



# **PDF Converter Services - User & Developer Guide**

Muhimbi Ltd

**Version 7.3**



## Document Control

Draft	Author	Date	Comment
3.0	Muhimbi	13/11/2009	Revised for version 3.0
3.1	Muhimbi	19/01/2010	Updated for version 3.1
3.2	Muhimbi	23/02/2010	Updated for version 3.2
3.4	Muhimbi	12/04/2010	Revised for PDF Converter Services
3.5	Muhimbi	03/06/2010	Updated for version 3.5
4.0	Muhimbi	10/09/2010	Updated for version 4.0
4.1	Muhimbi	03/01/2011	Updated for version 4.1
5.0	Muhimbi	26/04/2011	Updated for version 5.0
5.1	Muhimbi	06/09/2011	Updated for version 5.1
5.2	Muhimbi	12/01/2012	Updated for version 5.2
6.0	Muhimbi	08/06/2012	Updated for version 6.0
6.1	Muhimbi	02/10/2012	Updated for version 6.1
7.0	Muhimbi	28/03/2013	Updated for version 7.0
7.1	Muhimbi	24/10/2013	Updated for version 7.1
7.2	Muhimbi	04/04/2013	Updated for version 7.2
7.2.1	Muhimbi	04/07/2013	Updated for version 7.2.1
7.3	Muhimbi	26/02/2015	Updated for version 7.3

## Purpose and audience of document

This document explains how to access the *Muhimbi PDF Converter Services* (MDCS) using its Web Services interface.

The intended audience is any developer that wishes to convert documents or web pages to PDF format from their own code.

## Disclaimer

© Muhimbi. All rights reserved. No part of this document may be altered, reproduced or distributed in any form without the expressed written permission of Muhimbi.

This document was created strictly for information purposes. No guarantee, contractual specification or condition shall be derived from this document unless agreed to in writing. Muhimbi reserves the right to make changes in the products and services described in this document at any time without notice and this document does not represent a commitment on the part of Muhimbi in the future.

While Muhimbi uses reasonable efforts to ensure that the information and materials contained in this document are current and accurate, Muhimbi makes no representations or warranties as to the accuracy, reliability or completeness of the information, text, graphics, or other items contained in the document. Muhimbi expressly disclaims liability for any errors or omissions in the materials contained in the document and would welcome feedback as to any possible errors or inaccuracies contained herein.

Muhimbi shall not be liable for any special, indirect, incidental, or consequential damages, including without limitation, lost revenues or lost profits, which may result from the use of these materials. All offers are non-binding and without obligation unless agreed to in writing.

## Contents

1	Introduction	6
2	Features and functionality	7
2.1	Supported document formats	8
3	Web Services interface / Object Model	9
3.1	Overview	9
3.2	Conversion	10
3.2.1	The Convert Method	10
3.2.2	The OpenOptions class	11
3.2.3	The ConversionSettings class	11
3.2.4	The ConverterSpecificSettings_InfoPath class	12
3.2.5	The ConverterSpecificSettings_WordProcessing class	13
3.2.6	The ConverterSpecificSettings_HTML class	13
3.2.7	The ConverterSpecificSettings_Cad class	14
3.2.8	The ConverterSpecificSettings_Presentations class	16
3.2.9	The ConverterSpecificSettings_MSG class	17
3.2.10	The ConverterSpecificSettings_Spreadsheets class	18
3.2.11	The ConverterSpecificSettings_CommandLineConverter class	18
3.2.12	The ConverterSpecificSettings_PDF class	19
3.2.13	The OutputFormatSpecificSettings_PDF class	19
3.2.14	The SystemSettings class	20
3.2.15	The TaskMonitorSettings class	20
3.3	Working with ProcessBatch (Merging / Splitting files)	21
3.3.1	Merging files	21
3.3.2	Splitting files	22
3.3.3	The ProcessingOptions class	24
3.3.4	The MergeSettings class	24
3.3.5	The FileSplitOptions class	25
3.3.6	The SourceFile class	25
3.3.7	The FileMergeSettings class	25
3.3.8	The BatchResults class	26
3.3.9	The BatchResult class	26
3.4	OCR (Optical Character Recognition)	27
3.4.1	The OCRSettings class	27
3.4.2	The OCRRegion class	28
3.4.3	The OCRResult class	29
3.4.4	The RegionText class	29
3.5	Watermarking	30
3.5.1	The Watermark class	30
3.5.2	The Element class	31
3.5.3	Individual Element Types	32
3.5.4	The Defaults class	33
3.5.5	Embedding field codes in the Text element	35
3.6	Table Of Contents	36
3.6.1	The TocSettings class	36
3.6.2	The NameValuePair class	37
3.7	Configuration and Diagnostics	38
3.7.1	Retrieving Configuration settings	38
3.7.2	Running Diagnostic tests	39
3.8	Exception handling	40
4	Programmatically processing documents	41
4.1	PDF Conversion in .NET	41
4.2	PDF Conversion in Java (WSImport)	44

4.3	PDF Conversion in Java (Axis2)	47
4.4	PDF Conversion in Ruby / Rails	51
4.5	PDF Conversion in PHP	55
4.6	Cross-Converting between document types	59
4.6.1	Cross-Converting file types using a Web Service call	60
4.6.2	Convert InfoPath to MS-Word, Excel, XPS and PDF	60
4.7	Merging multiple files into a single PDF using .NET	64
4.8	Merging multiple files into a single PDF using Java	68
4.9	Splitting PDFs into multiple documents	71
4.10	Converting HTML / web pages using a Web Service call	74
4.10.1	Inserting Page Breaks when converting HTML to PDF	75
4.11	Converting PDF to PDF/A1b or PDF/A2b	76
4.12	Controlling which InfoPath views to Export to PDF	80
4.12.1	Use a special view for exporting to PDF	80
4.12.2	Determine at runtime which views to export	81
4.12.3	View prioritisation rules	81
5	Working with watermarks	82
5.1	Watermarking in .NET	82
5.2	Watermarking in Java	87
6	Carry out OCR (Optical Character Recognition)	90
6.1	OCR files using .NET	91
6.2	OCR files using Java	95
7	Post processing PDF Files	98
7.1	Specifying PDF Viewer Preferences	98
7.2	Set PDF Version, Enable Fast Web Views and control Font embedding	99
8	Building a Table Of Contents	101
8.1	Object Model	101
8.2	XML Source Data	103
8.3	XSL Transformation	104
8.4	Testing & Troubleshooting	107
9	Troubleshooting	108
9.1	Problems parsing the WSDL	108
9.2	Converting documents takes a long time	108
9.3	The PDF file does not look the same as the source file	108
9.4	An evaluation message is displayed in each converted document	108
9.5	InfoPath Forms fail to convert	109
9.6	Converting non supported files	109
	Appendix - Relevant articles on the Muhimbi Blog	110
	Appendix - Licensing	112
	Appendix – Class Diagrams	114

### 1 Introduction

This document explains how to access the *Muhimbi PDF Converter Services* (MDCS) using its Web Services interface. The intended audience is any developers that wish to convert documents or web pages to PDF format, OCR Images, Apply watermarks or PDF Security from their own code.

It is assumed that the audience has some familiarity with programming against Web Services based interfaces.

For more details about this product please see:

1. Product Information:  
<http://www.muhimbi.com/Products/PDF-Converter-Services.aspx>
2. Product Overview:  
<http://support.muhimbi.com/entries/21251792-Where-can-I-find-an-overview-of-the-capabilities-of-the-PDF-Converter->
3. Knowledge Base / Frequently Asked Questions:  
<http://support.muhimbi.com/>
4. Release Notes:  
<http://www.muhimbi.com/support/documentation/PDF-Converter-Services/Release-Notes.aspx>
5. Installation & Administration Guide:  
<http://www.muhimbi.com/support/documentation/PDF-Converter-Services/Administration-Guide.aspx>
6. PDF Converter related content on the Muhimbi Blog:  
<http://blog.muhimbi.com/search/label/PDF%20Converter%20Services>

To keep on top of the latest news and releases, please subscribe to our blog or twitter feed at <http://www.muhimbi.com/contact.aspx>.

## 2 Features and functionality

The MDCS is a highly scalable and robust server side framework for converting typical office documents to PDF format using a Web Services based interface.

The key features are:

- Convert popular document types including MS-Office, AutoCAD, HTML, MSG (email) and images to PDF or XPS format with perfect fidelity.
- Cross-convert between formats including XLS to XLSX, DOCX to DOC, XLS to DOC, InfoPath to DOC and XLS and many more.
- Apply Optical Character Recognition (OCR) to convert images and scans into fully searchable and indexable documents.
- Scalable architecture that allows multiple conversions to run in parallel. The service can be *scaled up* by adding additional CPUs and *scaled out* by using standard HTTP Load Balancers.
- Runs as a Windows Service. No need to install or configure IIS or other web service frameworks.
- Convert password protected documents.
- Apply security settings to generated PDF files including encryption, password protection and multiple levels of PDF Security options to prevent users from printing documents or copy a document's content.
- Flexible watermarking system allowing different watermarks for individual pages (odd, even, portrait, landscape, specific page numbers etc)
- Merge multiple documents into a single PDF file or split up a PDF file into multiple documents.
- Generate regular PDF files or files in PDF/A format.
- Strip or embed fonts.
- Set PDF Viewer Preferences.
- Linearize PDF files (a.k.a Fast Web View).
- Generate high resolution PDF Files optimised for printing or normal resolution files optimised for use on screen.
- Dynamically refresh a document's content before generating the PDF. Ideal for merging content from external sources into your PDF file.
- Control how to convert hidden / selected content such as PowerPoint Slides, InfoPath views and Excel worksheets.
- Add custom converters using a simple plug-in architecture.

In addition to the features described above, the MDCS software stack also contains a layer of functionality to control concurrency, request queuing and watchdog services to deal with unresponsive and runaway processes.

The MDCS is built on top of the WCF Framework. Full details about WCF and how it can be configured / tuned can be found on the following page:  
<http://msdn.microsoft.com/en-us/library/ms731925.aspx>

## 2.1 Supported document formats

The MDCS supports the most common file formats including MS-Word, Excel, PowerPoint, InfoPath, MSG, EML (email), Visio and Microsoft Publisher. Legacy file formats starting with Office 95 are supported as well as the latest formats used by Office 2013. Non MS-Office related file types such as HTML, AutoCAD and common image formats are supported as well.

	Supported	Not Supported
MS-Word	doc, docx, docm, rtf, txt, xml, odt, ott, wps, wpd	dotm, dotx, dot
Excel	xls,xlsx, xlsm, xlsb, xml, csv, dif, ods, ots	xltx, xltm, xlt, txt (tab delimited), prn, slk, xlam, xla
PowerPoint	ppt, pptx, pptm, xml, odp, otp, pps, ppsx, ppsm	potx, potm, pot, thmx, ppam, ppa
InfoPath	xml, infopathxml	
Publisher	pub	
Email	eml, msg	
Visio & Vector formats	vsd,vdx,vdw,svg,svgz	
HTML & Web pages	html, htm, mht and any url that returns HTML such as .aspx or .jsp.	
Image formats	gif, png, jpg, bmp, tif, tiff	
AutoCAD formats <sup>1</sup>	dwg, dxf	

As of version 6.0 the PDF Converter also supports output in non-PDF file formats. For details see section 4.6 *Cross-Converting between document types*.

<sup>1</sup> The AutoCAD converter has several automatic recolouring options. For details see *AutoCAD specific switches* in the *Administration Guide*, subsection *Tuning the Document Conversion Service*.

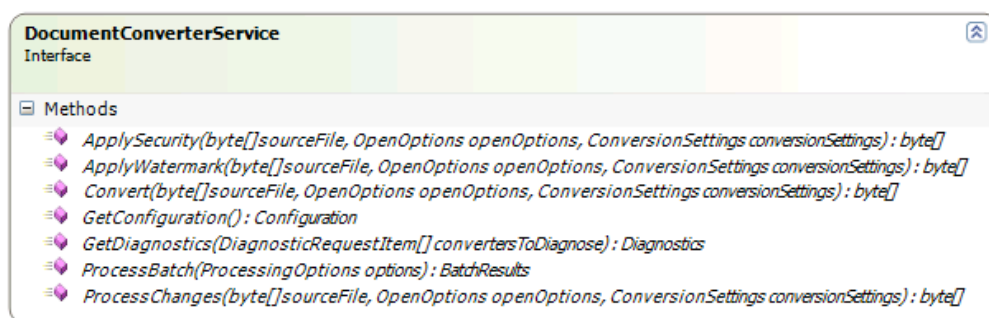


## 3 Web Services interface / Object Model

Although the Object Model exposed by the web service is easy to understand, the system provides very powerful functionality, including watermarking, security, PDF Merging and fine grained control over how PDF files are generated.

### 3.1 Overview

The web service contains the following methods:



- **Convert:** Convert the file in the *sourceFile* byte array using the specified *openOptions* and *conversionSettings*. The generated PDF or XPS file is returned as a byte array as well.
- **GetConfiguration:** Retrieve information about which converters are supported and the associated file extensions. Consider calling this service once to retrieve a list of valid file extensions, and check if a file is supported before it is submit to the web service. This will prevent a lot of redundant traffic resulting in increased scalability.
- **GetDiagnostics:** Run a diagnostics test that carries out an internal end-to-end test for each specified converter type. Call this method to check if the service and all prerequisites have been deployed correctly.
- **ProcessBatch:** Process multiple files in one call. Currently limited to merge and split operations.

The *ApplySecurity*, *ApplyWatermark* and *ProcessChanges* methods are identical at this moment in time and are provided for convenience only. They all take exactly the same parameters as the *Convert* method, but they can act on PDF files only and basically apply whatever combination of Watermarks, Security Settings and other information is provided.

The full object model is discussed below, larger versions of the diagrams can be found at the end of this document.

The WSDL can be found at the following location. Change *localhost* to the actual host name if the MDCS is located on a different machine.

<http://localhost:41734/Muhimbi.DocumentConverter.WebService/?wsdl>

This section describes these conversion related classes and methods in detail, the various enumerations are self describing. For code examples see chapters 4 - *Programmatically processing documents* and 5 - *Working with watermarks*.

The method returns a `byte[]` containing the generated file. Errors are raised as instances of the type *WebServiceFaultException*.

### 3.2.2 The OpenOptions class

An instance of this class is passed by the *Convert* method to provide details for opening the file, such as *Macro Security settings* and *security credentials*.

Property	Type	Description
AllowExternalConnections	Bool	Allow documents to connect to external data sources. Currently only supported by Excel.
AllowMacros	MacroSecurityOption	Specify what type of embedded macros to allow, if any.
FileExtension	String	Extension of the source file, indicating the document type.
OriginalFileName	String	File name of the original file for debugging and logging purposes.
Password	String	Optional password for protected documents.
RefreshContent	Bool	Refresh the content of the document after loading (apply MS-Word properties, recalculate content).
SystemSettings	System Settings	Optional System settings for the current request..
UserName	String	Optional user name for documents that require both user name and password (e.g. certain web pages)

### 3.2.3 The ConversionSettings class

An instance of this class is passed by the *Convert* method to provide settings to apply when converting the file, e.g. *watermarks*, *outputformat*, *security settings*, etc.

Property	Type	Description
ConverterSpecificSettings	Converter Specific Settings	An instance of an object that contains settings specific to the source document, e.g. how many PowerPoint slides to include in a page or how to include MS-Word revisions. See 3.2.4 - 0 for details.
EndPage	Int	The first page to render. Leave blank or specify -1 to ignore this value.
Fidelity	Conversion Fidelities	The type of converter to use. Usually <i>Full</i> , but in case of custom converters you may need to use <i>High</i> . For details see the Administration Guide.
Format	Output Format	Format to convert the file to. See 4.6 <i>Cross-Converting between document types</i> .
GenerateBookmarks	Bookmark Generation Option	Generate TOC based on bookmarks. Note that this functionality is not available for all document types.
OCRSettings	OCR Settings	Optional settings for when the source file is Image based and OCR must be carried out.

Property	Type	Description
OpenPassword	String	Optionally specify the password to secure the generated document with to prevent users without a valid password to open the file.
OwnerPassword	String	Optionally specify the password to secure the generated document with to prevent users without a valid password to access certain features
OutputFormatSpecificSettings	Output Format Specific Settings	An instance of an object that contains settings specific to the output format, e.g. <i>ViewerPreferences</i> or <i>Font Embedding</i> settings when the output format is PDF (See section 0 for details).
PageOrientation	PageOrientation	The orientation of the pages in the PDF file for converters that support this option, e.g. the HTML Converter.
PDFProfile	PDF Profile	The PDF Profile to use for rendering the document, e.g. PDF/A or PDF 1.5
Quality	Conversion Quality	Specify the required quality of the destination document.
Range	Conversion Range	For supported file types (Excel, PowerPoint etc) specify which parts of the file to render.
SecurityOptions	Security Options	Optionally specify one or more security options for the generated document.
StartPage	Int	The first page to render. Leave blank or specify -1 to ignore this value.
Watermarks	Watermark[]	Optional array of watermarks to apply to the generated PDF file. For details see <b>Error! eference source not found..</b>

### 3.2.4 The ConverterSpecificSettings\_InfoPath class

An instance of this class is optionally passed in the *ConverterSpecificSettings* property of the *ConversionSettings* class for source documents that represent InfoPath forms. When this information is not provided then the default settings for the various properties will be taken from the service's config file.

Property	Type	Description
AutoTrustForms	Bool	Automatically trust InfoPath 2010 forms. For details see <i>Appendix - Using InfoPath with External Data Sources</i> in the Administration Guide.
ConversionViews	InfoPath View[]	List of view names to convert. See 4.12 for details.
ConvertAttachments	Bool	Enable the conversion of attachments.
StripDataObjects	Bool	To allow forms to be converted without extensive server configuration, remove all

Property	Type	Description
		external data connections. It is recommended to always set this to 'true' unless you have a real good reason not to.
StripDotNETCode	Bool	To allow full trust forms to be converted without extensive server configuration, strip all custom .net code from the form. It is recommended to always set this to 'true' unless you have a real good reason not to.
ProcessFullTrustForms	Bool	Should InfoPath forms marked as requiring Full Trust be processed based on the other parameters (e.g. StripDotNETCode) or not?

### 3.2.5 The ConverterSpecificSettings\_WordProcessing class

An instance of this class is optionally passed in the *ConverterSpecificSettings* property of the *ConversionSettings* class for source documents that represent Word Processing documents such as MS-Word files. When this information is not provided then the default settings for the various properties will be taken from the service's config file.

Property	Type	Description
ProcessDocumentTemplate	Bool	Specify if the MS-Word template will need to be stripped out for DOCX files. Specify <i>true</i> unless you are experiencing formatting problems.
RevisionsAndComments MarkupMode	Revisions And Comments MarkupMode	Choose how to show revisions to the document. You can show revisions as balloons in the margins of the document or show them directly within the document itself.
RevisionsAndComments DisplayMode	Revisions And Comments DisplayMode	Choose how to view the proposed changes to the document.

### 3.2.6 The ConverterSpecificSettings\_HTML class

An instance of this class is optionally passed in the *ConverterSpecificSettings* property of the *ConversionSettings* class for HTML based source documents. When this information is not provided then the default settings for the various properties will be taken from the service's config file.

Property	Type	Description
PageMargins	String	The Margin / border around the generated PDF file. One or four {value}{dim} components separated by commas (,) where <ul style="list-style-type: none"> <li>{value} is a numerical value</li> <li>{dim} is the dimension which can be 'mm', 'in.' or 'inches'. (Defaults to inches)</li> </ul>

Property	Type	Description
		<p>when nothing is specified)</p> <p>When multiple values are specified then the sequence is: left, top, right and bottom.</p> <p>Example: "12mm, 24mm, 12mm, 24mm"</p>
PaperSize	String	<p>Specify the paper size to use for the PDF when converting HTML pages. Either:</p> <ul style="list-style-type: none"> <li>A 'Named' paper size such as 'A4' or 'Letter' (See <a href="#">MSDN</a>)</li> <li>or a custom size in "{width}{dim}{sep}{height}{dim}" format where <ul style="list-style-type: none"> <li>- {width} and {height} are numerical values (decimal separator must be colon '.')</li> <li>- {dim} is the dimension which can be 'mm', 'in.' or 'inches'. (Defaults to inches when nothing is specified)</li> <li>- {sep} separates the width and the height, either 'by', comma (,) or the letter 'x'</li> </ul> </li> </ul> <p>Example: "8.5 in. by 6 in."</p>
ScaleMode	HTMLScale Mode	<p>Determine how the HTML will be scaled to the PDF page size:</p> <ul style="list-style-type: none"> <li><i>FitWidth</i> - HTML is scaled to fit the width of the paper.</li> <li><i>NoScale</i> - HTML is not scaled, may result in truncating.</li> </ul>
ConversionDelay	int	Optional delay (in milliseconds) between loading the web page and converting to PDF. This allows asynchronous events such as JavaScript to complete in DHTML heavy web pages.
ClearBrowserCache	bool	Clear the browser's cache before carrying out the conversion.
SplitImages	Bool	Split images across page breaks or wrap the complete image to the next page.

### 3.2.7 The ConverterSpecificSettings\_Cad class

An instance of this class is optionally passed in the *ConverterSpecificSettings* property of the *ConversionSettings* class for CAD (dxf, wdg) based documents. When this information is not provided then the default settings for the various properties will be taken from the service's config file.

Property	Type	Description
BackgroundColor	String	<p>Specify the background color. Accepted values:</p> <ul style="list-style-type: none"> <li><i>Default</i> - The default black color is used.</li> </ul>

Property	Type	Description
		<ul style="list-style-type: none"> <li>Named color, e.g. 'White' as defined on <a href="#">MSDN</a>.</li> <li>Web color using the "#aarrggbb" or "#rrggbb" format.</li> </ul>
CadConversionElements	CadConversionElement []	Array of named views, layouts, 3D views to convert to PDF.
EmptyLayoutDetectionMode	CadEmptyLayoutDetectionMode	<p>Specifies how the conversion handles empty or nearly empty layouts. Accepted values are :</p> <ul style="list-style-type: none"> <li><i>SkipNone</i> - Every layout will be drawn regardless if it has anything in it or not</li> <li><i>SkipEmptyLayouts</i> - Layouts will be drawn only if they have entities or valid viewports attached to it</li> <li><i>SkipLayoutsWithoutViewports</i> - Only layouts with valid viewports are drawn</li> </ul>
ForegroundColor	String	<ul style="list-style-type: none"> <li><i>Default</i>: Objects are drawn in their original color.</li> <li><i>CorrectForBackground</i> - Objects are drawn in their own color, but colors matching the background color will be inverted to ensure visibility.</li> <li>Named color, e.g. 'White' as defined on <a href="#">MSDN</a>.</li> <li>Web color using the "#aarrggbb" or "#rrggbb" format.</li> <li><i>Greyscale</i> - All colors are converted to a shades of grey based on luminosity.</li> <li><i>GreyscaleDarken</i> - All colors are converted to a shade of grey then darkened</li> <li><i>GreyscaleLighten</i> - All colors are converted to a shade of grey then lightened</li> <li><i>Darken</i> - All colors are darkened</li> <li><i>Lighten</i> - All colors are lightened</li> </ul>
PageMargins	String	<p>The Margin / border around the generated PDF file. One or four {value}{dim} components separated by commas (,) where</p> <ul style="list-style-type: none"> <li>{value} is a numerical value</li> <li>{dim} is the dimension which can be 'mm', 'in.' or 'inches'. (Defaults to inches when nothing is specified)</li> </ul> <p>When multiple values are specified then the sequence is: left, top, right and bottom. Example: "12mm, 24mm, 12mm, 24mm"</p>



Property	Type	Description
PaperSize	String	<p>Specify the paper size to use for the PDF when converting HTML pages. Either:</p> <ul style="list-style-type: none"> <li>A 'Named' paper size such as 'A4' or 'Letter' (See <a href="#">MSDN</a>)</li> <li>or a custom size in "{width}{dim}{sep}{height}{dim}" format where <ul style="list-style-type: none"> <li>{width} and {height} are numerical values (decimal separator must be colon '.')</li> <li>{dim} is the dimension which can be 'mm', 'in.' or 'inches'. (Defaults to inches when nothing is specified)</li> <li>{sep} separates the width and the height, either 'by', comma (,) or the letter 'x'</li> </ul> </li> </ul> <p>Example: "8.5 in. by 6 in."</p>
LayoutSortOrder	CadLayout SortOrder	<p>Specify the sort order for layout names. Accepted values are:</p> <ul style="list-style-type: none"> <li><i>Default</i> - Use the order in which the layouts are stored in the source file.</li> <li><i>Ascending</i> - Sort the layout names from A-Z.</li> <li><i>Descending</i> - Sort the layout names from Z-A.</li> </ul>

### 3.2.8 The ConverterSpecificSettings\_Presentations class

An instance of this class is optionally passed in the *ConverterSpecificSettings* property of the *ConversionSettings* class for source documents that represent Presentations such as PowerPoint files. When this information is not provided then the default settings for the various properties will be taken from the service's config file.

