument",
  "options": {
    "FaultTolerantComparison": false,
    "IgnoreNumberedListStyle": false,
    "SingleColumnConversionFromPdf": false,
    "NoHeadersFootersConversionFromPdf": false,
    "IgnoreImagesWhenReadingSourceDocs": false,
    "IgnoreEmbeddedObjectsAndImagesWhenReadingSourceDocs":
false,
    "ShowStatistics": true,
    "ShowSummary": false,
    "InfoTablesAtStart": false,
    "ShowDeletionSummaryTable": false,
    "ShowRedlineOptionsSummaryTable": false,
    "ShowFileNamesInStatsTable": true,
    "CompareImages": false,
    "DetectFontChanges": false,
    "ShowChangeNumbers": false,
    "ChangeNumberBeforeChange": false,
    "DetectMoves": true,
    "CompressDeletionsToSingleCharacter": false,
    "InsertChangeMarkersInCells": true,
    "IgnoreChangesInTablesEntirely": false,
    "CompareTablesAtTheCellLevel": true,
    "ShowModifiedTableOnly": false,
    "ShowModifiedTableOnlyWithNoHighlighting": false,
    "DetectListNumberChanges": true,
    "UseWorkshareComparisonFooter": false,
    "CompareHeaderAndFooters": true,
    "CompareAtCharacterLevel": false,
    "CompareTextBoxContent": true,
    "DetectChangesToFieldCodes": true,
    "MakeFootersBlackAndWhite": false,
```

```
"MakeHeadersBlackAndWhite": false,  
"TurnSelectedFieldsToTextToAllowChangesInFieldResultToBeShown":  
false,  
  "DetectNoteNumberingChanges": true,  
  "DetectChangesToComments": false,  
  "DeletionReplacementCharacter": "",  
  "DetectWhitespaceChanges": true,  
  "DetectParagraphChanges": false,  
  "ShowDeletionsAdjacentToMovesAtTheMoveDestination": false,  
  "CompareCaseInsensitive": false,  
  "CompareNumbersAtCharacterLevel": false,  
  "CompareFootnotes": true,  
  "DetectParagraphStyleChanges": false,  
  "DetectCharacterStyleChanges": false,  
  "ParagraphTextMatchThresholdPercent": 10,  
  "CellInsertColor": "#CCCCFF",  
  "CellDeleteColor": "#FFCCCC",  
  "CellMoveColor": "#CCC0CC",  
  "CellMergeColor": "#FFFFCC",  
  "CellPaddingColor": "Silver",  
  "Insertion_TextDecoration": "DoubleUnderline",  
  "InsertionSurroundingStartChar": "",  
  "InsertionSurroundingEndChar": "",  
  "InsertionTextColor": "Blue",  
  "InsertionBackColor": "",  
  "Deletion_TextDecoration": "Strikethrough",  
  "Deletion_SurroundingStartChar": "",  
  "Deletion_SurroundingEndChar": "",  
  "Deletion_TextColor": "Red",  
  "Deletion_BackColor": "",  
  "MoveSource_TextDecoration": "Strikethrough",  
  "MoveSource_SurroundingStartChar": "",  
  "MoveSource_SurroundingEndChar": "",
```

```
"MoveSource_TextColor": "#00C000",
"MoveSource_BackColor": "",
"MoveDestination_TextDecoration": "DoubleUnderline",
"MoveDestination_SurroundingStartChar": "",
"MoveDestination_SurroundingEndChar": "",
"MoveDestination_TextColor": "#00C000",
"MoveDestination_BackColor": "",
"FormatChange_SurroundingStartChar": "",
"FormatChange_SurroundingEndChar": "",
"FormatChange_TextColor": "",
"FormatChange_BackColor": "",
"ChangeNumber_TextDecoration": "Superscript",
"ChangeNumber_TextColor": "Black",
"StyleLabel_TextColor": "Black",
"StyleChange_TextDecoration": "Outline",
"StyleChange_TextColor": "Black",
"StyleChange_BackColor": "White",
"StyleChangedSurroundingStartChar": "",
"StyleChangedSurroundingEndChar": "",
"RenderingSetName": "Unknown",
"OriginalDocId": null,
"ModifiedDocId": null,
"WriteDVBookmarks": true,
"DisableFieldFlattening": false,
"FlattenIfFields": false,
"ChangeBarsOnRight": false,
"ChangeBarsOnLeft": true,
"RemovePagesWithoutChanges": false
},
"content": [
  {
    "_type": "section",
    "headersandfooters": [
      {
```