Property	Type	Description
FrameSlides	Bool	Include a frame / border around the slides.
PrintOutputType	Presentations PrintOutput Type	Specify the part of the presentation to print. You can print the slides, handouts, speaker notes or the outline.



### 3.2.9 The ConverterSpecificSettings\_MSG class

An instance of this class is optionally passed in the *ConverterSpecificSettings* property of the *ConversionSettings* class for source documents that represent MSG (email) files. When this information is not provided then the default settings for the various properties will be taken from the service's config file.

Property	Type	Description
BestBodyMode	MSGBestBodyMode	Determine which email body content (Text / HTML / RTF / RTFHTML) to extract when processing MSG files.
BreakOnUnsupportedAttachment	Bool	When an unsupported attachment is found, e.g. a file type not supported by the conversion service, the conversion is halted and an error message is returned.
BreakOnUnsupportedEmbeddedObject	Bool	When an unsupported embedded object is found, e.g. an embedded OLE object where no file type identification is provided, the conversion is halted and an error message is returned.
ConvertAttachments	Bool	Enable the conversion of attachments.
DisplayAttachmentSummary	Bool	Specify whether the attachment filenames are displayed in the email header. This setting works independently to ConvertAttachments.
EmailAddressDisplayMode	MSGEmail Address Display Mode	Determine how the <i>To</i> , <i>Cc</i> and <i>Bcc</i> email addresses are displayed when processing emails.
FromEmailAddressDisplayMode	MSGEmail Address Display Mode	Determine how <i>From</i> email addresses are displayed when processing emails.
EmbeddedObjectDisplayMode	MSGEmbedded ObjectDisplay Mode	Determines how embedded objects are displayed. NOTE: Where the embedded object is displayed as an icon, use EmbeddedObjectIconDisplayMode.
EmbeddedObjectIconDisplayMode	MSGEmbedded ObjectIcon DisplayMode	Determines how embedded objects are displayed where they are stored as an icon.
EmbeddedObjectScalePercentage	Decimal	The percentage by which embedded objects are scaled prior to rendering. It defaults to 3.33(%).
HTMLScaleMode	HTMLScale Mode	Scale mode for HTML MSG files. Either <i>FitWidth</i> , <i>FitWidthScaleImagesOnly</i> or <i>NoScale</i> . Unless there is a good reason to change this, use <i>FitWidthScaleImagesOnly</i> .
PaperSize	String	Specify the paper size to use for the PDF when converting HTML based email. Either:

Property	Type	Description
		<ul style="list-style-type: none"> <li>A 'Named' paper size such as 'A4' or 'Letter' (See <a href="#">MSDN</a>)</li> <li>or a custom size in "{width}{dim}{sep}{height}{dim}" format where               <ul style="list-style-type: none"> <li>{width} and {height} are numerical values (decimal separator must be colon '.')</li> <li>{dim} is the dimension which can be 'mm', 'in.' or 'inches'. (Defaults to inches when nothing is specified)</li> <li>{sep} separates the width and the height, either 'by', comma (,) or the letter 'x'</li> </ul> </li> </ul> <p>Example: "8.5 in. by 6 in."</p>
PlainTextLineBreaks	MSGPlain TextLine Breaks	Determine how return characters (new lines) in plain text MSG bodies are handled. One of <i>RetainAll</i> , <i>RemoveExtra</i> or <i>Legacy</i> .

### 3.2.10 The ConverterSpecificSettings\_Spreadsheets class

An instance of this class is optionally passed in the *ConverterSpecificSettings* property of the *ConversionSettings* class for source documents that represent Spreadsheets (Excel) files. When this information is not provided then the default settings for the various properties will be taken from the service's config file.

Property	Type	Description
FitToPagesTall	int	Sets the number of pages tall the worksheet will be scaled to when it's converted.
FitToPagesWide	Int	Sets the number of pages wide the worksheet will be scaled to when it's converted.

### 3.2.11 The ConverterSpecificSettings\_CommandLineConverter class

An instance of this class is optionally passed in the *ConverterSpecificSettings* property of the *ConversionSettings* class for file types that have been configured to use the Command Line Converter (See Admin Guide, *Appendix - Invoke 3rd party Converters*)

Property	Type	Description
Parameter1 – 10	String	Content for any command line arguments passed to the external executable using the {ParameterX} syntax.

### 3.2.12 The ConverterSpecificSettings\_PDF class

An instance of this class is optionally passed in the *ConverterSpecificSettings* property of the *ConversionSettings* class for conversions where the input file is of type PDF. Please note the difference between this class and the *OutputFormatSpecificSettings\_PDF* class described in 3.2.13.

Property	Type	Description
ConvertAttachments	bool	Convert, and Merge, files attached to PDF files. For details see <a href="http://goo.gl/49Lkn8">http://goo.gl/49Lkn8</a>
ConvertAttachmentMode	PDFConvert Attachment Mode	<p><b>RemoveAll:</b> When a PDF file is processed, all attachments will be converted and merged to the main PDF. All attachments will be removed from the PDF, including those of attachments for which the file type is not recognised by the converter.</p> <p><b>RemoveSupported:</b> When a PDF file is processed, all attachments will be converted and merged to the main PDF, but only those attachments that are supported by the converter are removed from the PDF, all other attachments remain present in the main file.</p>

### 3.2.13 The OutputFormatSpecificSettings\_PDF class

An instance of this class is optionally passed in the *OutputFormatSpecificSettings* property of the *ConversionSettings* class for operations where the output format is PDF. For further details see chapter 7 *Post processing PDF Files*.

Property	Type	Description
FastWebView	bool	Enable Fast Web View / Linearization to optimize the PDF for output on the web. (Requires a Muhimbi PDF Converter Professional license).
EmbedAllFonts	bool	Strip or Embed all fonts into the PDF. Certain licensed fonts may not allow embedding and will therefore not be embedded. (Requires a Muhimbi PDF Converter Professional license).
SubsetFonts	bool	Specify if font-subsetting is enabled or not. Font subsetting embeds only those characters that are used in a document, instead of the entire font. This reduces the size of a PDF file that contains embedded fonts, but may make future content changes problematic.
ViewerPreferences	PDFViewer Preferences	Settings related to how PDF files behave when opened in a PDF Reader. Please note that some settings may not work in all PDF readers. See section 7.1 for details.

Property	Type	Description
PostProcessFile	bool	Pass the generated PDF through the Post Processor to strip / embed fonts, apply Fast Web View or convert to a different PDF Version. Setting this value is not needed to apply options specified in <i>Viewer Preferences</i> . Setting this value to true requires a license for the PDF Converter Professional.

### 3.2.14 The SystemSettings class

System settings can optionally be overridden using the *OpenOptions.SystemSettings* property.

Property	Type	Description
TaskMonitorSettings	TaskMonitor Settings	Optional settings for the service's Task Monitor (for the current request only).

### 3.2.15 The TaskMonitorSettings class

Optional settings for the service's Task Monitor (for the current request only).

Property	Type	Description
MaxHungCount	int	Maximum number of seconds before a converter is considered 'hanging' and will be terminated. Specify '0' to use the default value from the config file.
MaxPendingCount	int	Maximum number of seconds to wait after a request for termination has been made and the program has not responded. Specify '0' to use the default value in the config file.
MaxRunDuration	int	Maximum run time in seconds after which a conversion process will be terminated. Specify '0' to use the default value in the config file.

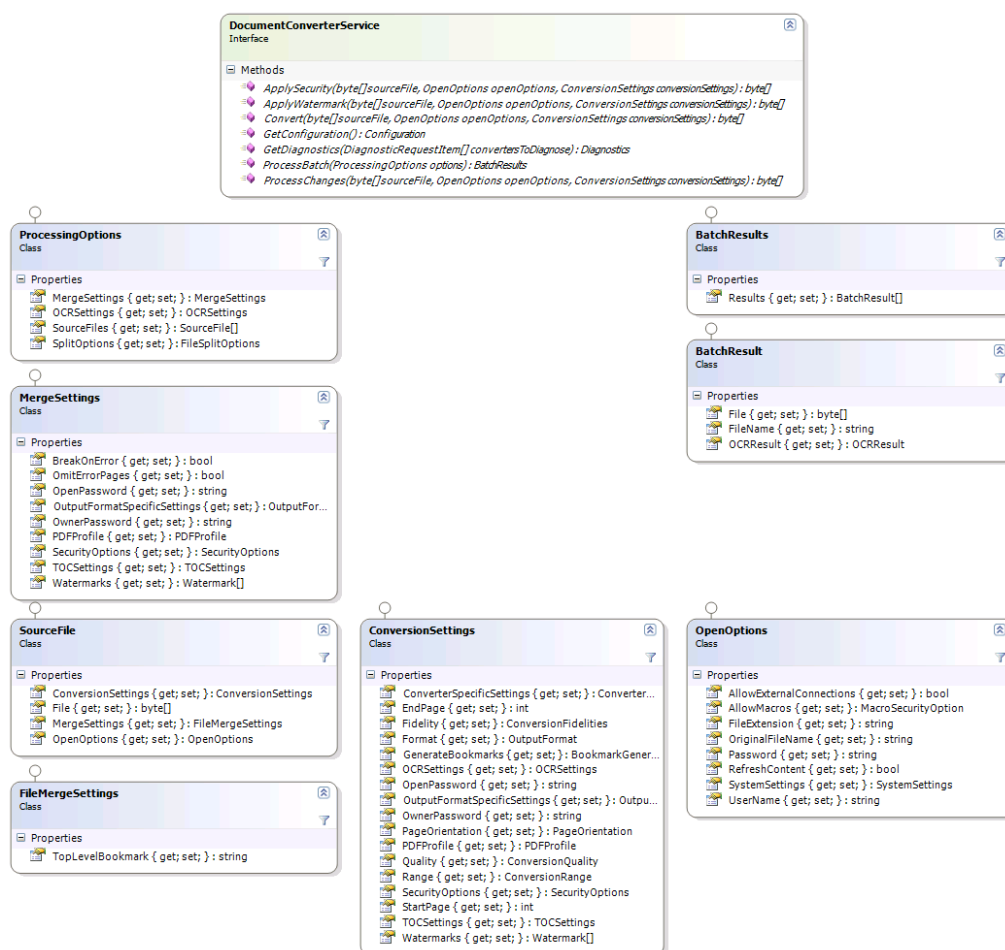
## 3.3 Working with ProcessBatch (Merging / Splitting files)

The *Muhimbi Document Conversion Service* allows multiple files to be merged into a single PDF file or a single file to be split into separate files. These actions are carried out using the *ProcessBatch* method described in this section.

### 3.3.1 Merging files

The key features of the merging facility are as follows:

1. Convert and merge any supported file format / URL (inc. HTML, AutoCAD, MS-Office, InfoPath, TIFF) or merge existing PDF files.
2. Apply different watermarks on each individual file as well as on the entire merged file (e.g. page numbering).
3. Apply PDF Security settings and restrictions on the merged file.
4. Optionally skip (and report) corrupt / unsupported files.
5. Add PDF Bookmarks for each converted file.
6. Apply any *ConversionSetting* supported by the regular conversion process.



The Web Service method that controls merging of files is called *ProcessBatch* (highlighted in the screenshot above). It accepts a *ProcessingOptions* object that holds all information about the source files to convert and the *MergeSettings* to apply, which may include security and watermarking related settings. A *BatchResults* object is returned that, when it comes to merging of files, always contains a single file that holds the byte array for the merged PDF file.

For a full code example see section 4.7 *Merging multiple files into a single PDF using .NET*.

### 3.3.2 Splitting files

The key features of the splitting facility are as follows:

1. Split a single PDF file into one or more individual PDF files.
2. Split based on number of pages or bookmarks.
3. Automatically generate numbered file names using .NET's formatting syntax, e.g. 'split-{0:D3}.pdf' will use 3 digits for the sequential numbers starting at 'split-001.pdf'. When splitting by bookmark then an optional {1} parameter can be inserted in the file name to include the name of the bookmark as well.
4. Can be combined in combination with other actions, e.g. convert & merge.

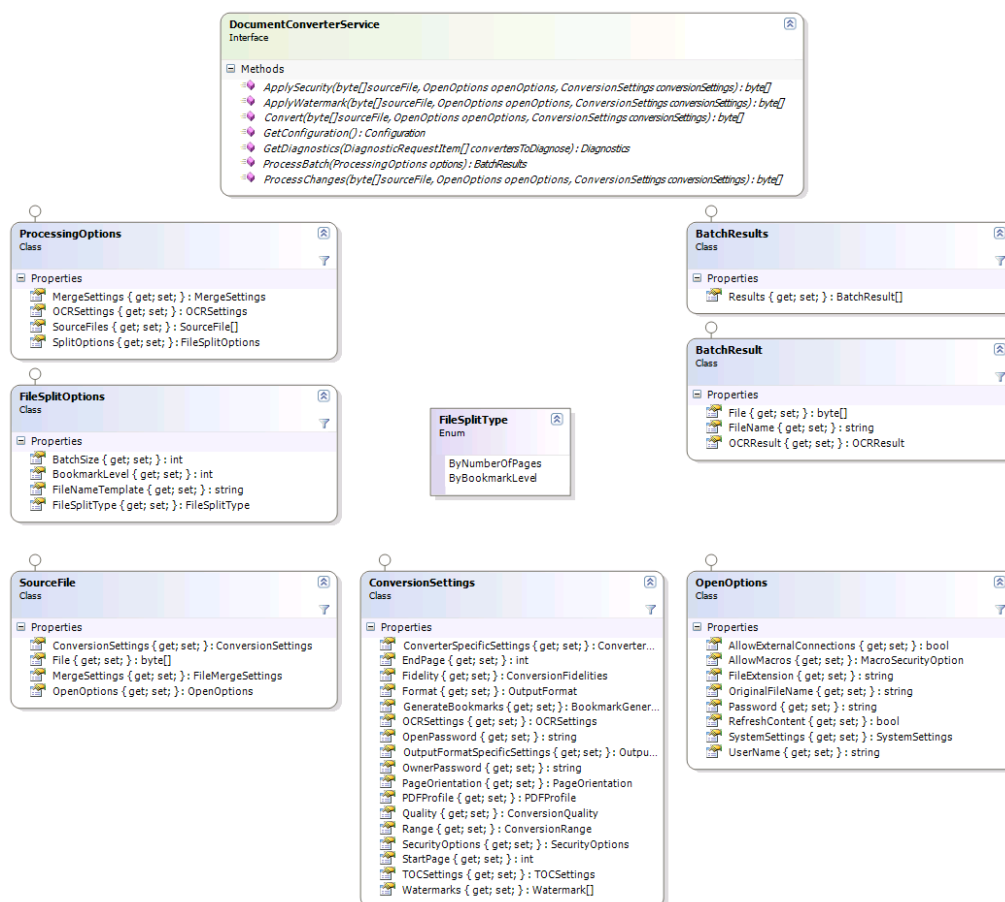
*A note about splitting based on bookmark levels:* PDFs store bookmarks at the page level, so it is not clear on what part of the page a heading starts or ends. As a result an extra page will always be exported for each file split based on bookmark levels.

For example let's assume the following document:

- **Page 1:** Contains chapter 1 and sections 1.1. and 1.2.
- **Page 2:** Contains the last paragraph of 1.2 and all of chapter 2.
- **Page 3:** Contains Chapter 3.

When splitting this document based on bookmarks using '1' as the batch size then the following files will be created:

- **File 1:** Contains page 1 and 2 as expected.
- **File 2:** Contains pages 2 and 3 even though Chapter 2 is only really part of page 2. This is because there is no way to know if Chapter 2 runs over into page 3 or not.
- **File 3:** Contains Chapter 3.



The object classes involved in splitting files is similar to the ones used by the merging facility described in 3.3.1.

The Web Service method that controls splitting (as well as merging) of files is called *ProcessBatch*. It accepts a *ProcessingOptions* object that holds all information about the files to process and the operations to apply. A *Results* object is returned that, when it comes to splitting of files, contains one or more results that hold the contents of the file as well as the suggested output file name, which you may use to save the file locally.

As the *ProcessingOptions* class accepts both *MergeSettings* and *SplitOptions* it is possible to *convert and merge* a set of input files (see 3.3.1) and then split up the results, all in a single web service call. Just populate the various properties and the system will take care of the rest.

Details about the various classes involved can be found below. A code sample can be found in section 4.9.



### 3.3.3 The ProcessingOptions class

This object is the only parameter passed into the *ProcessBatch* method. It allows all parameters required for the batch operation to be passed in.

Property	Type	Description
MergeSettings	MergeSettings	Settings associated with PDF Merge operations, see 3.3.4.
OCRSettings	OCR Settings	Optional settings for when the source file is Image based and OCR must be carried out.
SourceFiles	SourceFile[]	An array of files associated with the batch operation.
SplitOptions	FileSplitOptions	Settings associated with PDF Split operations, see 3.3.5.

### 3.3.4 The MergeSettings class

Any settings associated with a PDF Merge batch process are communicated using this class.

Property	Type	Description
BreakOnError	Bool	Specify if any error should abort the entire batch process or if the offending file should be skipped.
OmitErrorPages	Bool	Control if error pages are inserted in the merged document for files that fail to convert. This only has effect if <i>BreakOnError</i> is set to 'False'.
OutputFormatSpecific Settings	OutputFormat SpecificSettings	An instance of an object that contains settings specific to the output format, e.g. <i>ViewerPreferences</i> or <i>Font Embedding</i> settings when the output format is PDF (See section 0 for details).
OpenPassword	String	The 'open password' to be applied to the PDF file containing all merged documents. See 3.2.3 for details.
OwnerPassword	String	The 'owner password' to be applied to the PDF file containing all merged documents. See 3.2.3 for details.
PDFProfile	PDFProfile	The PDF Profile to use for the PDF file containing all merged documents. See 3.2.3 for details.
SecurityOptions	SecurityOptions	Security restrictions to apply to the PDF file containing all merged documents. See 3.2.3 for details.
Watermarks	Watermark[]	Watermarks to apply to the PDF file containing all merged documents. Note that it is still possible to specify Watermarks for each individual file in the batch as well using the <i>SourceFile.ConversionSettings</i> property (see 3.3.6).



### 3.3.5 The FileSplitOptions class

Any settings associated with PDF Split operations are communicated using this class.

Property	Type	Description
FileSplitType	FileSplitType	How to split the file: <i>ByNumberOfPages</i> or <i>ByBookmarkLevel</i> .
BatchSize	Int	When splitting by the number of pages set this value to the number of pages to use per file.
BookmarkLevel	Int	When splitting by bookmark set this value to the bookmark level to split on.
FileNameTemplate	String	Template to use for generating file names using .NET formatting standards, e.g. 'spf-{0:D3}.pdf' generates names starting with 'spf-001.pdf'. When splitting by bookmark then an optional {1} parameter can be inserted in the file name to include the name of the bookmark as well.

### 3.3.6 The SourceFile class

An array of *SourceFile* objects is passed to the server as part of the *ProcessingOptions* class.

Property	Type	Description
ConversionSettings	ConversionSettings	The settings to use for this particular file, including Quality, Watermarks and page ranges. See 3.2.3 for details.
File	Byte[]	The content of the file to process or the Byte array of the URL to convert <i>System.Text.Encoding.UTF8.GetBytes(url)</i>
MergeSettings	FileMergeSettings	Settings associated with merging this file. See 3.3.7 for details.
OpenOptions	OpenOptions	Any options for opening the file, see 3.2.2 for details.

### 3.3.7 The FileMergeSettings class

File specific settings associated with merging individual documents are passed using this class.

Property	Type	Description
TopLevelBookmark	String	The name to use as the 'top level bookmark' in the combined PDF file.

### 3.3.8 The BatchResults class

The results of a batch operation are passed back in the *BatchResult* class.

Property	Type	Description
Results	BatchResult[]	One or more results coming out of the batch operation. Note that in case of a <i>file merge operation</i> the merged file is always stored in element 0.

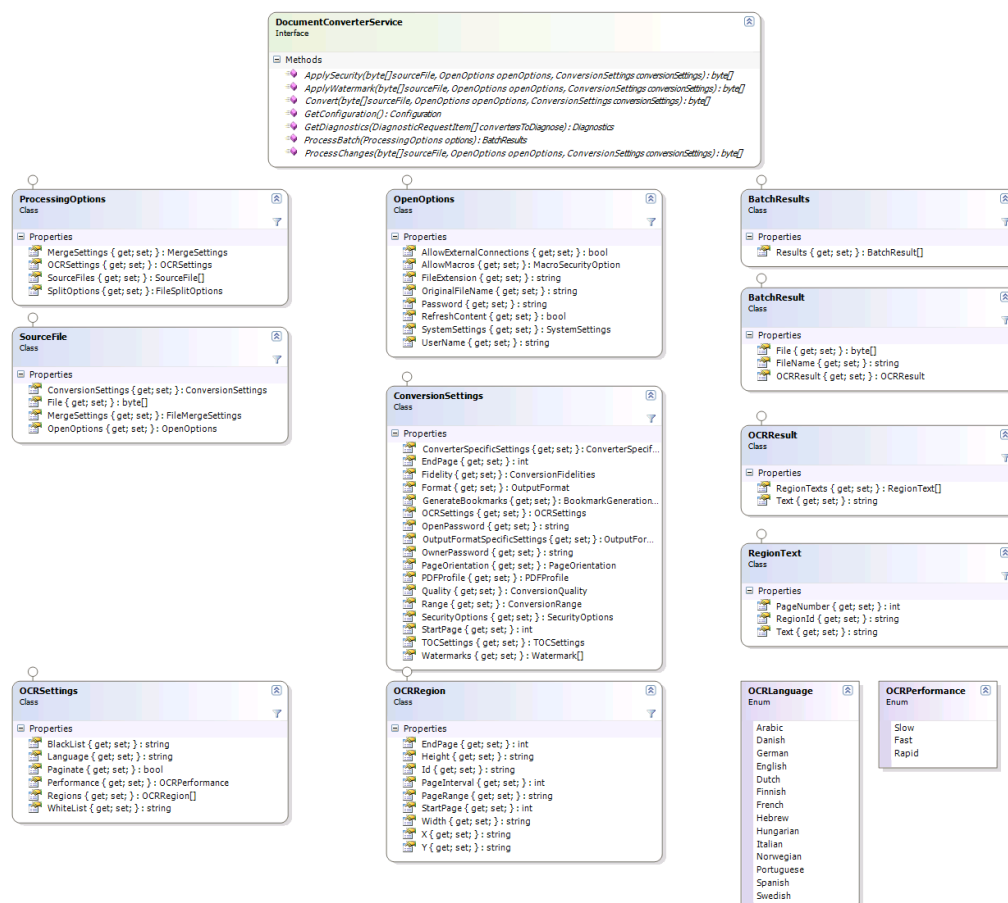
### 3.3.9 The BatchResult class

Individual results, part of the *BatchResults* class, are passed using the *BatchResult* class.

Property	Type	Description
File	Byte[]	The file associated with the result, e.g. the split or merged file.
Filename	String	The suggested file name to use for saving the file. Please note that this is just a suggestion and can be ignored. This is mainly used when splitting PDF files, see 3.3.5.
OCRResult	OCRResult	Object containing the result of OCR processing of a document.

## 3.4 OCR (Optical Character Recognition)

The *Muhimbi Document Conversion Service* provides support for two OCR scenarios: Converting bitmap based content to searchable and indexable PDFs AND extracting text from bitmap based content. For examples see chapter 6.



### 3.4.1 The OCRSettings class

An instance of this class is optionally passed in the *OCRSettings* property of the *ConversionSettings* class for operations where OCR needs to be carried out during conversion.

Property	Type	Description
Performance	OCRPerfor mance	Indicate what performance / accuracy to use. <i>Slow</i> will give best result, but usually takes longer (depending on the source material).
Language	String	The language to use for recognizing text. Can be any value of the <i>OCRLanguage</i> enumeration or custom values if custom character sets / languages have been defined.

Property	Type	Description
WhiteList	String	Optional string of characters to limit recognition to. For example specify <i>1234567890</i> to only recognize numbers to prevent a 1 being recognized as <i>i</i> or <i>l</i> .
BlackList	String	Optional string of characters to skip recognition for. For example specify <i>1234567890</i> to not recognize any numbers, which will improve accuracy for normal text providing the text doesn't contain numbers.
Regions	OCRRegion[]	Optional regions to OCR if only part of the page or document need processing.
Paginate	bool	Should the source files be paginated (for images that span multiple pages)?

### 3.4.2 The OCRRegion class

When OCR must be carried out on a section of a document, rather than the entire document, a set of regions can optionally be passed into the *OCRSettings.Regions* property.

Property	Type	Description
Id	String	Optional tag which can later be used to retrieve the recognised text. (Not supported in 7.1)
X	String	The X coordinate of the region in pt (1/72")
Y	String	The Y coordinate of the region in pt (1/72")
Width	String	The Width of the region in pt (1/72")
Height	String	The Height of the region in pt (1/72")
StartPage	int	The index of the first page this region applies to.
EndPage	int	The last page this region applies to.
PageInterval	int	The interval the region applies to (e.g. '2' to skip every other page in double sided documents)
PageRange	String	An optional string representation of the range of pages the region applies to. For example "1,3,7,10-15". If specified, this is in addition to the values stored in the <i>StartPage</i> and <i>EndPage</i> properties.

### 3.4.3 The OCRResult class

For OCR operations that return text, details are returned in an instance of the *OCRResult* class.

Property	Type	Description
RegionTexts	RegionText[]	Individual results for the various regions specified in <i>OCRRegion</i> .
Text	string	The full textual result of OCR processing.

### 3.4.4 The RegionText class

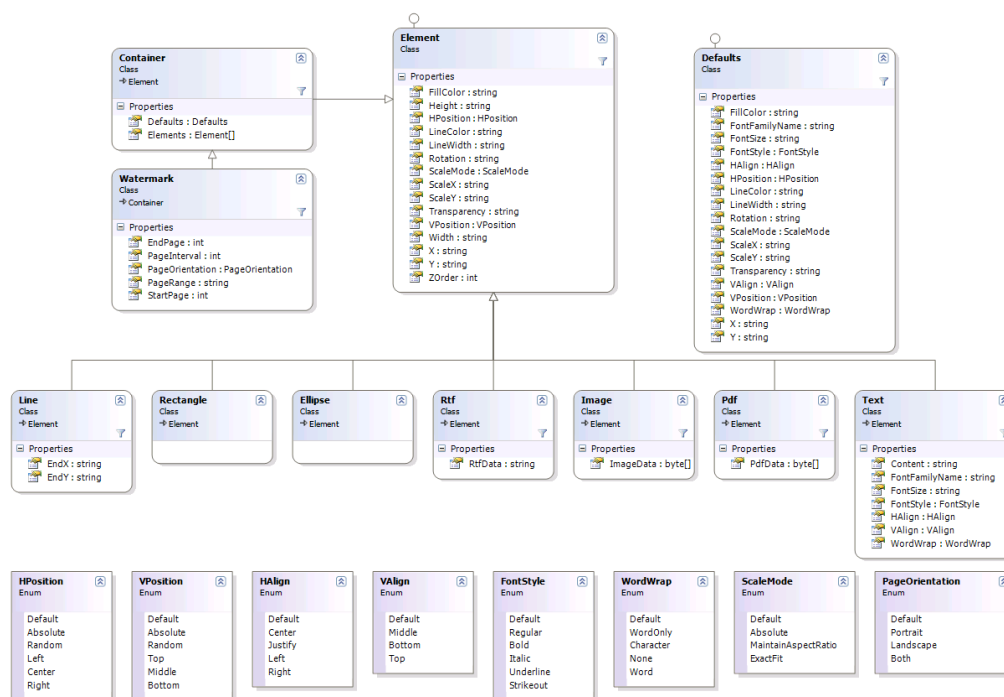
Text associated with a region is returned in an instance of the *RegionText* class.

Property	Type	Description
RegionId	String	The ID of the region the text belongs to, as specified in <i>OCRRegion.Id</i> .
PageNumber	int	The page number the text belongs to.
Text	String	The OCR'd text for the region.

## 3.5 Watermarking

The *Muhimbi Document Conversion Service* contains a very flexible system for applying watermarks to documents. Multiple watermarks can be applied to the same page and watermarks can be applied to page ranges or certain page types such as *odd*, *even*, *portrait* or *landscape*.

Watermarks are passed as part of the *ConversionSettings* object, a parameter of the *Convert* method. For details see sections 3.2.1 and 3.2.3, for a code example see chapter 4.11.



### 3.5.1 The Watermark class

An instance of this class is passed by the *Convert* method, as part of the *ConversionSettings* object, in order to apply watermarks to the converted document.

Note that some of this class' properties are inherited from the *Container* type, which in turn inherits from the *Element* type. The properties are largely self describing, the ones that require explanation are as follows:

Property	Type	Description
Defaults	Defaults	The default values for each of the watermark's elements, e.g. <i>LineColor</i> , <i>alignment</i> , <i>transparency</i> , etc. For details see 3.5.4.
Elements	Element[]	A list of elements, e.g. <i>Text</i> , <i>Line</i> or <i>Image</i> that make up the watermark. For details see 3.5.2.
EndPage	Int	The last page the watermark applies to.

		Defaults to the last page.
PageInterval	Int	The page interval that determines if a watermark should be applied to the current page number, e.g. '2' to apply the watermark to every other page.
PageOrientation	Page Orientation	Specifies what page orientation the watermark applies to: <i>Portrait</i> , <i>Landscape</i> or <i>Both</i> .
PageRange	String	An optional string representation of the range of pages the watermark applies to. For example "1,3,7,10-15". If specified, this is in addition to the values stored in the <i>StartPage</i> and <i>EndPage</i> properties.
StartPage	Int	The first page of the document the watermark applies to. Defaults to the first page.
ZOrder	Int	For the watermark, not for individual elements, a negative z-order means that the watermark will be displayed behind the content of the document. A positive value will display the watermark on top of the content.

### 3.5.2 The Element class

The *Element* class is the base class for the individual watermark elements such as *Line*, *Rectangle*, *Image*, *Text*, *PDF* etc. Do not instantiate this class directly, instead use one of the derived types defined in 3.5.3.

The properties shared by all individual element types are as described below. Note that some properties, which you would have expected to be of type *int* or *float*, are of type *string*. The reason for this is to make it possible to determine if a value has been specified at all and to allow different units of measure. **If a value has not been specified then for most properties its value will be read from the corresponding *Defaults* instance.**

Property	Type	Description
FillColor	String	The color of the element's fill in <i>#rrggbb</i> or <i>#aarrggbb</i> format where <i>aa</i> represents the alpha / transparency.
Height	String	The height of the element. Note that this field is of type <i>string</i> to allow the unit of measure to be specified (future version).
HPosition	HPosition	The horizontal position of the element.
LineColor	String	The color of the element's line in <i>#rrggbb</i> or <i>#aarrggbb</i> format where <i>aa</i> represents the alpha / transparency.
LineWidth	String	The width of the element's line. Note that this field is of type <i>string</i> to allow the unit of

Property	Type	Description
		measure to be specified (future version).
Rotation	String	The rotation to apply to the element in degrees. Note that this field is of type <i>string</i> to allow the system to determine if it has been specified or not.
ScaleMode	ScaleMode	The behaviour to use when scaling the element, e.g. maintain Aspect Ratio or ExactFit.
ScaleX	String	The horizontal scaling to apply to the element, where 1 is the original size. Any number between 0 and 1 reduces the size whereas any number above 1 increases the size. Note that this field is of type <i>string</i> to allow different scaling units to be specified in a future version.
ScaleY	String	The vertical scaling to apply to the element, where 1 is the original size. Any number between 0 and 1 reduces the size whereas any number above 1 increases the size. Note that this field is of type <i>string</i> to allow different scaling units to be specified in a future version.
Transparency	String	The element's transparency where 1 is opaque and 0 is completely transparent.
VPosition	VPosition	The vertical position of the element.
Width	String	The width of the element. Note that this field is of type <i>string</i> to allow the unit of measure to be specified (future version).
X	String	The x-coordinate of the element. Note that this field is of type <i>string</i> to allow the unit of measure to be specified (future version).
Y	String	The y-coordinate of the element. Note that this field is of type <i>string</i> to allow the unit of measure to be specified (future version).
ZOrder	Int	The z-order (layer index) of the element. A lower value indicates that the element will be drawn further in the background.

### 3.5.3 Individual Element Types

As all individual elements inherit from the *Element* class, they largely share the same properties.

The currently recognised Element Types (Shapes) are as follows.

- **Line:** Represents a single line. Please note that the *Width* and *Height* properties are ignored, instead it uses the *EndX* and *EndY* properties.



- **Rectangle:** Represents a rectangle. This shape does not implement any additional properties.
- **Ellipse:** Represents an ellipse. This shape does not implement any additional properties.
- **Rtf:** Represents a piece of text encoded in RTF format. The text is specified in the *RtfData* property.
- **Image:** Represents an image. The image's binary data is stored in the *ImageData* (byte[]) property. The following image types are supported:
  - Bmp
  - JPG
  - GIF
  - PNG
  - TIFF
  - WMF
  - EMF / EMF+
- **Pdf:** Represents an existing PDF file that is used as the watermark. If the PDF document contains multiple pages then the first page is used as the watermark. The PDF's data is stored in the *PdfData* (byte[]) property.
- **Text:** Represents a text box that allows plain text to be specified with full control over horizontal and vertical alignment, font face and size as well as word wrapping. The actual text is stored in the *Content* property. The text field also allows field codes such as *page number* to be embedded. For details see 3.5.5.

### 3.5.4 The Defaults class

The *Defaults* class allows default values to be specified for all elements in the watermark. For example, if all lines and text boxes are red then there is no need to specify the colour on each individual element.

The following properties are available:

Property	Type	Description
FillColor	String	The color of the element's fill in <i>#rrggbb</i> or <i>#aarrggbb</i> format where <i>aa</i> represents the alpha / transparency.
FontFamilyName	String	The name of the font to use. When the font is not found the system will throw an exception.
FontSize	String	The size of the font.
FontStyle	FontStyle	The style of the text. Multiple values can be combined, e.g. <i>FontStyle.Bold</i>   <i>FontStyle.Italic</i> .
HAlign	HAlign	Horizontal alignment of text stored in a <i>Text</i>

Property	Type	Description
		element.
HPosition	HPosition	The horizontal position of the element.
LineColor	String	The color of the element's line in <i>#rrggbb</i> or <i>#aarrggbb</i> format where <i>aa</i> represents the alpha / transparency.
LineWidth	String	The width of the element's line. Note that this field is of type <i>string</i> to allow the unit of measure to be specified (future version).
Rotation	String	The rotation to apply to the element in degrees. Note that this field is of type <i>string</i> to make allow the system to determine if it has been specified or not.
ScaleMode	ScaleMode	The behaviour to use when scaling the element, e.g. maintain Aspect Ratio or ExactFit.
ScaleX	String	The horizontal scaling to apply to the element, where 1 is the original size. Any number between 0 and 1 reduces the size whereas any number above 1 increases the size.  Note that this field is of type <i>string</i> to allow different scaling units to be specified in a future version.
ScaleY	String	The vertical scaling to apply to the element, where 1 is the original size. Any number between 0 and 1 reduces the size whereas any number above 1 increases the size.  Note that this field is of type <i>string</i> to allow different scaling units to be specified in a future version.
Transparency	String	The element's transparency where 1 is opaque and 0 is completely transparent.
VAlign	VAlign	Vertical alignment of text stored in a <i>Text</i> element.
VPosition	VPosition	The vertical position of the element.
WordWrap	WordWrap	The word wrapping behaviour of text stored in a <i>Text</i> element.
X	String	The x-coordinate of the element. Note that this field is of type <i>string</i> to allow the unit of measure to be specified (future version).
Y	String	The y-coordinate of the element. Note that this field is of type <i>string</i> to allow the unit of measure to be specified (future version).

### 3.5.5 Embedding field codes in the Text element

The *Text* and *RTF* elements allow field codes to be embedded, for example the *number of pages* or the *current date*. This makes it very simple to use watermarks to automatically generate headers and footers on each page, while taking orientation and page interval (Odd / Even pages) into account.

The following field codes are available for use:

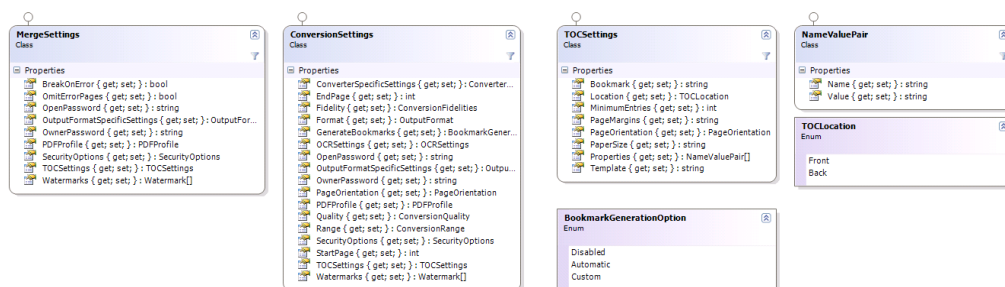
- **{LONG\_DATE}**: The long representation of the current date, equivalent in C# to *DateTime.Now.ToString()*
- **{LONG\_TIME}**: The long representation of the current time, equivalent in C# to *DateTime.Now.ToString()*
- **{DATE}**: The short representation of the current date, equivalent in C# to *DateTime.Now.ToShortDateString()*
- **{TIME}**: The short representation of the current time, equivalent in C# to *DateTime.Now.ToString()*
- **{PAGE}**: The number of the current page in the PDF file.
- **{NUMPAGES}**: The total number of pages in the PDF file.

Date and time fields are formatted using the regional settings of the account the Document Conversion Service is running under.

## 3.6 Table Of Contents

The *Muhimbi Document Conversion Service* allows for a Table Of Contents to be added to PDF files. And, although not limited to merge operations, it is particularly useful to create an overview of documents that have been merged into a single file.

For a detailed example see chapter 8 *Building a Table Of Contents*.



### 3.6.1 The TocSettings class

This class defines the various settings associated with the Table Of Contents. An instance of this class is passed to either *MergeSettings.TOCSettings* or *ConversionSettings.TOCSettings*.

Property	Type	Description
Bookmark	String	The TOC itself can have its own PDF bookmark to aid with navigation. Specify the text in this property.
Location	TOCLocation	TOCs can be added to the Front or Back of the document. Enter the relevant option here.
MinimumEntries	int	For certain, simple, documents that only have one or 2 bookmarks, it may not make sense to add a table of contents. Use this property to specify the minimum number of entries before a TOC is generated. The default value is '0', which will always create a TOC regardless of the number of entries.
PageMargins	String	<p>The margin / border around the generated TOC. It defaults to a uniform half inch margin.</p> <p>One or four {value}{dim} components separated by commas (,) where</p> <ul style="list-style-type: none"> <li>{value} is a numerical value</li> <li>{dim} is the dimension which can be 'mm', 'in.' or 'inches'. (Defaults to inches when nothing is specified)</li> </ul> <p>When multiple values are specified then the</p>

		sequence is: left, top, right and bottom. Example: "12mm, 24mm, 12mm, 24mm"
PageOrientation	PageOrientation	The orientation used by the TOC. <i>Portrait</i> , <i>Landscape</i> or <i>Default</i> . The <i>Default</i> option uses the same orientation as the page following (or preceding) the TOC depending on the value specified in <i>Location</i> .
PaperSize	String	The optional paper size to use for the TOC. Either: <ul style="list-style-type: none"> <li>A 'Named' paper size such as 'A4' or 'Letter' (See <a href="#">MSDN</a>)</li> <li>or a custom size in "{width}{dim}{sep}{height}{dim}" format where <ul style="list-style-type: none"> <li>- {width} and {height} are numerical values (decimal separator must be colon '.')</li> <li>- {dim} is the dimension which can be 'mm', 'in.' or 'inches'. (Defaults to inches when nothing is specified)</li> <li>- {sep} separates the width and the height, either 'by', comma (,) or the letter 'x'</li> </ul> </li> </ul> Example: "8.5 in. by 6 in."
Properties	NameValue Pair[]	Optional properties to pass to the XSL template for display or processing purposes.
Template	String	The XSL template (See 8.3) to use for formatting purposes. This can either be a string containing all the XSL, a path - local to the server running the conversion service - to the location of the XSL file, or a URL to the XSL file on a web (or SharePoint) server.

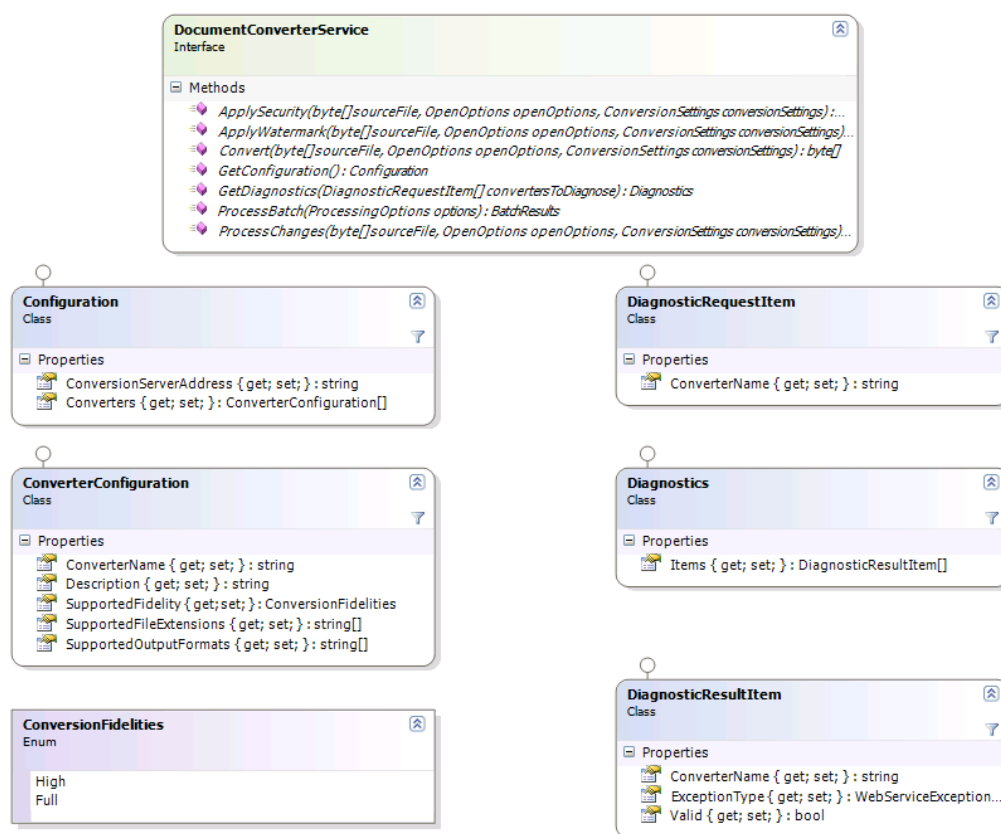
### 3.6.2 The NameValuePair class

Utility class for passing *named values* to a method or property. For example see *TocSettings.Properties*.

Property	Type	Description
Name	String	The name of the value to pass.
Value	String	The actual value to pass.

## 3.7 Configuration and Diagnostics

The Document Conversion Service comes with a *Configuration and Diagnostics* interface that allows the individual converters to be tested and information about the available converters to be retrieved.



### 3.7.1 Retrieving Configuration settings

The *GetConfiguration* method, part of the *DocumentConverterService* interface retrieves the server's configuration. The method call does not require any parameters and returns the results in an instance of the *Configuration* class.

The Configuration class has the following properties:

Property	Type	Description
ConversionServerAddress	String	The exact address the web service is listening on.
Converters	Converter Configuration[]	An array containing the list of converters available in the system. This list contains both converters that are shipped in the box as well as any custom converters.

Each item in the *Converters* array is represented by an instance of the *ConverterConfiguration* class, which has the following properties:

Property	Type	Description
ConverterName	String	The short name of the converter used for uniquely identifying it.
Description	String	A human readable description. Typically used for display in a user interface.
SupportedFidelity	Conversion Fidelities	The fidelity supported by the converter.
SupportedFileExtensions	String[]	An array of file extensions supported by the converter.
SupportedOutputFormats	String[]	An array of file extensions / formats the converter can output.

The array of converters largely matches the information stored in the Document Conversion Server's config file. For details see section 2.4.6 of the Administration Guide.

### 3.7.2 Running Diagnostic tests

The *GetDiagnostics* method, part of the *DocumentConverterService* interface, runs an end-to-end diagnostic test on each of the specified converters to see if everything has been configured properly and is working as expected. The method accepts an array of *DiagnosticRequestItem* instances and returns an object of type *DiagnosticResultItem*.

The *DiagnosticRequestItem* class has the following properties:

Property	Type	Description
ConverterName	String	The short name of the converter to run the diagnostics for.