```

    "_type": "footer",
    "type": "main",
    "content": [
      {
        "_type": "paragraph",
        "font": "Times New Roman",
        "content": [
          {
            "text": "\t"
          },
          {
            "_type": "field",
            "instruction": {
              "_type": "fieldinst",
              "content": [
                {
                  "text": " PAGE "
                }
              ]
            },
            "result": {
              "_type": "fieldresult",
              "content": [
                {
                  "_type":
                    "paragraph",
                  "content": [
                    {
                      "font":
                        "Times New Roman",
                      "text":
                        "16"
                    }
                  ]
                }
              ]
            }
          }
        ]
      }
    ]

```

```
    }
  ]
}
],
{
  "_type": "paragraph"
}
]
}
],
"content": [
  {
    "_type": "paragraph",
    "font": "Arial",
    "content": [
      {
        "text": "Dear "
      },
      {
        "_type": "change",
        "number": 1,
        "type": "deletion",
        "content": [
          {
            "text": "Sirs"
          }
        ]
      },
      {
        "_type": "change",
        "number": 2,
        "type": "insertion",
```

```
        "content": [
            {
                "text": "Sir/Madam"
            }
        ]
    },
    {
        "text": ","
    }
]
},
{
    "_type": "paragraph",
    "font": "Arial",
    "content": [
        {
            "_type": "listnumber",
            "text": "1.\t"
        },
        {
            "text": "This arrangement letter sets
out the scope and limitations of the work to be performed by us in
related with the above transaction, namely the proposed issue of [
] ("the issue") which will involve the preparation by the issuer,
and for which the issuer will be solely responsible, of an offering
circular [in accordance with the listing rules of the [relevant]
stock exchange]. This letter is written in the context of the
respective roles of the directors of the issuer, the head manager
("the lead manager"), the other managers (as defined in paragraph 2
below) and ourselves."
        }
    ]
},
{
    "_type": "paragraph"
},
```

```
{
  "_type": "table",
  "rows": [
    {
      "_type": "row",
      "cells": [
        {
          "_type": "cell",
          "content": [
            {
              "_type": "paragraph",
              "content": [
                {
                  "font":
                    "Arial",
                  "text": "JAN"
                }
              ]
            }
          ]
        },
        {
          "_type": "cell",
          "column": 1,
          "content": [
            {
              "_type": "paragraph",
              "content": [
                {
                  "font":
                    "Arial",
                  "text": "FEB"
                }
              ]
            }
          ]
        }
      ]
    }
  ]
}
```

```
        }
      ]
    }
  ]
},
{
  "_type": "row",
  "cells": [
    {
      "_type": "cell",
      "content": [
        {
          "_type": "paragraph",
          "content": [
            {
              "font":
"Arial",
              "text": "31"
            },
            {
              "_type":
"change",
              "number": 43,
              "type":
"insertion",
              "content": [
                {
                  "font":
"Arial",
                  "text":
"st"
                }
              ]
            }
          ]
        }
      ]
    }
  ]
}
```

```
    }
  ]
},
{
  "_type": "cell",
  "column": 1,
  "content": [
    {
      "_type": "paragraph",
      "content": [
        {
          "font":
            "Arial",
          "text": "28"
        }
      ]
    }
  ]
},
{
  "_type": "paragraph",
  "isdeleted": true
},
{
  "_type": "paragraph",
  "font": "Arial",
  "content": [
    {
      "text": "Applicable Law and
Jurisdiction"
```