The *Diagnostics* class has the following properties:

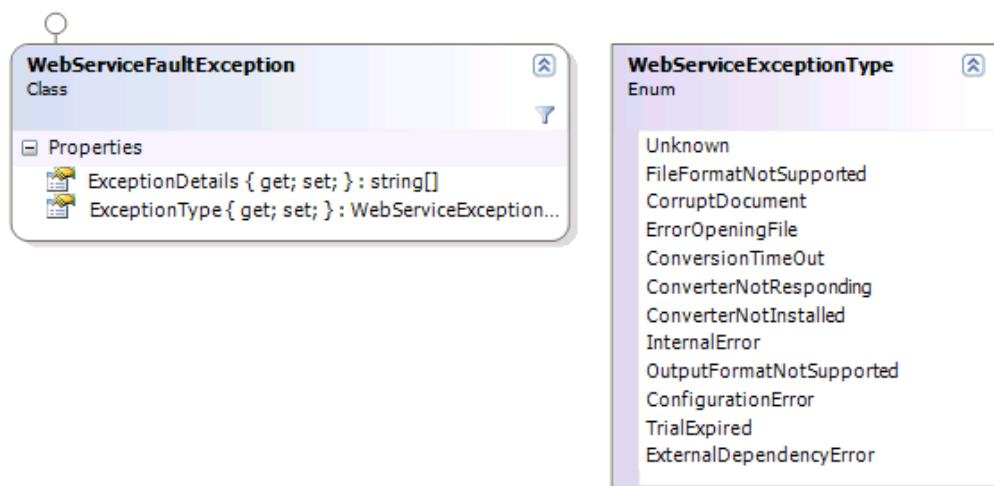
Property	Type	Description
Items	Diagnostic ResultItem[]	An array of items that holds the results.

The *DiagnosticResultItem* class has the following properties:

Property	Type	Description
ConverterName	String	The short name of the converter that this item holds the results for.
ExceptionType	WebService Exception Type	The type of exception that occurred during the validation (e.g. <i>ConverterNotInstalled</i> )
Valid	Bool	A flag indicating if the converter is valid (No errors encountered during diagnosis).

### 3.8 Exception handling

Any Exception that occurs anywhere inside the web service is automatically wrapped in a *WebServiceFaultException*. If the cause of the internal exception is known then the *ExceptionType* property is set to a value of the *WebServiceExceptionType* enumeration.



For examples of how to deal with these kind of exceptions see the sample code in sections 4.1 (.net) and 4.2 (Java).



## 4 Programmatically processing documents

### 4.1 PDF Conversion in .NET

Listed below is a basic example of how to convert a document to PDF format using a simple *WinForms* application. For a more comprehensive example see the .NET Sample code installed alongside each copy of the MDCS. Use the Start Menu to open the appropriate folder.

The latest version of this example can be found at the following page:

<http://blog.muhimbi.com/2009/12/converting-office-files-to-pdf-format.html>

This example does not explicitly set *ConversionSettings.Format*. As a result the file is converted to the default PDF format. It is possible to convert files to other file formats as well by setting this property to a value of the *OutputFormat* enumeration. For details see 4.6 *Cross-Converting between document types*.

1. Start a new Visual Studio project and select the project type of your choice. In this example we are using a standard .net 3.0 project of type *Windows Forms Application*. Name it 'Simple PDF Converter Sample'.
2. Add a *TextBox* and *Button* control to the form. Accept the default names of *textBox1* and *button1*.
3. In the *Solution Explorer* window, right-click *References* and select *Add Service Reference*.
4. In the *Address* box enter the WSDL address listed at the end of section 3. If the MDCS is located on a different machine then substitute *localhost* with the server's name.
5. Accept the default Namespace of *ServiceReference1* and click the OK button to generate the proxy classes.
6. Double click *Button1* and replace the content of the entire code file with the following:

```
using System;
using System.IO;
using System.ServiceModel;
using System.Windows.Forms;
using Simple_PDF_Converter_Sample.ServiceReference1;

namespace Simple_PDF_Converter_Sample
{
    public partial class Form1 : Form
    {
        // ** The URL where the Web Service is located. Amend host name if needed.
        string SERVICE_URL = "http://localhost:41734/Muhimbi.DocumentConverter.WebService/";

        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {

```

```

DocumentConverterServiceClient client = null;

try
{
    // ** Determine the source file and read it into a byte array.
    string sourceFileName = textBox1.Text;
    byte[] sourceFile = File.ReadAllBytes(sourceFileName);

    // ** Open the service and configure the bindings
    client = OpenService(SERVICE_URL);

    /** Set the absolute minimum open options
    OpenOptions openOptions = new OpenOptions();
    openOptions.OriginalFileName = Path.GetFileName(sourceFileName);
    openOptions.FileExtension = Path.GetExtension(sourceFileName);

    // ** Set the absolute minimum conversion settings.
    ConversionSettings conversionSettings = new ConversionSettings();
    conversionSettings.Fidelity = ConversionFidelities.Full;
    conversionSettings.Quality = ConversionQuality.OptimizeForPrint;

    // ** Carry out the conversion.
    byte[] convFile = client.Convert(sourceFile, openOptions, conversionSettings);

    // ** Write the converted file back to the file system with a PDF extension.
    string destinationFileName = Path.GetDirectoryName(sourceFileName) + @"\" +
                                Path.GetFileNameWithoutExtension(sourceFileName) +
                                "." + conversionSettings.Format;
    using (FileStream fs = File.Create(destinationFileName))
    {
        fs.Write(convFile, 0, convFile.Length);
        fs.Close();
    }

    MessageBox.Show("File converted to " + destinationFileName);
}
catch (FaultException<WebServiceFaultException> ex)
{
    MessageBox.Show("FaultException occurred: ExceptionType: " +
                    ex.Detail.ExceptionType.ToString());
}
catch (Exception ex)
{
    MessageBox.Show(ex.ToString());
}
finally
{
    CloseService(client);
}
}

/// <summary>
/// Configure the Bindings, endpoints and open the service using the specified address.
/// </summary>
/// <returns>An instance of the Web Service.</returns>
public static DocumentConverterServiceClient OpenService(string address)
{
    DocumentConverterServiceClient client = null;

    try
    {
        BasicHttpBinding binding = new BasicHttpBinding();
        // ** Use standard Windows Security.
        binding.Security.Mode = BasicHttpSecurityMode.TransportCredentialOnly;
        binding.Security.Transport.ClientCredentialType =
                                HttpClientCredentialType.Windows;
        // ** Increase the client Timeout to deal with (very) long running requests.
        binding.SendTimeout = TimeSpan.FromMinutes(30);
        binding.ReceiveTimeout = TimeSpan.FromMinutes(30);
        // ** Set the maximum document size to 50MB
    }
}

```

```

binding.MaxReceivedMessageSize = 50*1024*1024;
binding.ReaderQuotas.MaxArrayLength = 50 * 1024 * 1024;
binding.ReaderQuotas.MaxStringLength = 50 * 1024 * 1024;

// ** Specify an identity (any identity) in order to get it past .net3.5 sp1
EndpointIdentity epi = EndpointIdentity.CreateUpnIdentity("unknown");
EndpointAddress epa = new EndpointAddress(new Uri(address), epi);

client = new DocumentConverterServiceClient(binding, epa);

client.Open();

return client;
}
catch (Exception)
{
    CloseService(client);
    throw;
}
}

/// <summary>
/// Check if the client is open and then close it.
/// </summary>
/// <param name="client">The client to close</param>
public static void CloseService(DocumentConverterServiceClient client)
{
    if (client != null && client.State == CommunicationState.Opened)
        client.Close();
}
}
}

```

Providing the project and all controls are named as per the steps above, the project should compile without errors. Run it, enter the full path to the source file, e.g. an MS-Word document, and click the button to start the conversion process. The conversion may take a few seconds depending on the complexity of the document.

Note that in this example we are programmatically configuring the WCF Bindings and End Points. If you wish you can use a declarative approach using the config file. For more information about working with WCF see [http://msdn.microsoft.com/en-us/library/ms735119\(v=VS.90\).aspx](http://msdn.microsoft.com/en-us/library/ms735119(v=VS.90).aspx)

## 4.2 PDF Conversion in Java (WSImport)

Even though the MDCS itself must run on a Windows based server, it has been designed to interoperate with non Windows platforms such as Java. This section describes how to convert documents to PDF format using a Java based environment.

The full version of the sample code discussed in this chapter, including pre generated proxies, is installed alongside each copy of the MDCS. Use the Start Menu to open the appropriate folder.

The example described below assumes the following:

1. The JDK has been installed and configured.
2. The MDCS and all prerequisites have been installed in line with the Administration Guide.
3. The MDCS is running in the default *anonymous mode*. This is not an absolute requirement, but it makes initial experimentation much easier.

The first step is to generate proxy classes for the web service by executing the following command:

```
wsimport http://localhost:41734/Muhimbi.DocumentConverter.WebService/?wsdl
-d src -Xnocompile -p com.muhimbi.ws
```

Feel free to change the package name and destination directory to something more suitable for your organisation.

*Wsimport* automatically generates the Java class names. Unfortunately some of the generated names are rather long and ugly so you may want to consider renaming some, particularly the Exception classes, to something friendlier. This, however, means that if you ever run *wsimport* again you will need to re-apply those changes.

Once the proxy classes have been created add the following sample code to your project. Run the code and make sure the path to the document to convert is specified on the command line.

This example sets *ConversionSettings.Format* to *OutputFormat.PDF*. As a result the file is converted to the default PDF format. It is possible to convert files to other file formats as well by setting this property to a different value. For details see 4.6 *Cross-Converting between document types*.

```
package com.muhimbi.app;

import com.muhimbi.ws.*;
import java.io.*;
import java.net.URL;
import java.util.List;
import javax.xml.bind.JAXBElement;
import javax.xml.namespace.QName;

public class WsClient {

    private final static String DOCUMENTCONVERTERSERVICE_WSDL_LOCATION =
        "http://localhost:41734/Muhimbi.DocumentConverter.WebService/?wsdl";
```

```

public static void main(String[] args) {
    try {
        if (args.length != 1) {
            System.out.println("Please specify a single file name on the command line.");
        } else {
            // ** Process command line parameters
            String sourceDocumentPath = args[0];
            File file = new File(sourceDocumentPath);
            String fileName = getFileName(file);
            String fileExt = getFileExtension(file);
            System.out.println("Converting file " + sourceDocumentPath);

            // ** Initialise Web Service
            DocumentConverterService_Service dcss = new DocumentConverterService_Service(
                new URL(DOCUMENTCONVERTERSERVICE_WSDL_LOCATION),
                new QName("http://tempuri.org/", "DocumentConverterService"));
            DocumentConverterService dcs = dcss.getBasicHttpBindingDocumentConverterService();

            // ** Only call conversion if the file's extension is supported
            if (isFileExtensionSupported(fileExt, dcs)) {
                // ** Read source file from disk
                byte[] fileContent = readFile(sourceDocumentPath);

                // ** Converting the file
                OpenOptions openOptions = getOpenOptions(fileName, fileExt);
                ConversionSettings conversionSettings = getConversionSettings();
                byte[] convertedFile = dcs.convert(fileContent, openOptions, conversionSettings);

                // ** Writing converted file to file system
                String destinationDocumentPath = getPDFDocumentPath(file);
                writeFile(convertedFile, destinationDocumentPath);
                System.out.println("File converted successfully to " + destinationDocumentPath);
            } else {
                System.out.println("The file extension is not supported.");
            }
        }
    } catch (IOException e) {
        System.out.println(e.getMessage());
    } catch (DocumentConverterServiceGetConfigurationWebServiceFaultExceptionFaultFaultMessage e) {
        printException(e.getFaultInfo());
    } catch (DocumentConverterServiceConvertWebServiceFaultExceptionFaultFaultMessage e) {
        printException(e.getFaultInfo());
    }
}

public static OpenOptions getOpenOptions(String fileName, String fileExtension) {
    ObjectFactory objectFactory = new ObjectFactory();
    OpenOptions openOptions = new OpenOptions();
    openOptions.setOriginalFileName(objectFactory.createOpenOptionsOriginalFileName(fileName));
    openOptions.setFileExtension(objectFactory.createOpenOptionsFileExtension(fileExtension));
    return openOptions;
}

public static ConversionSettings getConversionSettings() {
    ConversionSettings conversionSettings = new ConversionSettings();
    conversionSettings.setQuality(ConversionQuality.OPTIMIZE_FOR_PRINT);
    conversionSettings.setRange(ConversionRange.ALL_DOCUMENTS);
    conversionSettings.getFidelity().add("Full");
    conversionSettings.setFormat(OutputFormat.PDF);
    return conversionSettings;
}

public static String getFileName(File file) {
    String fileName = file.getName();
    return fileName.substring(0, fileName.lastIndexOf('.'));
}

```

```

public static String getFileExtension(File file) {
    String fileName = file.getName();
    return fileName.substring(fileName.lastIndexOf('.') + 1, fileName.length());
}

public static String getPDFDocumentPath(File file) {
    String fileName = getFileName(file);
    String folder = file.getParent();
    if (folder == null) {
        folder = new File(file.getAbsolutePath()).getParent();
    }
    return folder + File.separatorChar + fileName + '.' + OutputFormat.PDF.value();
}

public static byte[] readFile(String filepath) throws IOException {
    File file = new File(filepath);
    InputStream is = new FileInputStream(file);
    long length = file.length();
    byte[] bytes = new byte[(int) length];

    int offset = 0;
    int numRead;
    while (offset < bytes.length
        && (numRead = is.read(bytes, offset, bytes.length - offset)) >= 0) {
        offset += numRead;
    }

    if (offset < bytes.length) {
        throw new IOException("Could not completely read file " + file.getName());
    }
    is.close();
    return bytes;
}

public static void writeFile(byte[] fileContent, String filepath) throws IOException {
    OutputStream os = new FileOutputStream(filepath);
    os.write(fileContent);
    os.close();
}

public static boolean isFileExtensionSupported(String extension, DocumentConverterService dcs)
    throws DocumentConverterServiceGetConfigurationWebServiceFaultExceptionFaultFaultMessage
{
    Configuration configuration = dcs.getConfiguration();
    final JAXBELEMENT<ArrayOfConverterConfiguration> converters = configuration
        .getConverters();
    final ArrayOfConverterConfiguration ofConverterConfiguration = converters.getValue();
    final List<ConverterConfiguration> cList = ofConverterConfiguration
        .getConverterConfiguration();

    for (ConverterConfiguration cc : cList) {
        final List<String> supportedExtension = cc.getSupportedFileExtensions()
            .getValue().getString();
        if (supportedExtension.contains(extension)) {
            return true;
        }
    }
    return false;
}

public static void printException(WebServiceFaultException serviceFaultException) {
    System.out.println(serviceFaultException.getExceptionType());
    JAXBELEMENT<ArrayOfstring> element = serviceFaultException.getExceptionDetails();
    ArrayOfstring value = element.getValue();
    for (String msg : value.getString()) {
        System.out.println(msg);
    }
}
}

```

### 4.3 PDF Conversion in Java (Axis2)

To keep things as simple as possible, and the number of external dependencies to a minimum, the majority of our Java based Sample Code (See section 4.2) use *wsimport* to generate Java based web service proxy classes. Unfortunately *wsimport* does not generate very friendly syntax and, even worse, does not support Web Services that expose derived classes (A feature used by some of our more advanced facilities).

All is not lost as *Apache's Axis2 Web Services framework* solves both problems. The generated proxy classes are much easier to use and derived classes can be populated and sent to the server without problems.

Below you will find an example about how to setup *Apache Axis2*, generate proxy classes and use those classes to convert a document to PDF. This is just a simple sample, for full details see the rest of this Developer Guide.

It is assumed that the reader is familiar with Java. Our samples are generated using Microsoft Windows, please note that the command line syntax on other platforms may be slightly different.

The full version of the sample code discussed here, including pre-generated proxies, is installed alongside each copy of the Muhimbi Conversion Service and includes Windows batch files for generating proxies, compiling and executing the code.

The latest version of this chapter is [available on our Blog](#).

The example described below assumes the following:

1. JDK 1.5 (or newer) has been installed and configured.
2. JAVA\_HOME is set and *javac* is on the path.
3. The Muhimbi Conversion Service and all prerequisites have been installed in line with the *Administration Guide*.
4. The Muhimbi Conversion Service is running in the default *anonymous mode*. This is not an absolute requirement, but it makes initial experimentation much easier.

#### Installing Axis2

The installation process for Axis 2 is relatively simple. The steps are as follows:

1. Download the [Axis2 binary distribution](#)
2. Unpack and place *axis2-<version>* in a location of your choice. (This will be your AXIS2\_HOME).
3. Have a look at *installation-std-bin.txt* and set environment variables depending on your platform.
4. Add “%AXIS2\_HOME%\bin” to your Path to make sure the executables can be found.
5. On non-Windows Machines execute `chmod 744 $AXIS2_HOME/bin/*.sh`



## Generating Proxies

With all the prerequisites in place, proxy classes for the web service can be generated by executing the following command:

```
wSDL2java.bat -uri
http://localhost:41734/Muhimbi.DocumentConverter.Web
Service/?wsdl -p com.muhimbi.ws
```

Feel free to change the package name to something more suitable to your organisation. The example below assumes *com.muhimbi.ws* is used.

If the Muhimbi Conversion Service is not located on the same system as where *wSDL2java* is executed then change *localhost* to the name of the server running the Conversion Service. You will also need to change the host name in the Conversion Service's config file. A convenient shortcut to the Installation folder is located in the Muhimbi Start Menu Group. Open *Muhimbi.DocumentConverter.Service.exe.config*, search for *baseAddress* and change the host name. Restart the *Muhimbi Document Converter Service* to activate the change.

## Sample Code

The sample code is as follows. Please note that exception handling has been omitted for the sake of clarity.

```
package com.muhimbi.app;

import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.rmi.RemoteException;
import java.util.Arrays;
import java.util.List;

import javax.activation.DataHandler;
import javax.activation.FileDataSource;

import com.muhimbi.ws.DocumentConverterServiceStub;
import com.muhimbi.ws.DocumentConverterServiceStub.Configuration;
import com.muhimbi.ws.DocumentConverterServiceStub.ConversionFidelities;
import com.muhimbi.ws.DocumentConverterServiceStub.ConversionFidelities_type0;
import com.muhimbi.ws.DocumentConverterServiceStub.ConversionQuality;
import com.muhimbi.ws.DocumentConverterServiceStub.ConversionRange;
import com.muhimbi.ws.DocumentConverterServiceStub.ConversionSettings;
import com.muhimbi.ws.DocumentConverterServiceStub.Convert;
import com.muhimbi.ws.DocumentConverterServiceStub.ConvertResponse;
import com.muhimbi.ws.DocumentConverterServiceStub.ConverterConfiguration;
import com.muhimbi.ws.DocumentConverterServiceStub.ConverterSpecificSettings_WordProcessing;
import com.muhimbi.ws.DocumentConverterServiceStub.GetConfiguration;
import com.muhimbi.ws.DocumentConverterServiceStub.OpenOptions;
import com.muhimbi.ws.DocumentConverterServiceStub.OutputFormat;
import com.muhimbi.ws.DocumentConverterServiceStub.RevisionsAndCommentsDisplayMode;
import com.muhimbi.ws.DocumentConverterServiceStub.RevisionsAndCommentsMarkupMode;
import com.muhimbi.ws.DocumentConverterService_Convert_WebServiceFaultExceptionFault_FaultMessage;
import com.muhimbi.ws.DocumentConverterService_GetConfiguration_WebServiceFaultExceptionFault_FaultMessage;

public class WsClient {

    private final static String CONVERTERSERVICE_WSDL_LOCATION =
        "http://localhost:41734/Muhimbi.DocumentConverter.WebService/?wsdl";
```



```

public static void main (String[] args)
    throws DocumentConverterService_Convert_WebServiceFaultExceptionFault_FaultMessage, IOException,
    DocumentConverterService_GetConfiguration_WebServiceFaultExceptionFault_FaultMessage {
    if (args.length != 1) {
        System.out.println("Please specify a single file name on the command line.");
    } else {
        String fileNameFull = args[0];
        String fileExt = fileNameFull.lastIndexOf(".") == -1 ? "" :
            fileNameFull.substring(fileNameFull.lastIndexOf(".") + 1);
        String fileName = fileNameFull.replace(".", fileNameFull + fileExt, "");

        DocumentConverterServiceStub stub = new DocumentConverterServiceStub(CONVERTERSERVICE_WSDL_LOCATION);

        // ** Is the file extension supported by the Converter?
        if (!".equals(fileExt) && fileExtensionSupported(fileExt, stub)) {
            // ** Specify the minimum conversion settings
            ConversionSettings settings = new ConversionSettings();
            settings.setFormat(OutputFormat.PDF);
            settings.setQuality(ConversionQuality.OptimizeForPrint);
            settings.setRange(ConversionRange.VisibleDocuments);

            // ** Only send WordProcessing specific settings if the file is in MS-Word format.
            // ** This is just an example to demonstrate the use of derived classes in Axis2.
            if ("doc".equalsIgnoreCase(fileExt) || "docx".equalsIgnoreCase(fileExt)) {
                ConverterSpecificSettings_WordProcessing csc = new ConverterSpecificSettings_WordProcessing();
                csc.setRevisionsAndCommentsDisplayMode(RevisionsAndCommentsDisplayMode.OriginalShowingMarkup);
                csc.setRevisionsAndCommentsMarkupMode(RevisionsAndCommentsMarkupMode.Balloon);
                csc.setProcessDocumentTemplate(false);
                settings.setConverterSpecificSettings(csc);
            }

            ConversionFidelities fi = new ConversionFidelities();
            fi.setConversionFidelities_type0(new ConversionFidelities_type0[]{ConversionFidelities_type0.Full});
            settings.setFidelity(fi);

            // ** Set the minimum open options
            OpenOptions oo = new OpenOptions();
            oo.setFileExtension(fileExt);

            Convert con = new Convert();
            // ** Read the contents of the file to convert into a byte array.
            con.setSourceFile(new DataHandler(new FileDataSource(fileNameFull)));
            con.setConversionSettings(settings);
            con.setOpenOptions(oo);

            // ** Carry out the conversion and save the results.
            ConvertResponse res = stub.convert(con);
            saveResult(res.getConvertResult().getInputStream(), fileName + ".pdf");

        } else {
            System.out.println("File extension not supported or not specified.");
        }
    }
}

private static void saveResult(InputStream in, String file) throws IOException {
    FileOutputStream out = new FileOutputStream(file);

    int i = 0;
    while ((i=in.read()) != -1) {
        out.write(i);
    }
    out.flush();
    out.close();
    in.close();
}

private static boolean fileExtensionSupported(String fileExt, DocumentConverterServiceStub stub) throws
    RemoteException, DocumentConverterService_GetConfiguration_WebServiceFaultExceptionFault_FaultMessage {

```

```
Configuration configuration = stub.getConfiguration(new GetConfiguration()).getGetConfigurationResult();
ConverterConfiguration[] converters = configuration.getConverters().getConverterConfiguration();

for (ConverterConfiguration cc : converters) {
    List<String> supportedExtension = Arrays.asList(cc.getSupportedFileExtensions().getString());
    if (supportedExtension.contains(fileExt)) {
        return true;
    }
}
return false;
}
```

### Compiling Code

Steps to compile your code very much depends on your environment, solution and build system. To build the sample code in this chapter use the following:

```
rmdir /S /Q .\bin
md bin

javac -d ./bin -cp "%AXIS2_HOME%/lib/*" -verbose
./src/com/muhimbi/ws/*.java

javac -d ./bin -cp ".\bin;%AXIS2_HOME%/lib/*" -verbose
./src/com/muhimbi/app/*.java
```

Please note that on non-Windows platforms the classpath (-cp) separator is ':' rather than ';'.

### Running the Code

To execute the sample and carry out the PDF conversion of a file named *test.docx* issue the following command.

```
java -classpath ".\bin;%AXIS2_HOME%/lib/*"
com.muhimbi.app.WsClient test.docx
```

Please note that on non-Windows platforms the classpath (-cp) separator is ':' rather than ';'.

## 4.4 PDF Conversion in Ruby / Rails

In this section we'll show how to create a simple ROR application to send a file to the PDF Converter. The latest version of this section, including details about how to install a full ROR environment on Linux, is [available on our Blog](#).

### Creating the Rails application

The *Muhimbi PDF Converter* exposes a comprehensive API via a standards based Web Services interface. A number of Web Service frameworks are available for Ruby (*Savon*, *Handsoap*), but in this example we use *Soap4R* to pre-generate Ruby proxies as it is simple, and it works.

In the example below we will create a basic PDF Conversion Rails application. If you are looking to add PDF Conversion to an existing Rails application then modifying this example to suit your exact needs should be simple.

1. Use a terminal application of your choice to navigate to the location where you wish to create the Rails application. We use *~Sites*.
2. Execute the following command to create the skeleton for the application:

```
rails new MuhimbiPDFConverter -O
```

3. Navigate to *MuhimbiPDFConverter*, edit *Gemfile* using a text editor of your choice and add the following line:

```
gem 'soap4r'
```

4. Install *bundler* as follows:

```
gem install bundler
```

5. Execute the following command to pull in the applicable gems (Make sure you are still in the *MuhimbiPDFConverter* directory)

```
bundle install
```

### Generating proxies

The quickest way (also from a performance perspective) to interact with a Web Service is to pre-generate proxy classes. This can be achieved easily using *soap4r*, which has already been added to the application as described above.

Before we can generate the proxies we need to make sure that the Muhimbi Conversion Service has been installed and is running.

1. Install the *Muhimbi PDF Converter Services* as described in Chapter 2 of the *Administration guide*.
2. Open *Muhimbi.DocumentConverter.Service.exe.config* in your favourite text editor. A handy shortcut to the configuration / installation folder can be found in the Windows Start Menu Group.
3. Search for *baseAddress* and change *localhost* to the DNS name or IP address of the server running the Conversion Service.
4. Restart the Conversion Service as follows:

```
Net stop "Muhimbi Document Converter Service"
Net start "Muhimbi Document Converter Service"
```

Please use the included Diagnostics Tool to verify that your installation is correct.

Back on the Ruby system carry out the following steps to generate the proxies:

1. Navigate to *MuhimbiPDFConverter/lib*
2. Execute the following command. Please replace *localhost* with the name or ip address of the server that runs the Muhimbi PDF Converter Service.

```
bundle exec wsdl2ruby.rb --wsdl
http://localhost:41734/Muhimbi.DocumentConverter.WebService/?wsdl
--type client
```

This generates four new files and places them in the *lib* folder. Note that the generated property and method names follow Ruby's naming convention and not the convention used in this Developer Guide. This mainly impacts the capitalisation of the first letters.

### Implementing the sample

All prerequisites are now in place. Let's add some code to tie it all together. If you prefer you can access the full source code from the *Sample Code* folder.

1. Start by generating a controller where the form will be posted to:
 

```
rails generate controller home upload_file
```
2. Delete the home page that comes with every new Rails application (Execute in the *MuhimbiPDFConverter* folder)
 

```
rm public/index.html
```
3. If you are using *Sublime-Text* then this is the moment to execute '*subl .*' to open the text editor and display the entire folder structure.
4. Edit *config/routes.rb* and after the following line

```
get "home/upload_file"
```

Add

```
post "home/upload_file"
root :to => 'home#upload_file'
```

5. Edit *app/views/home/upload\_file.html.erb* and add the following HTML:

```
<form method="post" enctype="multipart/form-data">
  <br/>
  <label for="file">Document:</label>
  <input type="file" name="file" id="file" />
  <br/>
  <label for="outputFormat">Output format:</label>
```

```
<select name="outputFormat" id="outputFormat">
  <option value="PDF">PDF</option>
  <option value="XPS">XPS</option>
  <option value="DOCX">DOCX</option>
  <option value="DOC">DOC</option>
  <option value="ODT">ODT</option>
  <option value="RTF">RTF</option>
  <option value="TXT">TXT</option>
  <option value="MHT">MHT</option>
  <option value="HTML">HTML</option>
  <option value="XML">XML</option>
  <option value="XLS">XLS</option>
  <option value="XLSX">XLSX</option>
  <option value="CSV">CSV</option>
  <option value="ODS">ODS</option>
  <option value="PPT">PPT</option>
  <option value="PPTX">PPTX</option>
  <option value="ODP">ODP</option>
  <option value="PPS">PPS</option>
  <option value="PPSX">PPSX</option>
</select>
<br/>
<input type="submit" name="submit" value="Convert" />
</form>
```

6. Edit app/controllers/home\_controller.rb and replace it with the following:

```
require Rails.root.to_s + '/lib/DocumentConverterServiceDriver'
require "base64"
class HomeController < ApplicationController
  def upload_file
    *** Get a reference to the uploaded file and check it was specified
    file = params['file']
    if file
      *** Specify the URL of the server that holds the Conversion Service
      url = "http://localhost:41734/Muhimbi.DocumentConverter.WebService/?wsdl"
      conversionClient = DocumentConverterService.new(url)
      *** Create OpenOptions and specify the absolute minimum information
      openOptions = OpenOptions.new()
      openOptions.fileExtension = file.original_filename.split(".").last
      openOptions.originalFileName = file.original_filename
      *** Create ConversionSettings and set the minimum fields.
      conversionSettings = ConversionSettings.new()
      conversionSettings.format = params['outputFormat']
      conversionSettings.fidelity = "Full"
      conversionSettings.openPassword = ""
      conversionSettings.ownerPassword = ""
      *** Encode the source file into a Base64 encoded byte array
      sourceFile = Base64.encode64(file.read)
      *** Carry out the conversion
      convert = Convert.new(sourceFile, openOptions, conversionSettings)
      result = conversionClient.convert(convert)
      *** Send the converted file back to the browser. 'wsdl2ruby' needs
      *** double Base64 decoding for some reason.
      send_data(Base64.decode64(Base64.decode64(result.convertResult)),
        :filename => "convert." + conversionSettings.format,
        :content_type => 'application/octet-stream',
        :disposition => 'attachment')
    end
  end
end
```

Please update the *url* variable with the IP address or DNS name of the server that runs the Muhimbi Conversion Service.

That is it. Start the Rails server as follows:

```
rails s
```

Open a web browser and point it to *http://localhost:3000*. If the browser is opened on a system other than the one that runs the Rails application then replace *localhost* with the DNS name or IP number of that server.

This sample application is very basic. Select a file to convert (please make sure that the file extension matches its format). Then select the Output format, e.g. PDF, and click the *Convert* button.

This is a minimum code sample to illustrate how easy it is to convert a file using Ruby. This Developer Guide contains the entire object model, including details about how to *Convert*, *Watermark*, *Split*, *Merge* and *Secure* files.

### SOAP / Web Service Debugging

The Muhimbi Conversion Service is a Windows Service based on the Microsoft Windows Communication Foundation (WCF) framework. This comprehensive framework is used to expose a standards based Web Services interface that can be consumed by many different platforms including .NET, Java, PHP, SAP, Ruby, Documentum and many others.

Even though WCF Web Services are standards based, standards are not interpreted the same by everyone so from time to time you may need to do some troubleshooting when programming against the PDF Converter Web Service, especially from non-Microsoft platforms.

For details about how to debug Web Service / SOAP messages, see [this Knowledge Base Article](#).

## 4.5 PDF Conversion in PHP

In this section we'll show how to create a simple PHP application to send a file to the PDF Converter. The latest version of this section, including details about how to install PHP on a Windows Server, is [available on our Blog](#).

### Installing the Muhimbi PDF Conversion Services

Using the Muhimbi PDF Conversion Services in combination with PHP requires a standard installation. If PHP is running on the same system as the Muhimbi PDF Converter Services then you can skip steps 2, 3 and 4.

1. Install the *Muhimbi PDF Converter Services* as described in Chapter 2 of the Administration guide.
2. Open *Muhimbi.DocumentConverter.Service.exe.config* in your favourite text editor. A handy shortcut to the configuration / installation folder can be found in the Windows Start Menu Group.
3. Search for *baseAddress* and change *localhost* to the DNS name or IP address of the server running the Conversion Service.
4. Restart the Conversion Service as follows:

```
Net stop "Muhimbi Document Converter Service"  
Net start "Muhimbi Document Converter Service"
```

Please use the included Diagnostics Tool to verify that your installation is correct.

### Generating proxies

Although out-of-the-box PHP comes with a *SoapClient* class to interact with web services, it is much easier and faster to pre-generate proxy classes to talk to the web service.

Many tools are available for generating PHP proxies. The one that we are using in this tutorial is *wsdl2phpgenerator*. Pre-generated proxies are included in the Muhimbi PDF Converter Services' *Sample Code* folder. You can also generate your own proxies using the following steps:

1. [Download wsdl2phpgenerator](#) and unzip it to a location of your choice.
2. Make sure PHP is added to your path (In Windows this is done for you if PHP has been installed using the steps [in this blog post](#)).
3. Open a command prompt and navigate to the location where *wsdl2phpgenerator* was unzipped.
4. Execute the following command to generate the PHP proxies:

```
php wsdl2php.php -s -i  
"http://localhost:41734/Muhimbi.DocumentConverter.WebSe  
rvice/?wsdl" -o documentConverterServices
```

If the Conversion Service is running on a remote machine then please replace 'localhost' with the name of that machine.

5. Copy the newly generated *documentConverterServices.php* file to the folder that holds your PHP code.

### Sample Code

The sample code has been kept as simple as possible and is available from the *Sample Code/PHP* folder in the Conversion Service's installation folder.

Create the following *index.html* file that allows a file to be uploaded and the output file type to be set.

```
<html>
  <body>
    <form action="convert.php" method="post" enctype="multipart/form-data">
      <a href="phpInfo.php">PHP Info</a>
      <br/>
      <label for="file">Document:</label>
      <input type="file" name="file" id="file" />
      <br/>
      <label for="outputFormat">Output format:</label>
      <select name="outputFormat" id="outputFormat">
        <option value="PDF">PDF</option>
        <option value="XPS">XPS</option>
        <option value="DOCX">DOCX</option>
        <option value="DOC">DOC</option>
        <option value="ODT">ODT</option>
        <option value="RTF">RTF</option>
        <option value="TXT">TXT</option>
        <option value="MHT">MHT</option>
        <option value="HTML">HTML</option>
        <option value="XML">XML</option>
        <option value="XLS">XLS</option>
        <option value="XLSX">XLSX</option>
        <option value="CSV">CSV</option>
        <option value="ODS">ODS</option>
        <option value="PPT">PPT</option>
        <option value="PPTX">PPTX</option>
        <option value="ODP">ODP</option>
        <option value="PPS">PPS</option>
        <option value="PPSX">PPSX</option>
      </select>
      <br/>
      <input type="submit" name="submit" value="Convert" />
    </form>
  </body>
</html>
```

The HTML page submits the file to the following PHP file:

```
<?php
// Include the generated proxy classes
require_once "documentConverterServices.php";
// Check the uploaded file
if ($_FILES["file"]["error"] > 0)
```



```
{
    echo "Error uploading file: " . $_FILES["file"]["error"];
}
else
{
    // Get the uploaded file content
    $sourceFile = file_get_contents($_FILES["file"]["tmp_name"]);

    // Create OpenOptions
    $openOptions = new OpenOptions();
    // set file name and extension
    $openOptions->FileExtension = pathinfo($_FILES["file"]["name"],
                                           PATHINFO_EXTENSION);
    $openOptions->OriginalFileName = $_FILES["file"]["name"];
    // Create conversionSettings
    $conversionSettings = new ConversionSettings();
    // Set the output format
    if(isset($_POST["outputFormat"]))
    {
        $conversionSettings->Format = $_POST["outputFormat"];
    } else {
        $conversionSettings->Format = "PDF";
    }
    // Set fidelity
    $conversionSettings->Fidelity = "Full";
    // These values must be set to empty strings or actual passwords when
    // converting to non PDF formats
    $conversionSettings->OpenPassword="";
    $conversionSettings->OwnerPassword="";
    // Set some of the other conversion settings.
    // Completely optional and just an example
    $conversionSettings->StartPage = 0;
    $conversionSettings->EndPage = 0;
    $conversionSettings->Range = "VisibleDocuments";
    $conversionSettings->Quality = "OptimizeForPrint";
    $conversionSettings->PDFProfile = "PDF_1_5";
    $conversionSettings->GenerateBookmarks = "Automatic";
    $conversionSettings->PageOrientation="Default";
    // Create the Convert parameter that is send to the server
    $convert = new Convert($sourceFile, $openOptions, $conversionSettings);
    // Create the service client and point it to the correct Conversion Service
    $url = "http://localhost:41734/Muhimbi.DocumentConverter.WebService/?wsdl";
    $serviceClient = new DocumentConverterService(array(), $url);

    // If you are expecting long running operations then consider longer timeouts
    ini_set('default_socket_timeout', 60);

    try
    {
        // Execute the web service call
        $result = $serviceClient->Convert($convert)->ConvertResult;
        // Send the resulting file to the client.
        header("Cache-Control: must-revalidate, post-check=0, pre-check=0");
        header("Content-type: application/octet-stream");
        header("Content-Disposition: attachment; filename=\"convert.\" .
              $conversionSettings->Format . \"\");
        echo $result;
    }
    catch (Exception $e)
    {
        print "Error converting document: " . $e->getMessage();
    }
}
?>
```

Place all files in the same folder under your web server root. Open a web browser and point it to *index.html*. Select a file, specify the output format and click 'Convert'.

This is just a simple example. The full object model, including details about merging, splitting and watermarking files, is available in this Developer Guide.

If you expect to execute long running operations then you may want to read-up on dealing with PHP socket timeouts:

- <http://stackoverflow.com/questions/3500527/php-soapclient-timeout>
- <http://stackoverflow.com/questions/835184/handling-soap-timeouts-in-php>

### Troubleshooting

If you are experiencing any problems with this sample code, e.g. warning messages such as *Missing argument 1 for OpenOptions* then please read the comment dated 15 November, 2013 09:12 [at the end of this post](#).

### SOAP / Web Service Debugging

The Muhimbi Conversion Service is a Windows Service based on the Microsoft Windows Communication Foundation (WCF) framework. This comprehensive framework is used to expose a standards based Web Services interface that can be consumed by many different platforms including .NET, Java, PHP, SAP, Ruby, Documentum and many others.

Even though WCF Web Services are standards based, standards are not interpreted the same by everyone so from time to time you may need to do some troubleshooting when programming against the PDF Converter Web Service, especially from non-Microsoft platforms.

For details about how to debug Web Service / SOAP messages, see [this Knowledge Base Article](#).

### 4.6 Cross-Converting between document types

Although the product name refers to PDF Conversion, as of version 6.0 it is also possible to cross convert between document types, e.g. doc to docx, xlsx to xls and even xls to doc.

So, how is this useful? Well, let's say that you have a large amount of legacy Office 97-2003 files, but your company now requires all files to be saved in the more modern, and open, Office 2007-2010 formats. By using the Muhimbi PDF Converter you can convert between these formats automatically using simple web service call using Java or .NET.

Conversion in the other direction is possible as well. For example many users in an organisation may still be on Office 2000 or 2003, but those fancy guys in IT are saving documents in Office 2010 format, which no-one else can open. A simple application will automatically take care of this and convert all files to the desired format.

Naturally some thought needs to be given to what file formats to convert between. Converting between AutoCAD and Excel makes little sense, but from Excel to Word and Word to Excel could be useful. The table listed below shows which file formats can be converted between.

Input

		Output																		
		pdf	xps	doc	docx	rtf	txt	html	mht	xml	odt	xls	xlsx	csv	ods	ppt	pptx	odp	pps	ppsx
Word Processing	doc, docx, docm, rtf, txt, wps, xml, eml, odt, ott, mht																			
InfoPath	xml																			
Spreadsheets	xls, xlsx, xslm, xlsb, xml, csv, dif, ods, ots																			
Presentations	ppt, pptx, pptm, xml, odp, otp, pps, ppsx, ppsm																			
Publisher	pub																			
Vector	vsd, vdx, vdw, svg, svgz																			
CAD	dxf, dwg																			
HTML	htm, html, mht																			
Images	gif, jpg, png, bmp, tif																			
Email	msg																			

Only works with HTML / MHT files, not with URLs

Active sheet only

Some points of interest:

1. It is now possible to convert InfoPath files to MS-Word, Excel and HTML. For details see section 4.6.2.
2. Although not displayed in this chart, it is also possible to convert PDF (and any other file type) to PDF/A. For details see *Appendix - Post processing PDF output to PDF/A* in the Administration Guide.
3. It is even possible to 'convert' to the same format as the source, e.g. docx to docx, but specify additional settings such as a password on the document.

#### 4.6.1 Cross-Converting file types using a Web Service call

Converting files to non-PDF formats using web service calls works identical to converting files to PDF. The only difference is that the *Format* property on the *ConversionSettings* object must be set to the file type you are converting to. For details see the existing Convert to PDF sample code in chapters 4.1 and 4.2.

#### 4.6.2 Convert InfoPath to MS-Word, Excel, XPS and PDF

The PDF Converter's cross-conversion facility opens up a whole new world of possibilities such as converting between DOC and DOCX, XLS and XLSX, but more importantly it also supports conversion between completely different document types such as Excel to MS-Word and HTML to Excel.

This section describes another new conversion type that should be of particular interest to InfoPath users as it is now possible to convert InfoPath forms to MS-Word, Excel and HTML.

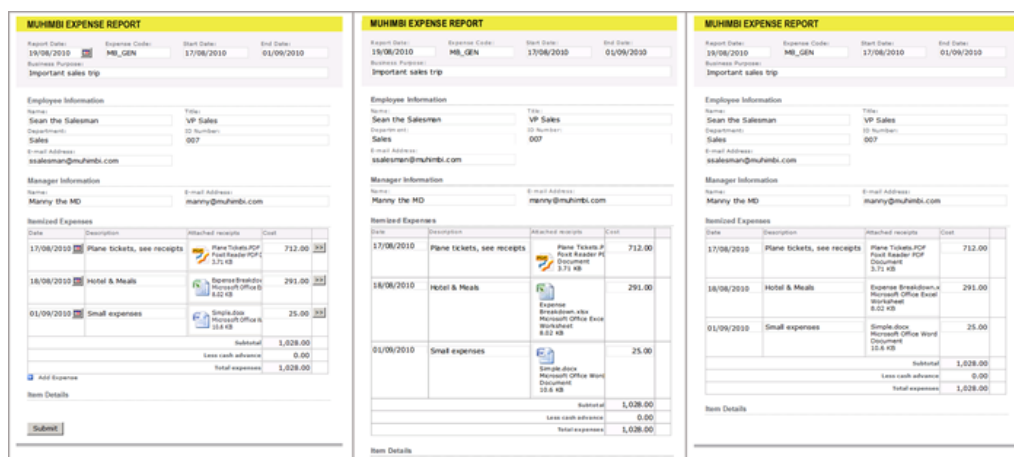
Conversion to these new formats generally works very well, but there are some limitations due to the nature of these non-PDF based destination formats. Specifically:

1. **Attachments:** When converting an InfoPath form to PDF the software also converts all attachments and merges them into the main PDF. This is possible because you can represent almost any file format in PDF and merge them together. Unfortunately this is not possible when converting to HTML, MS-Word or Excel.
2. **View Selection:** The software provides a number of ways to specify which view or views to convert (See chapter 4.12 *Controlling which InfoPath views to Export to PDF*). When converting to PDF it is possible to specify multiple views, which the converter then merges together into a single document. When converting to HTML, MS-Word or Excel it is only possible to convert a single view as these file formats don't support merging. As a workaround it is possible to create a 'conversion specific view' and combine the content of multiple views in it.  
  
Print Views are also ignored when converting to HTML, Word or Excel. Instead you will need to use Muhimbi's View Selection facilities if you wish to convert any view other than the default View.
3. **Formatting:** PDF is a very flexible format that allows any content to be placed anywhere on the page. MS-Word, Excel and HTML are not necessarily this flexible. For example, Excel uses a 'cell based approach' to display content. If an InfoPath form is not specifically designed for export to Excel, e.g. it uses nested tables or different column widths across a page, then you may need to optimise your InfoPath form for conversion, or create a 'conversion specific view'.

Some hints and tips related to converting to the various non-PDF formats can be found below.

## InfoPath to HTML (MHT)

When converting InfoPath to HTML the resulting file is a self contained MHT file that most modern browsers can display. All information including images, HTML and style sheets are included in this single file.

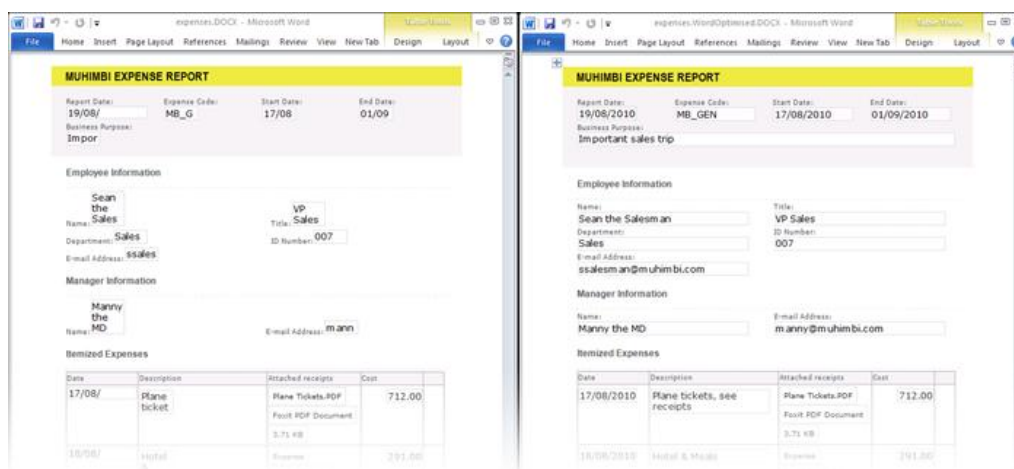


From left to right, the same Form in InfoPath, converted to PDF and converted to HTML

As this image shows, InfoPath data can be represented in HTML really well so it is usually not needed to make any changes to the XSN file.

## InfoPath to MS-Word

Depending on how an InfoPath form has been designed, some work may be required to make things look better when converting to MS-Word. This is mainly due to the fact that MS-Word does not like dimensions that are expressed in percentages, while it is common in InfoPath to create a table grid and populate that grid with controls that take up 100% of the available cell space.



Results when converted to MS-Word before optimisation (left) and afterwards (right).