```
    }
  ]
},
{
  "_type": "paragraph",
  "font": "Times New Roman",
  "content": [
    {
      "_type": "listnumber",
      "text": "29.\t"
    },
    {
      "font": "Arial",
      "text": "This arrangement letter shall
be governed by, and construed in accordance with, [specify country
(other than the us) whose laws govern the subscription agreement"
    },
    {
      "_type": "change",
      "number": 49,
      "type": "deletion",
      "content": [
        {
          "text": "7"
        }
      ]
    },
    {
      "_type": "change",
      "number": 50,
      "type": "insertion",
      "content": [
        {
          "text": "6"
```

```
    }
  ]
},
{
  "_type": "footnote",
  "content": [
    {
      "_type": "paragraph",
      "content": [
        {
          "_type": "change",
          "number": 51,
          "type": "deletion",
          "content": [
            {
              "font": "Times
New Roman",
              "text": "7"
            }
          ]
        },
        {
          "_type": "change",
          "number": 52,
          "type": "insertion",
          "content": [
            {
              "font": "Times
New Roman",
              "text": "6"
            }
          ]
        }
      ]
    },
    {
```


RedlineML

RedlineML is a 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:choice>
    </xs:sequence>
  </xs:group>
</xs:schema>
```

```
<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:choice>
  </xs:sequence>
</xs:group>
```

```
<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:restriction>
</xs:simpleType>
```

```
<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>
```

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

ChangeT values

0	DELETION
1	MOVESOURCE
2	MOVEDESTINATION
3	INSERTION
4	FORMAT_CHANGE
13	MOVEDDELETION
14	STYLECHANGE_TEXT
15	STYLECHANGE_LABEL

Code Samples

Extracting change summary information

Overview

When Workshare Compare Server compares documents, it generates a redline document and a change summary information document. The change summary information can be retrieved in either the RedlineML or the RedlineJSON format.

Once the change summary information has been retrieved you can extract information about the changes. The following code samples will extract the number of different types of changes such as:

- Total Changes Count
- Insertion Changes Count
- Deletion Changes Count
- Moved From Changes Count
- Moved To Changes Count
- Style Changes Count
- Format Changes Count

RedlineML

RedlineML is an XML format that contains the content of the redline document. You can use any language to access XML files but this code sample uses C# to access the change summary information in the RedlineML document.

Example (C#)

```
using System;
using System.Linq;
using System.Xml;

namespace ReadRedlineML
{
    class Program
    {
        private enum ChangeType
        {
            Deletion = 0,
            MoveSource = 1,
            MoveDestination = 2,
            Insertion = 3,
            FormatChange = 4,
            MovedDeletion = 13,
            StyleChangeText = 14,
            StyleChangeLabel = 15
        }

        static void Main(string[] args)
        {
            // Pass in the redline
            // Without any error checking
            string xml = System.IO.File.ReadAllText(args[0]);

            XmlDocument doc = new XmlDocument();
            doc.LoadXml(xml);
            XmlNodeList nodes = doc.GetElementsByTagName("change");

            var values = nodes.Cast<XmlNode>().Select(s => new
            {
                Name = s.Name,
```

```
        Number =
Convert.ToInt32(s.Attributes?["number"].Value),
        ChangeType =
Convert.ToInt32(s.Attributes?["type"].Value)
    }).ToList();

    int totalChangesCount = values.Count();
    int insertionChangesCount = values.Count(w =>
w.ChangeType == (int)ChangeType.Insertion);
    int deletionChangesCount = values.Count(w =>
w.ChangeType == (int)ChangeType.Deletion);
    int movedFromChangesCount = values.Count(w =>
w.ChangeType == (int)ChangeType.MoveSource);
    int movedToChangesCount = values.Count(w =>
w.ChangeType == (int)ChangeType.MoveDestination);
    int formatChangesCount = values.Count(w => w.ChangeType
== (int)ChangeType.FormatChange);
    int styleChangesCount = values.Count(w => w.ChangeType
== (int)ChangeType.StyleChangeLabel || w.ChangeType ==
(int)ChangeType.StyleChangeText);

    Console.WriteLine($"Number of total changes:
{totalChangesCount}");
    Console.WriteLine($"Number of inserted changes:
{insertionChangesCount}");
    Console.WriteLine($"Number of deleted changes:
{deletionChangesCount}");
    Console.WriteLine($"Number of moved from changes:
{movedFromChangesCount}");
    Console.WriteLine($"Number of moved to changes:
{movedToChangesCount}");
    Console.WriteLine($"Number of format changes:
{formatChangesCount}");
    Console.WriteLine($"Number of style changes:
{styleChangesCount}");
    }
}
}
```

RedlineJSON

The RedlineJSON is a JSON format which contains the content of the redline document. You can use any language to access JSON files but this code sample uses JavaScript to access the change summary information in the RedlineJSON document.