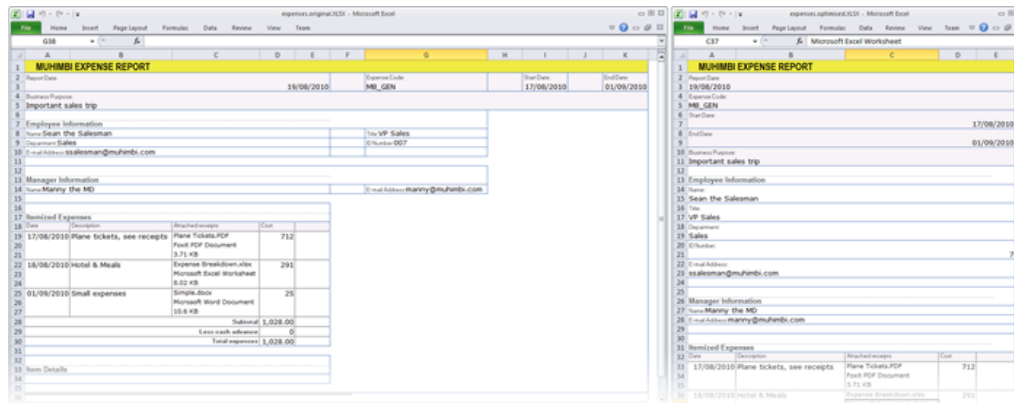
Looking at the 'before optimisation' conversion results in the image displayed above, there are 2 things that stand out:

1. **Dimension of text fields:** The dimensions of most text fields are not quite right. This can easily be changed by opening the form in InfoPath Designer and changing the width of the various fields from '100%' to the actual dimensions in cm or inches.
2. **Missing 'year' in date picker fields:** Due the way the Date Picker is structured internally, modifying its width does not translate properly when displayed in MS-Word. To solve this, change the date picker field to a regular text field either by creating a conversion specific view, or using a display rule.

The InfoPath to MS-Word facility can generate output in *doc*, *docx*, *rtf*, *txt*, *html* and *odt* formats.

### InfoPath to Excel

InfoPath to Excel conversion for existing forms that are not optimised for conversion to Excel are probably the trickiest ones to get right. If the 'look and feel' of the Excel sheet is not important then no change is required. However, if the Excel forms need to 'look good' then you may need to rethink the way the form is designed.



Results when converted to Excel before optimisation (left) and afterwards (right).

Looking at the 'before optimisation' in the image above things don't look too bad, but clearly it is not the same as the original. The main issues are as follows:

1. **Column Widths:** As Excel uses a cell / grid based approach it is not possible to mix different column widths. The information in the form's header requires different column width and spans than the columns used in the repeating table further down the page. By changing the horizontally oriented fields in the header to individual rows we no longer have this problem.

2. **Number formats:** Depending on a cell's content, Excel sometimes tries to be 'clever'. Most of the time this works great, but in this case a field with value '007' is changed into a '7'. This could be fixed by changing the content of the InfoPath field into a formula and concatenating an apostrophe in front of it.

The InfoPath to Excel facility can generate output in *xls*, *xlsx*, *csv* and *ods* format.



## 4.7 Merging multiple files into a single PDF using .NET

The following example describes the steps needed to convert all files in a directory, merge the results into a single file and apply page numbering to the merged file using the built in watermarking engine. We are using Visual Studio and C#, but any environment that can invoke web services should be able to access this functionality. Note that the WSDL can be found at <http://localhost:41734/Muhimbi.DocumentConverter.WebService/?wsdl>.

1. Start a new Visual Studio project and create the project type of your choice. In this example we are using a standard *.net 3.0* project of type *Windows Forms Application*. Name it 'Simple PDF Converter Sample'.
2. Add a *TextBox* and *Button* control to the form. Accept the default names of *textBox1* and *button1*.
3. In the *Solution Explorer* window, right-click *References* and select *Add Service Reference*. (Do not use web references!)
4. In the *Address* box enter the WSDL address listed in the introduction of this section. If the Conversion Service is located on a different machine then substitute *localhost* with the server's name.
5. Accept the default Namespace of *ServiceReference1* and click the *OK* button to generate the proxy classes.
6. Double click *Button1* and replace the content of the entire code file with the following:

```
using System;
using System.Collections.Generic;
using System.IO;
using System.ServiceModel;
using System.Windows.Forms;
using Simple_PDF_Converter_Sample.ServiceReference1;

namespace Simple_PDF_Converter_Sample
{
    public partial class Form1 : Form
    {
        // ** The URL where the Web Service is located. Amend host name if needed.
        string SERVICE_URL = "http://localhost:41734/Muhimbi.DocumentConverter.WebService/";

        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            DocumentConverterServiceClient client = null;

            try
            {
                // ** Options and all settings for batch conversion
                ProcessingOptions processingOptions = new ProcessingOptions();

                // ** Specify the minimum level of merge settings
                MergeSettings mergeSettings = new MergeSettings();
                mergeSettings.BreakOnError = false;
                mergeSettings.Watermarks = CreateWatermarks();
            }
        }
    }
}
```



```

processingOptions.MergeSettings = mergeSettings;

// ** Get all files in the folder
string sourceFolder = textBox1.Text;
string[] sourceFileNames = Directory.GetFiles(sourceFolder);

// ** Iterate over all files and create a list of SourceFile Objects
List<SourceFile> sourceFiles = new List<SourceFile>();
foreach (string sourceFileName in sourceFileNames)
{
    // ** Read the contents of the file
    byte[] sourceFileContent = File.ReadAllBytes(sourceFileName);

    // ** Set the absolute minimum open options
    OpenOptions openOptions = new OpenOptions();
    openOptions.OriginalFileName = Path.GetFileName(sourceFileName);
    openOptions.FileExtension = Path.GetExtension(sourceFileName);

    // ** Set the absolute minimum conversion settings.
    ConversionSettings conversionSettings = new ConversionSettings();
    conversionSettings.Fidelity = ConversionFidelities.Full;
    conversionSettings.Quality = ConversionQuality.OptimizeForPrint;

    // ** Create merge settings for each file and set name for the PDF bookmark
    FileMergeSettings fileMergeSettings = new FileMergeSettings();
    fileMergeSettings.TopLevelBookmark = openOptions.OriginalFileName;

    // ** Create a source file object and add it to the list
    SourceFile sourceFile = new SourceFile();
    sourceFile.OpenOptions = openOptions;
    sourceFile.ConversionSettings = conversionSettings;
    sourceFile.MergeSettings = fileMergeSettings;
    sourceFile.File = sourceFileContent;
    sourceFiles.Add(sourceFile);
}

// ** Assign source files
processingOptions.SourceFiles = sourceFiles.ToArray();

// ** Open the service and configure the bindings
client = OpenService(SERVICE_URL);
// ** Carry out the merge process
BatchResults results = client.ProcessBatch(processingOptions);
// ** Read the results of the merged file.
byte[] mergedFile = results.Results[0].File;

// ** Write the converted file back using the name of the folder
string folderName = new DirectoryInfo(sourceFolder).Name;
DirectoryInfo parentFolder = Directory.GetParent(sourceFolder);
string destinationFileName = Path.Combine(parentFolder.FullName,
                                           folderName + ".pdf");
using (FileStream fs = File.Create(destinationFileName))
{
    fs.Write(mergedFile, 0, mergedFile.Length);
    fs.Close();
}

MessageBox.Show("Contents of directory merged to " + destinationFileName);
}
catch (FaultException<WebServiceFaultException> ex)
{
    MessageBox.Show("FaultException occurred: ExceptionType: " +
                    ex.Detail.ExceptionType.ToString());
}
catch (Exception ex)
{
    MessageBox.Show(ex.ToString());
}
finally
{
    CloseService(client);
}

```

```

    }
}

/// <summary>
/// Configure the Bindings, endpoints and open the service using the specified address.
/// </summary>
/// <returns>An instance of the Web Service.</returns>
public static DocumentConverterServiceClient OpenService(string address)
{
    DocumentConverterServiceClient client = null;

    try
    {
        BasicHttpBinding binding = new BasicHttpBinding();
        // ** Use standard Windows Security.
        binding.Security.Mode = BasicHttpSecurityMode.TransportCredentialOnly;
        binding.Security.Transport.ClientCredentialType =
            HttpClientCredentialType.Windows;
        // ** Increase the Timeout to deal with (very) long running requests.
        binding.SendTimeout = TimeSpan.FromMinutes(30);
        binding.ReceiveTimeout = TimeSpan.FromMinutes(30);
        // ** Set the maximum document size to 40MB
        binding.MaxReceivedMessageSize = 50 * 1024 * 1024;
        binding.ReaderQuotas.MaxArrayLength = 50 * 1024 * 1024;
        binding.ReaderQuotas.MaxStringLength = 50 * 1024 * 1024;

        // ** Specify an identity (any identity) in order to get it past .net3.5 sp1
        EndpointIdentity epi = EndpointIdentity.CreateUpnIdentity("unknown");
        EndpointAddress epa = new EndpointAddress(new Uri(address), epi);

        client = new DocumentConverterServiceClient(binding, epa);
        client.Open();
        return client;
    }
    catch (Exception)
    {
        CloseService(client);
        throw;
    }
}

/// <summary>
/// Check if the client is open and then close it.
/// </summary>
/// <param name="client">The client to close</param>
public static void CloseService(DocumentConverterServiceClient client)
{
    if (client != null && client.State == CommunicationState.Opened)
        client.Close();
}

/// <summary>
/// This method creates watermarks for applying page numbers
/// </summary>
/// <returns>Array of watermarks</returns>
private Watermark[] CreateWatermarks()
{
    // ** Create watermark container
    Watermark pageWatermark = new Watermark();
    // ** Set positioning to the lower right of the page
    pageWatermark.HPosition = HPosition.Right;
    pageWatermark.VPosition = VPosition.Bottom;
    // ** Set size
    pageWatermark.Width = "200";
    pageWatermark.Height = "20";

    // ** Create text object for the page numbering
    Text oddPageText = new Text();
    // ** No need to position the element in the watermark container

```

```

oddPageText.Width = "200";
oddPageText.Height = "20";
// ** set content including field codes
oddPageText.Content = "Page {PAGE} of {NUMPAGES}";
// ** set font properties
oddPageText.FillColor = "#ffff0000";
oddPageText.FontFamilyName = "Verdana";
oddPageText.FontSize = "10";
oddPageText.FontStyle = FontStyle.Regular;
/* set text alignment
oddPageText.HAlign = HAlign.Right;
oddPageText.VAlign = VAlign.Top;

/** create array of watermark elements
Element[] pageWatermarkElements = new Element[] { oddPageText };

/** set elements of watermark
pageWatermark.Elements = pageWatermarkElements;

/* return array of watermarks
return new Watermark[] { pageWatermark };
    }
}
}

```

Providing the project and all controls are named as per the steps above, it should compile without errors. Run it, enter the full path to a folder that holds a couple of text files (PDF, Word, Excel, etc) and click the button to start the convert and merge process. The operation may take a while depending on the number and complexity of files in the folder.

Note that in this example we are programmatically configuring the WCF Bindings and End Points. If you wish you can use a declarative approach using the config file. For more information about working with WCF see [http://msdn.microsoft.com/en-us/library/ms735119\(v=VS.90\).aspx](http://msdn.microsoft.com/en-us/library/ms735119(v=VS.90).aspx)

A more complex and full featured sample application is installed, with full source code, alongside the Conversion Service.

## 4.8 Merging multiple files into a single PDF using Java

The following sample merges all files specified on the command line into a single PDF. If the source files are not already in PDF format then it automatically converts them in the process. A PDF bookmark is automatically generated for each merged file as well.

The full version of the sample code discussed in this chapter, including pre generated proxies, is installed alongside each copy of the product. Use the Start Menu to open the appropriate folder. The latest version of this tutorial can be found on-line at <http://blog.muhimbi.com/2011/12/converting-and-merging-multiple-files.html>.

For details about setting up all Java prerequisites as well as using *wsimport* to generate Java proxies for the web service see section 4.2 *PDF Conversion in Java*.

Once the proxy classes have been created add the following code to your project. Run the code and make sure the paths to multiple documents to convert and merge are specified on the command line.

```
package com.muhimbi.app;

import com.muhimbi.ws.*;
import java.io.*;
import java.net.URL;
import javax.xml.bind.JAXBElement;
import javax.xml.namespace.QName;

public class WsClient {

    private final static String DOCUMENTCONVERTERSERVICE_WSDL_LOCATION =
        "http://localhost:41734/Muhimbi.DocumentConverter.WebService/?wsdl";

    private static ObjectFactory _objectFactory = new ObjectFactory();

    public static void main(String[] args) {
        try {
            if (args.length == 0) {
                System.out
                    .println("Please specify one or more file names to convert and merge.");
            } else {
                System.out.println("Merging files");

                // ** Initialise Web Service
                DocumentConverterService_Service dcss = new DocumentConverterService_Service(
                    new URL(DOCUMENTCONVERTERSERVICE_WSDL_LOCATION),
                    new QName("http://tempuri.org/", "DocumentConverterService"));
                DocumentConverterService dcs = dcss.getBasicHttpBindingDocumentConverterService();

                // ** Get the options for all files that need to be merged
                ProcessingOptions processingOptions = getProcessingOptions(args);

                // ** Carry out the merging (and converting if needed)
                BatchResults results = dcs.processBatch(processingOptions);

                // ** Get the content of the first file, which holds the merged file in the byte array
                byte[] convertedFile =

                    results.getResults().getValue().getBatchResult().get(0).getFile().getValue();

                // ** Write converted file to file system
                writeFile(convertedFile, "merged.pdf");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```

        System.out.println("Files merged into 'merged.pdf'");
    }

    } catch (IOException e) {
        System.out.println(e.getMessage());
    } catch (DocumentConverterServiceProcessBatchWebServiceFaultExceptionFaultFaultMessage e) {
        printException(e.getFaultInfo());
    }
}

public static ProcessingOptions getProcessingOptions(String[] sourceFileNames) throws
IOException
{
    // ** Options and all settings for batch conversion
    ProcessingOptions processingOptions = new ProcessingOptions();

    // ** Specify the minimum level of merge settings, optionally add watermarks and security
    MergeSettings mergeSettings = new MergeSettings();
    mergeSettings.setBreakOnError(false);
    processingOptions.setMergeSettings(
        _objectFactory.createProcessingOptionsMergeSettings( mergeSettings ));

    // ** Create an array of files to merge
    ArrayOfSourceFile sourceFiles = new ArrayOfSourceFile();
    for(int i =0; i<sourceFileNames.length; i++)
    {
        SourceFile sourceFile = getSourceFile(sourceFileNames[i]);
        sourceFiles.getSourceFile().add(sourceFile);
    }

    processingOptions.setSourceFiles(
        _objectFactory.createProcessingOptionsSourceFiles(sourceFiles));
    return processingOptions;
}

public static SourceFile getSourceFile(String fileName) throws IOException
{
    File file = new File(fileName);

    // ** Read the contents of the file
    System.out.println("- Reading: " + fileName);
    byte[] sourceFileContent = readFile(fileName);

    // ** Set the absolute minimum open options
    OpenOptions openOptions = getOpenOptions(getFileName(file), getFileExtension(file) );

    // ** Set the absolute minimum conversion settings.
    ConversionSettings conversionSettings = getConversionSettings();

    // ** Create merge settings for each file and set the name for the PDF bookmark
    FileMergeSettings fileMergeSettings = new FileMergeSettings();
    fileMergeSettings.setTopLevelBookmark(
        _objectFactory.createFileMergeSettingsTopLevelBookmark( file.getName() ));

    // ** Create a source file object and return it
    SourceFile sourceFile = new SourceFile();
    sourceFile.setOpenOptions(_objectFactory.createSourceFileOpenOptions(openOptions));
    sourceFile.setConversionSettings(
        _objectFactory.createSourceFileConversionSettings(conversionSettings));
    sourceFile.setMergeSettings(_objectFactory.createSourceFileMergeSettings(fileMergeSettings));
    sourceFile.setFile(_objectFactory.createSourceFileFile(sourceFileContent));
    return sourceFile;
}

public static OpenOptions getOpenOptions(String fileName, String fileExtension) {
    OpenOptions openOptions = new OpenOptions();
    // ** Set the minimum required open options. Additional options are available
    openOptions.setOriginalFileName(_objectFactory.createOpenOptionsOriginalFileName(fileName));
    openOptions.setFileExtension(_objectFactory.createOpenOptionsFileExtension(fileExtension));
}

```

```

    return openOptions;
}

public static ConversionSettings getConversionSettings() {
    ConversionSettings conversionSettings = new ConversionSettings();
    // ** Set the minimum required conversion settings. Additional settings are available
    conversionSettings.setQuality(ConversionQuality.OPTIMIZE_FOR_PRINT);
    conversionSettings.setRange(ConversionRange.ALL_DOCUMENTS);
    conversionSettings.getFidelity().add("Full");
    conversionSettings.setFormat(OutputFormat.PDF);
    return conversionSettings;
}

public static String getFileName(File file) {
    String fileName = file.getName();
    return fileName;
}

public static String getFileExtension(File file) {
    String fileName = file.getName();
    return fileName.substring(fileName.lastIndexOf('.') + 1, fileName.length());
}

public static byte[] readFile(String filepath) throws IOException {
    File file = new File(filepath);
    InputStream is = new FileInputStream(file);
    long length = file.length();
    byte[] bytes = new byte[(int) length];

    int offset = 0;
    int numRead;
    while (offset < bytes.length
        && (numRead = is.read(bytes, offset, bytes.length - offset)) >= 0) {
        offset += numRead;
    }

    if (offset < bytes.length) {
        throw new IOException("Could not completely read file " + file.getName());
    }
    is.close();
    return bytes;
}

public static void writeFile(byte[] fileContent, String filepath)
    throws IOException {
    OutputStream os = new FileOutputStream(filepath);
    os.write(fileContent);
    os.close();
}

public static void printException(WebServiceFaultException serviceFaultException) {
    System.out.println(serviceFaultException.getExceptionType());
    JAXBElement<ArrayOfstring> element = serviceFaultException.getExceptionDetails();
    ArrayOfstring value = element.getValue();
    for (String msg : value.getString()) {
        System.out.println(msg);
    }
}
}

```

## 4.9 Splitting PDFs into multiple documents

The following sample describes the steps needed to split up a single PDF file based on the number of pages. We are using Visual Studio and C#, but any environment that can invoke web services should be able to access this functionality. Note that the WSDL can be found at <http://localhost:41734/Muhimbi.DocumentConverter.WebService/?wsdl>.

The source code for this example can be found in the folder the Muhimbi Conversion service has been installed to. For more details see <http://blog.muhimbi.com/2011/10/splitting-pdf-files-using-pdf-converter.html>.

1. Start a new Visual Studio project and create the project type of your choice. In this example we are using a standard *.net 3.0* project of type *Console Application*. Name it 'Split PDF'.
2. In the *Solution Explorer* window, right-click *References* and select *Add Service Reference*. (Do not use web references!)
3. In the *Address* box enter the WSDL address listed in the introduction of this section. If the Conversion Service is located on a different machine then substitute *localhost* with the server's name.
4. Accept the default Namespace of *ServiceReference1* and click the *OK* button to generate the proxy classes.
5. Optionally add a PDF file to the solution, set the *Build Action* to *None* and *Copy to Output Directory* to *Copy if newer*. By doing this there will always be a valid test file in the same directory as the compiled executable.
6. Copy and paste the following code and replace the contents of *Program.cs*.

```
using System;
using System.IO;
using System.ServiceModel;
using Split_PDF.ServiceReference1;

namespace Split_PDF
{
    class Program
    {
        // ** The URL where the Web Service is located. Amend host name if needed.
        static string SERVICE_URL = "http://localhost:41734/Muhimbi.DocumentConverter.WebService/";

        static void Main(string[] args)
        {
            DocumentConverterServiceClient client = null;
            try
            {
                // ** Determine the source file and read it into a byte array.
                string sourceFileName = null;
                if (args.Length == 0)
                {
                    /** Delete any split files from a previous test run.
                    foreach (string file in Directory.GetFiles(Directory.GetCurrentDirectory(),
                                                                "spf-*.pdf"))
                    {
                        File.Delete(file);
                    }

                    // ** If nothing is specified then read the first PDF file.
                    string[] sourceFiles = Directory.GetFiles(Directory.GetCurrentDirectory(),
```

```

        "*.pdf");
    if (sourceFiles.Length > 0)
        sourceFileName = sourceFiles[0];
    else
    {
        Console.WriteLine("Please specify a document to split.");
        Console.ReadKey();
        return;
    }
}
else
    sourceFileName = args[0];

byte[] sourceFile = File.ReadAllBytes(sourceFileName);

// ** Open the service and configure the bindings
client = OpenService(SERVICE_URL);

/** Set the absolute minimum open options
OpenOptions openOptions = new OpenOptions();
openOptions.OriginalFileName = Path.GetFileName(sourceFileName);
openOptions.FileExtension = "pdf";

// ** Set the absolute minimum conversion settings.
ConversionSettings conversionSettings = new ConversionSettings();

// ** Create the ProcessingOptions for the splitting task.
ProcessingOptions processingOptions = new ProcessingOptions()
{
    MergeSettings = null,
    SplitOptions = new FileSplitOptions()
    {
        FileNameTemplate = "spf-{0:D3}",
        FileSplitType = FileSplitType.ByNumberOfPages,
        BatchSize = 5,
        BookmarkLevel = 0
    },
    SourceFiles = new SourceFile[1]
    {
        new SourceFile()
        {
            MergeSettings = null,
            OpenOptions = openOptions,
            ConversionSettings = conversionSettings,
            File = sourceFile
        }
    }
};

// ** Carry out the splitting.
Console.WriteLine("Splitting file " + sourceFileName);
BatchResults batchResults = client.ProcessBatch(processingOptions);

// ** Process the returned files
foreach (BatchResult result in batchResults.Results)
{
    Console.WriteLine("Writing split file " + result.FileName);
    File.WriteAllBytes(result.FileName, result.File);
}

Console.WriteLine("Finished.");
}
catch (FaultException<WebServiceFaultException> ex)
{
    Console.WriteLine("FaultException occurred: ExceptionType: " +
        ex.Detail.ExceptionType.ToString());
}
catch (Exception ex)
{
    Console.WriteLine(ex.ToString());
}

```



```

        finally
        {
            CloseService(client);
        }
        Console.ReadKey();
    }

    /// <summary>
    /// Configure the Bindings, endpoints and open the service using the specified address.
    /// </summary>
    /// <returns>An instance of the Web Service.</returns>
    public static DocumentConverterServiceClient OpenService(string address)
    {
        DocumentConverterServiceClient client = null;

        try
        {
            BasicHttpBinding binding = new BasicHttpBinding();
            // ** Use standard Windows Security.
            binding.Security.Mode = BasicHttpSecurityMode.TransportCredentialOnly;
            binding.Security.Transport.ClientCredentialType =
                HttpClientCredentialType.Windows;
            // ** Increase the client Timeout to deal with (very) long running requests.
            binding.SendTimeout = TimeSpan.FromMinutes(30);
            binding.ReceiveTimeout = TimeSpan.FromMinutes(30);
            // ** Set the maximum document size to 50MB
            binding.MaxReceivedMessageSize = 50 * 1024 * 1024;
            binding.ReaderQuotas.MaxArrayLength = 50 * 1024 * 1024;
            binding.ReaderQuotas.MaxStringLength = 50 * 1024 * 1024;

            // ** Specify an identity (any identity) in order to get it past .net3.5 sp1
            EndpointIdentity epi = EndpointIdentity.CreateUpnIdentity("unknown");
            EndpointAddress epa = new EndpointAddress(new Uri(address), epi);

            client = new DocumentConverterServiceClient(binding, epa);
            client.Open();

            return client;
        }
        catch (Exception)
        {
            CloseService(client);
            throw;
        }
    }

    /// <summary>
    /// Check if the client is open and then close it.
    /// </summary>
    /// <param name="client">The client to close</param>
    public static void CloseService(DocumentConverterServiceClient client)
    {
        if (client != null && client.State == CommunicationState.Opened)
            client.Close();
    }
}

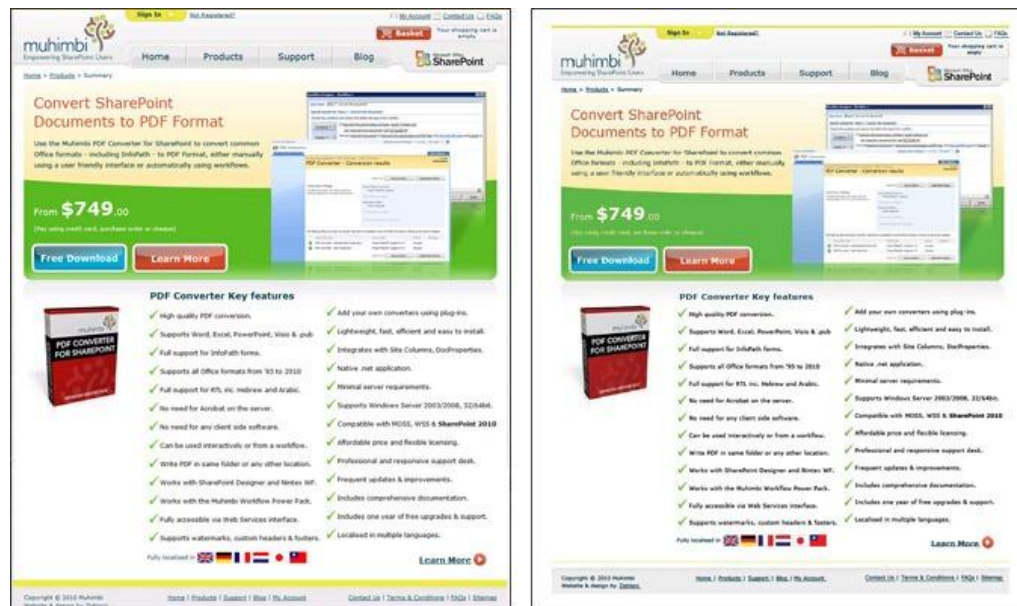
```

Compile the application and run it either from the command prompt, with a path to the PDF file to split on the command line, or – if a PDF file is present in the executable's folder – just run it.

Note that in this example we are programmatically configuring the WCF Bindings and End Points. If you wish you can use a declarative approach using the config file. For more information about working with WCF see [http://msdn.microsoft.com/en-us/library/ms735119\(v=VS.90\).aspx](http://msdn.microsoft.com/en-us/library/ms735119(v=VS.90).aspx)

## 4.10 Converting HTML / web pages using a Web Service call

HTML to PDF Conversion is accessible via the web services based interface as well. The latest version of this tutorial is available on the Muhimbi Blog at <http://blog.muhimbi.com/2010/08/convert-sharepoint-html-pages-to-pdf.html>



Example of the original web page (left) and the converted PDF file (right)

Listed below is a simple C# example of how to carry out a conversion from your own code. The sample code is not complete as it calls into some shared functions from our [main C# example](#) to keep things short.

Our existing [Java based examples](#) can easily be extended to carry out the same type of conversions.

```
/// <summary>
/// Simple sample to convert either a URL or HTML code fragment to PDF format
/// </summary>
/// <param name="htmlOnly">A flag indicating if an HTML Code fragment (true)
/// or URL (false) should be converted.</param>
private void ConvertHTML(bool htmlOnly)
{
    DocumentConverterServiceClient client = null;

    try
    {
        string sourceFileName = null;
        byte[] sourceFile = null;

        client = OpenService("http://localhost:41734/Muhimbi.DocumentConverter.WebService/");

        OpenOptions openOptions = new OpenOptions();

        /** Specify optional authentication settings for the web page
        openOptions.UserName = "";
        openOptions.Password = "";
```

```

if (htmlOnly == true)
{
    /** Specify the HTML to convert
    sourceFile = System.Text.Encoding.UTF8.GetBytes("Hello <b>world</b>");
}
else
{
    // ** Specify the URL to convert
    openOptions.OriginalFileName = "http://www.muhimbi.com/";
}
openOptions.FileExtension = "html";
/** Generate a temp file name that is later used to write the PDF to
sourceFileName = Path.GetTempFileName();
File.Delete(sourceFileName);

// ** Enable JavaScript on the page to convert.
openOptions.AllowMacros = MacroSecurityOption.All;

// ** Set the various conversion settings
ConversionSettings conversionSettings = new ConversionSettings();
conversionSettings.Fidelity = ConversionFidelities.Full;
conversionSettings.PDFProfile = PDFProfile.PDF_1_5;
conversionSettings.PageOrientation = PageOrientation.Portrait;
conversionSettings.Quality = ConversionQuality.OptimizeForPrint;

// ** Carry out the actual conversion
byte[] convertedFile = client.Convert(sourceFile, openOptions, conversionSettings);

// ** Write the PDF file to the local file system.
string destinationFileName = Path.GetDirectoryName(sourceFileName) + @"\" +
                             Path.GetFileNameWithoutExtension(sourceFileName) +
                             "." + conversionSettings.Format;

using (FileStream fs = File.Create(destinationFileName))
{
    fs.Write(convertedFile, 0, convertedFile.Length);
    fs.Close();
}

// ** Display the converted file in a PDF viewer.
NavigateBrowser(destinationFileName);
}
finally
{
    CloseService(client);
}
}

```

Both C# and Java based sample code is available from the Windows Start menu as well.

## 4.10.1 Inserting Page Breaks when converting HTML to PDF

The Muhimbi PDF Converter supports HTML page breaks using the standard 'page-break-after' CSS syntax. For example:

```

<html><body>
  <div style="page-break-after:always">Page 1</div>
  <div style="page-break-after:always">Page 2</div>
</body></html>

```

## 4.11 Converting PDF to PDF/A1b or PDF/A2b

As of version 5.2 of the product, the Muhimbi PDF converter allows PDF files to be post processed for output as PDF/A. This does require some configuration changes, which are outlined in the Administration Guide under *Appendix – Post processing PDF output to PDF/A*.

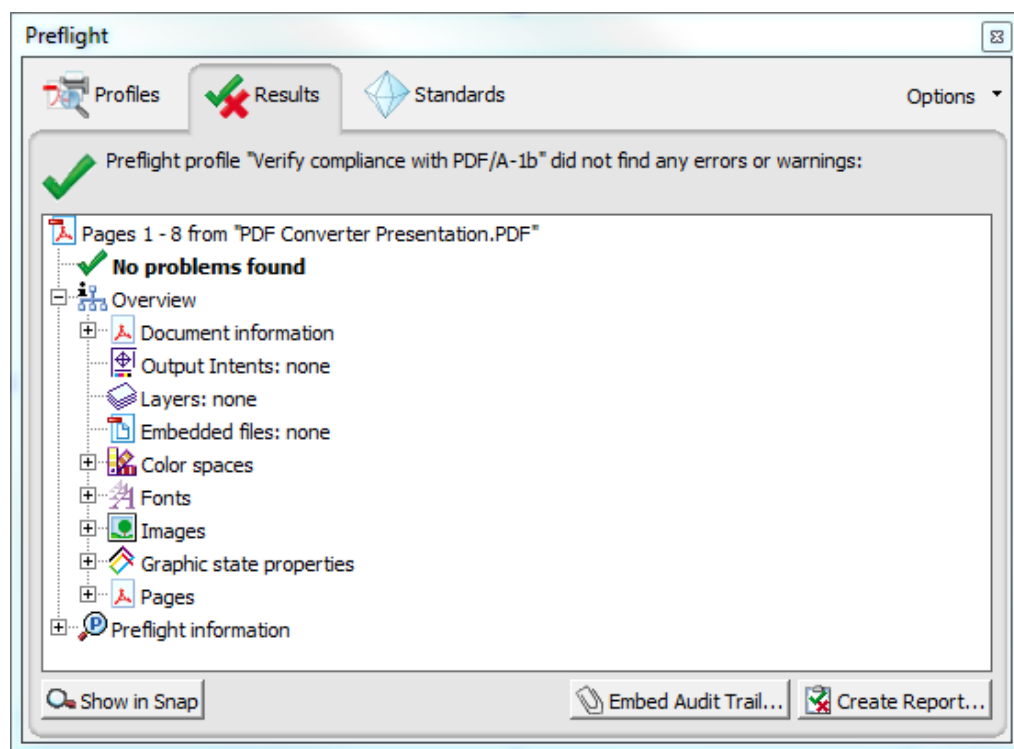
The on-line equivalent of this section can be found in the following blog post <http://blog.muhimbi.com/2011/09/converting-pdf-document-to-pdfa1b-using.html>

In this section we'll provide a simple .NET sample that invokes our Web Services interface to carry out the conversion from PDF to PDF/A1b. The code is nearly identical to the code to convert and watermark a simple MS-Word file (see 5.1) with the following exceptions.

1. *openOptions.FileExtension* is set to *pdf*.
2. *conversionSettings.PDFProfile* is set to *PDFProfile.PDF\_A1B*.
3. The *client.ProcessChanges()* method is invoked rather than *client.Convert()*
4. All references to watermarks have been removed as they are not part of this sample.

You can apply the same changes to the Java sample in section 5.2 to carry out the same conversion using that language.

Some minor clean-up has been carried out as well to make the code even shorter. After running the example the resulting file validates perfectly according to Acrobat X Pro.



### Sample Code

Listed below is sample code to convert PDF to PDF/A. You can either copy the code or open the VS project from the *Sample Code* folder in the *Start Menu*.

The sample code expects the path of the PDF file on the command line. If the path is omitted then the first PDF file found in the current directory will be used.

Please note that you need the [PDF Converter Professional](#) add-on license in addition to a valid *PDF Converter for SharePoint* or *PDF Converter Services* License in order to use this functionality.

1. Download and install version 5.2 of the Muhimbi PDF Converter Services or PDF Converter for SharePoint.
2. Install the prerequisites and enable PDF/A post processing in the service's configuration file as described in the Administration Guide under *Appendix – Post processing PDF output to PDF/A*.
3. Create a new Visual Studio C# Console application named *PDFA\_Conversion*.
4. Add a Service Reference to the following URL and specify *ConversionService* as the namespace  
`http://localhost:41734/Muhimbi.DocumentConverter.WebService/?wsdl`
5. Paste the following code into *Program.cs*.
6. Make sure the output folder contains a PDF file.
7. Compile and execute the application. The converted PDF/A file will automatically be opened in your system's PDF reader.

```
using System;
using System.Diagnostics;
using System.IO;
using System.ServiceModel;
using Watermarking.ConversionService;

namespace PDFA_Conversion
{
    class Program
    {
        // ** The URL where the Web Service is located. Amend host name if needed.
        static string SERVICE_URL = "http://localhost:41734/Muhimbi.DocumentConverter.WebService/";

        static void Main(string[] args)
        {
            DocumentConverterServiceClient client = null;
            try
            {
                // ** Determine the source file and read it into a byte array.
                string sourceFileName = null;
                if (args.Length == 0)
                {
                    // ** If nothing is specified then read the first PDF file from the folder.
                    string[] sourceFiles = Directory.GetFiles(
```

```

Directory.GetCurrentDirectory(), "*.pdf");

if (sourceFiles.Length > 0)
    sourceFileName = sourceFiles[0];
else
{
    Console.WriteLine("Please specify a document to convert to PDF/A.");
    Console.ReadKey();
    return;
}
}
else
    sourceFileName = args[0];

byte[] sourceFile = File.ReadAllBytes(sourceFileName);

// ** Open the service and configure the bindings
client = OpenService(SERVICE_URL);

/** Set the absolute minimum open options
OpenOptions openOptions = new OpenOptions();
openOptions.OriginalFileName = Path.GetFileName(sourceFileName);
openOptions.FileExtension = "pdf";

// ** Set the absolute minimum conversion settings.
ConversionSettings conversionSettings = new ConversionSettings();
conversionSettings.PDFProfile = PDFProfile.PDF_A1B;

// ** Carry out the conversion.
Console.WriteLine("Converting file " + sourceFileName + " to PDF/A.");
byte[] convFile = client.ProcessChanges(sourceFile, openOptions,
                                        conversionSettings);

// ** Write the converted file back to the file system using the same name.
string destinationFileName = Path.GetFileName(sourceFileName);
using (FileStream fs = File.Create(destinationFileName))
{
    fs.Write(convFile, 0, convFile.Length);
    fs.Close();
}
Console.WriteLine("File converted to " + destinationFileName);
// ** Open the generated PDF/A file in a PDF Reader
Console.WriteLine("Launching file in PDF Reader");
Process.Start(destinationFileName);
}
catch (FaultException<WebServiceFaultException> ex)
{
    Console.WriteLine("FaultException occurred: ExceptionType: " +
        ex.Detail.ExceptionType.ToString());
}
catch (Exception ex)
{
    Console.WriteLine(ex.ToString());
}
}

```

```

        finally
        {
            CloseService(client);
        }
        Console.ReadKey();
    }

    /// <summary>
    /// Configure the Bindings, endpoints and open the service using the specified address.
    /// </summary>
    /// <returns>An instance of the Web Service.</returns>
    public static DocumentConverterServiceClient OpenService(string address)
    {
        DocumentConverterServiceClient client = null;
        try
        {
            BasicHttpBinding binding = new BasicHttpBinding();
            // ** Use standard Windows Security.
            binding.Security.Mode = BasicHttpSecurityMode.TransportCredentialOnly;
            binding.Security.Transport.ClientCredentialType =
                HttpClientCredentialType.Windows;
            // ** Increase the client Timeout to deal with (very) long running requests.
            binding.SendTimeout = TimeSpan.FromMinutes(30);
            binding.ReceiveTimeout = TimeSpan.FromMinutes(30);
            // ** Set the maximum document size to 50MB
            binding.MaxReceivedMessageSize = 50 * 1024 * 1024;
            binding.ReaderQuotas.MaxArrayLength = 50 * 1024 * 1024;
            binding.ReaderQuotas.MaxStringLength = 50 * 1024 * 1024;
            // ** Specify an identity (any identity) in order to get it past .net3.5 sp1
            EndpointIdentity epi = EndpointIdentity.CreateUpnIdentity("unknown");
            EndpointAddress epa = new EndpointAddress(new Uri(address), epi);
            client = new DocumentConverterServiceClient(binding, epa);
            client.Open();
            return client;
        }
        catch (Exception)
        {
            CloseService(client);
            throw;
        }
    }

    /// <summary>
    /// Check if the client is open and then close it.
    /// </summary>
    /// <param name="client">The client to close</param>
    public static void CloseService(DocumentConverterServiceClient client)
    {
        if (client != null && client.State == CommunicationState.Opened)
            client.Close();
    }
}

```



## 4.12 Controlling which InfoPath views to Export to PDF

Being able to select which views to export is very useful as quite often different views are used for exporting a form to PDF. Sometimes using the *Print View* is good enough, but other times you need to export a different view or multiple views to PDF format. There are even occasions where different views are exported depending on the state of the data entered in the form.

As always, the best way to illustrate this is by example. The latest version of this tutorial is available on the Muhimbi Blog at the following URL:

<http://blog.muhimbi.com/2010/08/controlling-which-views-to-export-to.html>

### 4.12.1 Use a special view for exporting to PDF

In this scenario we have an Employee Review form with the following 3 views:

1. **Data entry view:** A view used for populating data using the InfoPath client or Forms Services. This is the default view.
2. **Print View:** A special view that is optimised for printing to a network laser printer. This is specified as View 1's Print View.
3. **PDF Export view:** A separate view that is used to export the InfoPath form to PDF format as it contains some information that should only show up in exported PDF files.

As *View 1* is the default view and *View 2* is the Print View for *View 1*, under normal circumstance the 2nd view is used for exporting to PDF. However, we want to use *View 3* for this purpose. We can achieve this by starting the name of View 3 with “\_MuhimbiView”. The Muhimbi PDF Converter will automatically detect all views that start with this name, export them all and merge them together into a single PDF file. Naturally these views can be hidden from the end user by marking them as such.



This is a great solution if you know beforehand that you will always be exporting the same view(s) to PDF format.



### 4.12.2 Determine at runtime which views to export

The previous solution, using view names that start with “\_MuhimbiView”, works great. However, sometimes you need to export a different view depending on the state of the data.

For example, our Expense Claim form consists of the following Views:

1. **Data Entry View 1:** Used by the employee to report expenses.
2. **Data Entry View 2:** Used by the manager to add comments and additional information.
3. **PDF Export View 1:** The view that is used to export the form to PDF format *before* the manager has reviewed the form.
4. **PDF Export View 2:** The view that is used to export the form to PDF format *after* the manager has reviewed the form.

We can implement this by adding a (hidden) text box named “\_MuhimbiViews” (case sensitive **and using the default ‘my’ namespace**) to any of the views and populating it with the name of one or more comma separated view names. The Muhimbi PDF Converter will automatically pick up these names and export them to PDF format. If multiple views are specified then they are automatically concatenated together.

In addition to adding the “\_MuhimbiViews” text field to the form, all the developer of the form needs to do is add a little bit of logic to the Submit event to specify in the “\_MuhimbiViews” field which view name(s) to export.

### 4.12.3 View prioritisation rules

To determine which view or views to export, the Muhimbi PDF Converter uses the following prioritisation rules:

1. When using the web services interface, any *ConversionViews* specified in the *ConverterSpecificSettings* property will be converted. If this property is not set then the following rules will be used to determine which views to convert to PDF.
2. If a field named “\_MuhimbiViews” is found anywhere in the InfoPath form then the content of this field is used to determine which views to export.
3. If the previous field does not exist, is empty or the specified view name does not exist then the converter looks at all view names that start with “\_MuhimbiView”.
4. If none of the previous options apply then the view marked as the Default View is exported.

Regardless of how a view or views are selected for export, if the selected view has a Print View specified then that view is given priority.

Do not use Muhimbi's View selection features in combination with InfoPath's 'Print multiple views' facility. The latter is given priority when converting to PDF.

When the final PDF file is assembled then all selected views are included first, followed by any [converted attachments](#).

## 5 Working with watermarks

As described in chapter **Error! Reference source not found. Error! reference source not found.**, the PDF Conversion Service contains a powerful watermarking engine that can be used to add visible and invisible watermarks to pages as well as adding headers, footers and other recurring items.

### 5.1 Watermarking in .NET

The following C# example shows how to decorate a document with the following watermarks:

1. The word 'Confidential' in the background of the cover page.
2. Page numbers in the right-hand side of the footer on all even pages.
3. Page numbers in the left-hand side of the footer on all odd pages.



The sample code expects the path of the PDF file on the command line. If the path is omitted then the first MS-Word file found in the current directory will be used.

Follow the steps described below to create the sample watermarking application.

1. Create a new Visual Studio C# Console application named *Watermarking*.
2. Add a *Service Reference* to the following URL and specify *ConversionService* as the namespace

<http://localhost:41734/Muhimbi.DocumentConverter.WebService/?wsdl>

3. Paste the following code into Program.cs. Note that this code is practically identical to the sample provided in a previous chapter, with the exception of the *CreateWatermarks* method and the line that assigns the watermarks to the *ConversionSettings* object.

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.IO;
using System.ServiceModel;
using Watermarking.ConversionService;

namespace Watermarking
{
    class Program
    {
        // ** The URL where the Web Service is located. Amend host name if needed.
        static string SERVICE_URL = "http://localhost:41734/Muhimbi.DocumentConverter.WebService/";

        static void Main(string[] args)
        {
            DocumentConverterServiceClient client = null;

            try
            {
                // ** Determine the source file and read it into a byte array.
                string sourceFileName = null;
                if (args.Length == 0)
                {
                    string[] sourceFiles = Directory.GetFiles(
                        Directory.GetCurrentDirectory(), "*.doc");

                    if (sourceFiles.Length > 0)
                        sourceFileName = sourceFiles[0];
                    else
                    {
                        Console.WriteLine("Please specify a document to convert and watermark.");
                        Console.ReadKey();
                        return;
                    }
                }
                else
                {
                    sourceFileName = args[0];
                }

                byte[] sourceFile = File.ReadAllBytes(sourceFileName);

                // ** Open the service and configure the bindings
                client = OpenService(SERVICE_URL);

                /** Set the absolute minimum open options
                OpenOptions openOptions = new OpenOptions();
                openOptions.OriginalFileName = Path.GetFileName(sourceFileName);
                openOptions.FileExtension = Path.GetExtension(sourceFileName);

                // ** Set the absolute minimum conversion settings.
                ConversionSettings conversionSettings = new ConversionSettings();
                conversionSettings.Fidelity = ConversionFidelities.Full;
                conversionSettings.Quality = ConversionQuality.OptimizeForPrint;

                // ** Get the list of watermarks to apply.
                conversionSettings.Watermarks = CreateWatermarks();

                // ** Carry out the conversion.
                Console.WriteLine("Converting file " + sourceFileName);
                byte[] convFile = client.Convert(sourceFile, openOptions, conversionSettings);

                // ** Write the converted file back to the file system with a PDF extension.
                string destinationFileName = Path.GetDirectoryPath(sourceFileName) + @"\" +
                    Path.GetFileNameWithoutExtension(sourceFileName) +
                    "." + conversionSettings.Format;
                using (FileStream fs = File.Create(destinationFileName))
                {
                    fs.Write(convFile, 0, convFile.Length);
                    fs.Close();
                }
            }
        }
    }
}
```

```

        Console.WriteLine("File converted to " + destinationFileName);

        // ** Open the generated PDF file in a PDF Reader
        Process.Start(destinationFileName);
    }
    catch (FaultException<WebServiceFaultException> ex)
    {
        Console.WriteLine("FaultException occurred: ExceptionType: " +
            ex.Detail.ExceptionType.ToString());
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.ToString());
    }
    finally
    {
        CloseService(client);
    }
    Console.ReadKey();
}

/// <summary>
/// Create the watermarks.
/// </summary>
/// <returns>An array of watermarks to apply</returns>
public static Watermark[] CreateWatermarks()
{
    List<Watermark> watermarks = new List<Watermark>();

    // ** Specify the default settings for properties
    Defaults wmDefaults = new Defaults();
    wmDefaults.FillColor = "#000000";
    wmDefaults.LineColor = "#000000";
    wmDefaults.FontFamilyName = "Arial";
    wmDefaults.FontSize = "10";

    // ***** 'Confidential' Text *****

    // ** 'Confidential' watermark for front page
    Watermark confidential = new Watermark();
    confidential.Defaults = wmDefaults;
    confidential.StartPage = 1;
    confidential.EndPage = 1;
    confidential.Rotation = "-45";
    confidential.Width = "500";
    confidential.Height = "500";
    confidential.HPosition = HPosition.Center;
    confidential.VPosition = VPosition.Middle;
    confidential.ZOrder = -1;

    // ** Create a new Text element that goes inside the watermark
    Text cfText = new Text();
    cfText.Content = "Confidential";
    cfText.FontSize = "40";
    cfText.Width = "500";
    cfText.Height = "500";
    cfText.Transparency = "0.10";