Example (JavaScript)

```
<!DOCTYPE html>
<html>
<head>
  <script>
    window.onload = function () {

      var changeList = [];

document.getElementById('file').addEventListener('change',
onChange);

      function onChange(event) {
        var reader = new FileReader();
        reader.onload = onReaderLoad;
        reader.readAsText(event.target.files[0]);
      }

      function onReaderLoad(event) {
        var obj = JSON.parse(event.target.result);
        getChangesList(obj);
        extractSummary(obj);
      }

      function getChangesList(o) {
        for (var key in o) {
          if (o[key] !== null && typeof (o[key]) ==
"object") {
              if (o[key].hasOwnProperty('_type')
&& o[key]["_type"] == "change") {
                  changeList.push(o[key]);
                }
            }
          }
        }
      }
    }
  </script>
</head>
</html>
```

```
        }
        getChangesList(o[key]);
    }
}

function extractSummary(obj) {

    var totalChangesCount = changeList.length;
    var insersionChangesCount =
changeList.filter(function (c) {
        return c.type == "insertion"}).length;
(c) {
    var deletionChangesCount = changeList.filter(function
        return c.type == "deletion"}).length;
(c) {
    var movedFromChangesCount =
changeList.filter(function (c) {
        return c.type == "movesource"}).length;
(c) {
    var movedToChangesCount = changeList.filter(function
        return c.type == "movedestination"}).length;
(c) {
    var formatChangesCount = changeList.filter(function
        return c.type == "format_change"}).length;
(c) {
    var styleChangesCount = changeList.filter(function
        return c.type == "stylechange_label"
|| c.type == "stylechange_text"}).length;

    var summary = 'Number of total changes: ' +
totalChangesCount + '\n' +
```

```
        'Number of inserted changes: ' +
insertionChangesCount + '\n' +
        'Number of deleted changes: ' +
deletionChangesCount + '\n' +
        'Number of moved from changes: ' +
movedFromChangesCount + '\n' +
        'Number of moved to changes: ' +
movedToChangesCount + '\n' +
        'Number of format changes: ' +
formatChangesCount + '\n' +
        'Number of style changes: ' + styleChangesCount
+ '\n';

        alert(summary);
    }
};
</script>
</head>
<body>
    <input id="file" type="file" />
    <p>Select a RedLine JSON file</p>
</body>
</html>
```

 Workshare Ltd.

© 2019. Workshare Ltd. All rights reserved.

Copyright

Workshare Professional and Workshare DeltaView are registered trademarks of Workshare Ltd. Workshare Compare, Workshare Protect, Workshare 3, Workshare DeltaServer, SafetyGain, and the Workshare logo are trademarks of Workshare Ltd. All other trademarks are those of their respective holders.

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.

Disclaimer

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.

Revisions

- Published for Workshare Compare Server 9.5: 14/9/18
- Revised for Workshare Compare Server 9.5.1: 8/1/18
- Revised for Workshare Compare Server 9.5.2: 17/5/18
- Revised for Workshare Compare Server 9.5.3: 6/12/18
- Revised for Workshare Compare Server 9.5.4: 18/2/19
- Revised for Workshare Compare Server 9.5.5: 29/3/19
- Revised for Workshare Compare Server 9.5.6: 22/5/19
- Revised for Workshare Compare Server 9.5.7: 28/6/19
- Revised for Workshare Compare Server 9.6: 20/8/19
- Revised for Workshare Compare Server 9.7: 25/10/19

Workshare Ltd., 20 Fashion Street, London E1 6PX www.workshare.com