    // ** And add it to the list of watermark elements.
    confidential.Elements = new Element[] { cfText };

    // ** And add the watermark to the list of watermarks
    watermarks.Add(confidential);

    // ***** Watermark for Odd pages *****

    Watermark oddPages = new Watermark();
    oddPages.Defaults = wmDefaults;
    oddPages.StartPage = 3;
    oddPages.PageInterval = 2;

```

```

oddPages.Width = "600";
oddPages.Height = "50";
oddPages.HPosition = HPosition.Right;
oddPages.VPosition = VPosition.Bottom;

// ** Add a horizontal line
Line line = new Line();
line.X = "1";
line.Y = "1";
line.EndX = "600";
line.EndY = "1";
line.Width = "5";

// ** Add a page counter
Text oddText = new Text();
oddText.Content = "Page: {PAGE} of {NUMPAGES}";
oddText.Width = "100";
oddText.Height = "20";
oddText.X = "475";
oddText.Y = "15";
oddText.LineWidth = "-1";
oddText.FontStyle = FontStyle.Regular;
oddText.HAlign = HAlign.Right;

// ** And add it to the list of watermark elements
oddPages.Elements = new Element[] { line, oddText };

// ** And add the watermark to the list of watermarks
watermarks.Add(oddPages);

// ***** Watermark for Even pages *****

Watermark evenPages = new Watermark();
evenPages.Defaults = wmDefaults;
evenPages.StartPage = 2;
evenPages.PageInterval = 2;
evenPages.Width = "600";
evenPages.Height = "50";
evenPages.HPosition = HPosition.Left;
evenPages.VPosition = VPosition.Bottom;

// ** No need to create an additional line, re-use the previous one

// ** Add a page counter
Text evenText = new Text();
evenText.Content = "Page: {PAGE} of {NUMPAGES}";
evenText.Width = "100";
evenText.Height = "20";
evenText.X = "25";
evenText.Y = "15";
evenText.LineWidth = "-1";
evenText.FontStyle = FontStyle.Regular;
evenText.HAlign = HAlign.Left;

// ** And add it to the list of watermark elements
evenPages.Elements = new Element[] { line, evenText };

// ** And add the watermark to the list of watermarks
watermarks.Add(evenPages);

return watermarks.ToArray();
}

/// <summary>
/// Configure the Bindings, endpoints and open the service using the specified address.
/// </summary>
/// <returns>An instance of the Web Service.</returns>
public static DocumentConverterServiceClient OpenService(string address)
{
    DocumentConverterServiceClient client = null;

```

```

try
{
    BasicHttpBinding binding = new BasicHttpBinding();
    // ** Use standard Windows Security.
    binding.Security.Mode = BasicHttpSecurityMode.TransportCredentialOnly;
    binding.Security.Transport.ClientCredentialType =
        HttpClientCredentialType.Windows;
    // ** Increase the client Timeout to deal with (very) long running requests.
    binding.SendTimeout = TimeSpan.FromMinutes(30);
    binding.ReceiveTimeout = TimeSpan.FromMinutes(30);
    // ** Set the maximum document size to 50MB
    binding.MaxReceivedMessageSize = 50 * 1024 * 1024;
    binding.ReaderQuotas.MaxArrayLength = 50 * 1024 * 1024;
    binding.ReaderQuotas.MaxStringLength = 50 * 1024 * 1024;

    // ** Specify an identity (any identity) in order to get it past .net3.5 sp1
    EndpointIdentity epi = EndpointIdentity.CreateUpnIdentity("unknown");
    EndpointAddress epa = new EndpointAddress(new Uri(address), epi);

    client = new DocumentConverterServiceClient(binding, epa);

    client.Open();

    return client;
}
catch (Exception)
{
    CloseService(client);
    throw;
}
}

/// <summary>
/// Check if the client is open and then close it.
/// </summary>
/// <param name="client">The client to close</param>
public static void CloseService(DocumentConverterServiceClient client)
{
    if (client != null && client.State == CommunicationState.Opened)
        client.Close();
}
}
}

```

4. Make sure the output folder contains an MS-Word file.
5. Compile and execute the application.

## 5.2 Watermarking in Java

The following Java based sample code is identical to the example provided in section 4.2 with the exception that the *Watermarks* property in the *ConversionSettings* class is now populated with a simple watermark that prints the word 'Confidential' on the front page in combination with the current date.

For details on how to setup your Java environment and generate the Web Service proxies see the before mentioned section 4.2

```
package com.muhimbi.app;

import com.muhimbi.ws.*;
import java.io.*;
import java.net.URL;
import java.util.List;
import javax.xml.bind.JAXBElement;
import javax.xml.namespace.QName;

public class WsClient {

    private final static String DOCUMENTCONVERTERSERVICE_WSDL_LOCATION =
        "http://localhost:41734/Muhimbi.DocumentConverter.WebService/?wsdl";

    public static void main(String[] args) {
        try {
            if (args.length != 1) {
                System.out.println("Please specify a single file name on the command line.");
            } else {
                // ** Process command line parameters
                String sourceDocumentPath = args[0];
                File file = new File(sourceDocumentPath);
                String fileName = getFileName(file);
                String fileExt = getFileExtension(file);
                System.out.println("Converting file " + sourceDocumentPath);

                // ** Initialise Web Service
                DocumentConverterService_Service dcss = new DocumentConverterService_Service(
                    new URL(DOCUMENTCONVERTERSERVICE_WSDL_LOCATION),
                    new QName("http://tempuri.org/", "DocumentConverterService"));
                DocumentConverterService dcs = dcss.getBasicHttpBindingDocumentConverterService();

                // ** Only call conversion if file extension is supported
                if (isFileExtensionSupported(fileExt, dcs)) {
                    // ** Read source file from disk
                    byte[] fileContent = readFile(sourceDocumentPath);

                    // ** Converting the file
                    OpenOptions openOptions = getOpenOptions(fileName, fileExt);
                    ConversionSettings conversionSettings = getConversionSettings();
                    byte[] convertedFile = dcs.convert(fileContent, openOptions, conversionSettings);

                    // ** Writing converted file to file system
                    String destinationDocumentPath = getPDFDocumentPath(file);
                    writeFile(convertedFile, destinationDocumentPath);
                    System.out.println("File converted sucessfully to " + destinationDocumentPath);
                } else {
                    System.out.println("The file extension is not supported.");
                }
            }
        } catch (IOException e) {
            System.out.println(e.getMessage());
        } catch (DocumentConverterServiceGetConfigurationWebServiceFaultExceptionFaultFaultMessage e) {
            printException(e.getFaultInfo());
        }
    }
}
```



```

    } catch (DocumentConverterServiceConvertWebServiceFaultExceptionFaultFaultMessage e) {
        printException(e.getFaultInfo());
    }
}

public static OpenOptions getOpenOptions(String fileName, String fileExtension) {
    ObjectFactory objectFactory = new ObjectFactory();
    OpenOptions openOptions = new OpenOptions();
    // ** Set the minimum required open options. Additional options are available
    openOptions.setOriginalFileName(objectFactory.createOpenOptionsOriginalFileName(fileName));
    openOptions.setFileExtension(objectFactory.createOpenOptionsFileExtension(fileExtension));
    return openOptions;
}

public static ConversionSettings getConversionSettings() {
    ConversionSettings conversionSettings = new ConversionSettings();
    // ** Set the minimum required conversion settings. Additional settings are available
    conversionSettings.setQuality(ConversionQuality.OPTIMIZE_FOR_PRINT);
    conversionSettings.setRange(ConversionRange.ALL_DOCUMENTS);
    conversionSettings.getFidelity().add("Full");
    conversionSettings.setFormat(OutputFormat.PDF);
    conversionSettings.setWatermarks(getWatermarks());
    return conversionSettings;
}

public static JAXBElement<ArrayOfWatermark> getWatermarks()
{
    ObjectFactory objectFactory = new ObjectFactory();
    ArrayOfWatermark watermarks = new ArrayOfWatermark();

    // ** Specify some of the default settings for properties
    Defaults wmDefaults = new Defaults();
    wmDefaults.setFillColor(objectFactory.createDefaultsFillColor("#FF0000"));
    wmDefaults.setFontFamilyName(objectFactory.createDefaultsFontFamilyName("Arial"));

    // ** 'Confidential' watermark for front page
    Watermark confidential = new Watermark();
    confidential.setDefaults(objectFactory.createContainerDefaults(wmDefaults));
    confidential.setStartPage(1);
    confidential.setEndPage(1);
    confidential.setRotation(objectFactory.createElementRotation("-15"));
    confidential.setWidth(objectFactory.createElementWidth("500"));
    confidential.setHeight(objectFactory.createElementHeight("250"));
    confidential.setHPosition(HPosition.CENTER);
    confidential.setVPosition(VPosition.ABSOLUTE);
    confidential.setY(objectFactory.createElementY("275"));
    confidential.setZOrder(-1);

    // ** Create a new Text element that goes inside the watermark
    Text cfText = new Text();
    cfText.setContent(objectFactory.createTextContent("Confidential - {DATE}"));
    cfText.setFontSize(objectFactory.createTextFontSize("40"));
    cfText.getFontStyle().add("Bold");
    cfText.getFontStyle().add("Italic");
    cfText.setWidth(objectFactory.createElementWidth("500"));
    cfText.setHeight(objectFactory.createElementHeight("250"));
    cfText.setTransparency(objectFactory.createElementTransparency("0.10"));

    // ** And add it to the list of watermark elements.
    ArrayOfElement cfElements = new ArrayOfElement();
    cfElements.getElement().add(cfText);
    confidential.setElements(objectFactory.createContainerElements(cfElements));

    // ** And add the watermark to the list of watermarks
    watermarks.getWatermark().add(confidential);

    return objectFactory.createConversionSettingsWatermarks(watermarks);
}

public static String getFileName(File file) {
    String fileName = file.getName();
}

```



```

    return fileName.substring(0, fileName.lastIndexOf('.'));
}

public static String getFileExtension(File file) {
    String fileName = file.getName();
    return fileName.substring(fileName.lastIndexOf('.') + 1, fileName.length());
}

public static String getPDFDocumentPath(File file) {
    String fileName = getFileName(file);
    String folder = file.getParent();
    if (folder == null) {
        folder = new File(file.getAbsolutePath()).getParent();
    }
    return folder + File.separatorChar + fileName + '.' + OutputFormat.PDF.value();
}

public static byte[] readFile(String filepath) throws IOException {
    File file = new File(filepath);
    InputStream is = new FileInputStream(file);
    long length = file.length();
    byte[] bytes = new byte[(int) length];

    int offset = 0;
    int numRead;
    while (offset < bytes.length
        && (numRead = is.read(bytes, offset, bytes.length - offset)) >= 0) {
        offset += numRead;
    }
    if (offset < bytes.length) {
        throw new IOException("Could not completely read file " + file.getName());
    }
    is.close();
    return bytes;
}

public static void writeFile(byte[] fileContent, String filepath) throws IOException {
    OutputStream os = new FileOutputStream(filepath);
    os.write(fileContent);
    os.close();
}

public static boolean isFileExtensionSupported(String extension, DocumentConverterService dcs)
    throws DocumentConverterServiceGetConfigurationWebServiceFaultExceptionFaultFaultMessage
{
    Configuration configuration = dcs.getConfiguration();
    final JAXBElement<ArrayOfConverterConfiguration> converters = configuration
        .getConverters();
    final ArrayOfConverterConfiguration ofConverterConfiguration = converters.getValue();
    final List<ConverterConfiguration> cList = ofConverterConfiguration
        .getConverterConfiguration();

    for (ConverterConfiguration cc : cList) {
        final List<String> supportedExtension = cc.getSupportedFileExtensions()
            .getValue().getString();
        if (supportedExtension.contains(extension)) {
            return true;
        }
    }
    return false;
}

public static void printException(WebServiceFaultException serviceFaultException) {
    System.out.println(serviceFaultException.getExceptionType());
    JAXBElement<ArrayOfstring> element = serviceFaultException.getExceptionDetails();
    ArrayOfstring value = element.getValue();
    for (String msg : value.getString()) {
        System.out.println(msg);
    }
}
}

```

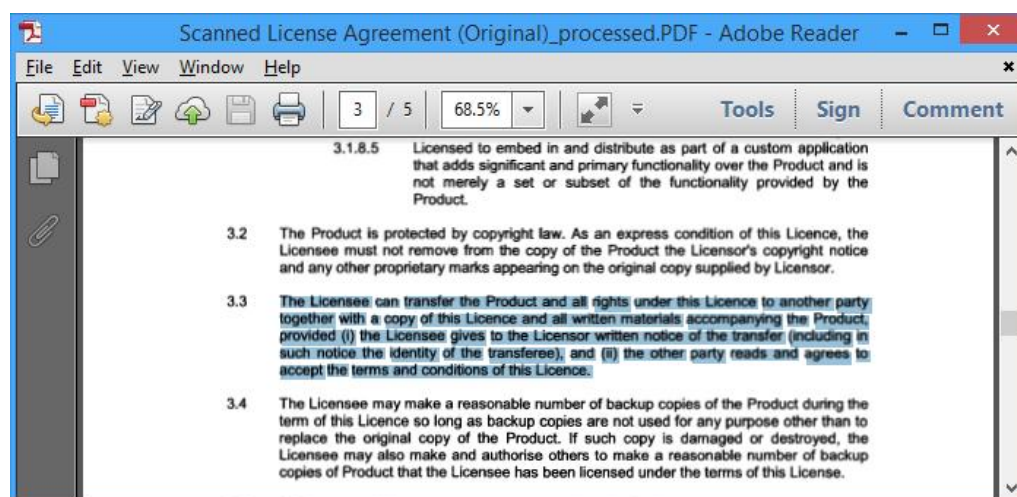
## 6 Carry out OCR (Optical Character Recognition)

One of the more popular questions our support desk receives is about converted PDF files being *searchable* by users and *indexable* by search engines. The answer to that question has always been Yes ..... providing the source document consists of *real* text such as *MS-Word*, *Excel*, *MSG*, *EML*, *HTML* and most of the other [file formats we support](#).

The story is quite different when the source file is a scanned document, which just contains a picture of the text. Generally search engines do not understand these image based files and will simply skip them.

The solution is to *OCR* these documents, a process that recognises text and places it in a hidden layer. The resulting document still looks identical to the original file, but search engines and PDF readers are intelligent enough to retrieve the text. The processed documents are fully searchable and content can even be copied to the clipboard for pasting in other applications.

As of version 7.1 the PDF Converter supports the use of OCR to process Image based files and generate searchable PDFs.



Scanned Document with OCRed text selected

The key features are as follows:

- Server based solution, accessible via a modern Web Service interface (*Java*, *C#*, *Ruby*, *PHP* etc)
- Convert image based files such as *TIFF*, *Scanned PDF*, *PNG*, *JPG*, *BMP*, *GIF* to searchable PDFs.
- Support for multiple languages (*Arabic*, *Danish*, *German*, *English*, *Dutch*, *Finnish*, *French*, *Hebrew*, *Hungarian*, *Italian*, *Norwegian*, *Portuguese*, *Spanish* and *Swedish* with more to come).
- Additional languages and custom fonts can be added by customers and third parties.
- Fully integrated with the conversion pipeline allowing a single web service call to *Convert*, *OCR*, *Watermark*, *Merge* and *Secure* documents.

- Whitelist / Blacklist certain characters. For example limit recognition to *numbers* by *white-listing* 1234567890. This prevents, for example, a 0 (zero) to be recognised as the letter o or O.

Please keep in mind that OCR has its limitations. If the source material is of poor quality (a lot of noise, scratches, low resolution or unusual fonts) then text will most likely not be recognised with a high level of accuracy. However, when the scans use 300dpi and the font size is not smaller than 10pt, then the results are generally very good.

Similarly to the other facilities provided by the PDF Converter, the OCR module will be continuously improved over the years.

The main limitations are currently as follows:

- Some image encoding types such as *JPXDecode* (JPEG2000) are currently not supported. As a workaround use our software to convert the JPEG2000 encoded PDF to a PDF version that uses different encoding (e.g. PDF 1.4).
- Performance is not yet as quick as we would like it to be. Note that OCR performance is measured in *seconds per page*, not milliseconds per page like most of the other operations carried out by our software.
- The system cannot be used to recognise handwriting.

Please note that you need a *PDF Converter Professional* add-on license in addition to a valid *PDF Converter for SharePoint* or *PDF Converter Services* License in order to use this functionality.

## 6.1 OCR files using .NET

In this section we'll show how to use C# to invoke the Web Services interface and create a searchable PDF. The code is nearly identical to a regular conversion request (see 4.1) with the following exceptions:

1. The code looks for PDF source files (an image based PDF is included in the Sample Code folder).
2. The *conversionSettings.OCRSettings* property is populated with relevant OCR settings such as the *language*.
3. The *client.ProcessChanges()* method is invoked rather than *client.Convert()*. (Although this is an optional change)

You can apply the same changes to the PHP (See 4.5) and Ruby (See 4.4) samples. A separate Java based OCR sample can be found in section 6.2. All sample code, including this one, is installed alongside the product and can be accessed from the *Sample Code* shortcut in the *Windows Start Menu*.

This example expects the path of the source PDF file on the command line. If the path is omitted then the first PDF file found in the current directory will be used.

Please carry out the following steps to build the sample application.

1. Install version 7.1 (or newer) of the *Muhimbi PDF Converter Services* or *PDF Converter for SharePoint*.
2. Create a new Visual Studio C# Console application named *OCR\_PDF*.
3. Add a Service Reference to the following URL and specify *ConversionService* as the namespace. If you are developing on a remote system (a system that doesn't run the Muhimbi Conversion Service) then please see [this Knowledge Base Article](#).

`http://localhost:41734/Muhimbi.DocumentConverter.WebService/?wsdl`

4. Paste the following code into *Program.cs*.

```
using System;
using System.Diagnostics;
using System.IO;
using System.ServiceModel;
using OCR_PDF.ConversionService;

namespace OCR_PDF
{
    class Program
    {
        /** ***** **
        /** This code sample is identical to a normal conversion request except for **
        /** the part marked with "OCR OCR OCR". For more information see **
        /** http://blog.muhimbi.com/2013/09/ocr-facilities-provide-by-muhimbis.html **
        /** ***** **

        // ** The URL where the Web Service is located. Amend host name if needed.
        static string SERVICE_URL = "http://localhost:41734/Muhimbi.DocumentConverter.WebService/";

        static void Main(string[] args)
        {
            DocumentConverterServiceClient client = null;

            try
            {
                // ** Delete any processed files from a previous run
                foreach (FileInfo f in new DirectoryInfo(".").GetFiles("*_ocr.pdf"))
                    f.Delete();

                // ** Determine the source file and read it into a byte array.
                string sourceFileName = null;
                if (args.Length == 0)
                {
                    // ** If nothing is specified then read the first PDF file from the folder.
                    string[] sourceFiles = Directory.GetFiles(Directory.GetCurrentDirectory(),
                                                                "*.pdf");

                    if (sourceFiles.Length > 0)
                        sourceFileName = sourceFiles[0];
                    else
                    {
                        Console.WriteLine("Please specify a document to OCR.");
                        Console.ReadKey();
                        return;
                    }
                }
            }
        }
    }
}
```

```

    }
    else
        sourceFileName = args[0];

    byte[] sourceFile = File.ReadAllBytes(sourceFileName);

    // ** Open the service and configure the bindings
    client = OpenService(SERVICE_URL);

    /** Set the absolute minimum open options
    OpenOptions openOptions = new OpenOptions();
    openOptions.OriginalFileName = Path.GetFileName(sourceFileName);
    openOptions.FileExtension = Path.GetExtension(sourceFileName);

    // ** Set the absolute minimum conversion settings.
    ConversionSettings conversionSettings = new ConversionSettings();

    // ** OCR OCR OCR OCR OCR OCR OCR OCR OCR OCR OCR OCR OCR OCR OCR OCR
    OCRSettings ocr = new OCRSettings();
    ocr.Language = OCRLanguage.English.ToString();
    ocr.Performance = OCRPerformance.Slow;
    ocr.Whitelist = string.Empty;
    ocr.BlackList = string.Empty;
    conversionSettings.OCRSettings = ocr;
    // ** OCR OCR OCR OCR OCR OCR OCR OCR OCR OCR OCR OCR OCR OCR OCR OCR

    // ** Carry out the conversion.
    Console.WriteLine("Processing file " + sourceFileName + ".");
    byte[] convFile = client.ProcessChanges(sourceFile, openOptions,
                                           conversionSettings);

    // ** Write the processed file back to the file system with a PDF extension.
    string destinationFileName = Path.GetFileNameWithoutExtension(sourceFileName)
                                + "_ocr.pdf";

    using (FileStream fs = File.Create(destinationFileName))
    {
        fs.Write(convFile, 0, convFile.Length);
        fs.Close();
    }

    Console.WriteLine("File written to " + destinationFileName);

    // ** Open the generated PDF file in a PDF Reader
    Console.WriteLine("Launching file in PDF Reader");
    Process.Start(destinationFileName);
}
catch (FaultException<WebServiceFaultException> ex)
{
    Console.WriteLine("FaultException occurred: ExceptionType: " +
                    ex.Detail.ExceptionType.ToString());
}
catch (Exception ex)
{
    Console.WriteLine(ex.ToString());
}
finally
{
    CloseService(client);
}
Console.ReadKey();
}

/// <summary>
/// Configure the Bindings, endpoints and open the service using the specified address.
/// </summary>
/// <returns>An instance of the Web Service.</returns>
public static DocumentConverterServiceClient OpenService(string address)
{
    DocumentConverterServiceClient client = null;

```

```

try
{
    BasicHttpBinding binding = new BasicHttpBinding();
    // ** Use standard Windows Security.
    binding.Security.Mode = BasicHttpSecurityMode.TransportCredentialOnly;
    binding.Security.Transport.ClientCredentialType =
        HttpClientCredentialType.Windows;
    // ** Increase the client Timeout to deal with (very) long running requests.
    binding.SendTimeout = TimeSpan.FromMinutes(30);
    binding.ReceiveTimeout = TimeSpan.FromMinutes(30);
    // ** Set the maximum document size to 50MB
    binding.MaxReceivedMessageSize = 50 * 1024 * 1024;
    binding.ReaderQuotas.MaxArrayLength = 50 * 1024 * 1024;
    binding.ReaderQuotas.MaxStringLength = 50 * 1024 * 1024;

    // ** Specify an identity (any identity) in order to get it past .net3.5 sp1
    EndpointIdentity epi = EndpointIdentity.CreateUpnIdentity("unknown");
    EndpointAddress epa = new EndpointAddress(new Uri(address), epi);

    client = new DocumentConverterServiceClient(binding, epa);

    client.Open();

    return client;
}
catch (Exception)
{
    CloseService(client);
    throw;
}
}

/// <summary>
/// Check if the client is open and then close it.
/// </summary>
/// <param name="client">The client to close</param>
public static void CloseService(DocumentConverterServiceClient client)
{
    if (client != null && client.State == CommunicationState.Opened)
        client.Close();
}
}
}

```

5. Make sure the output folder contains an image based PDF (e.g. a scan).
6. Compile and execute the application. The processed PDF file will automatically be opened in your system's PDF reader. Try using your PDF Reader's search facility to find and highlight the OCR'd text.

As all this functionality is exposed via a Web Services interface, it works equally well from Java, PHP, Ruby and other web services enabled environments.

Please note that you need a *PDF Converter Professional* add-on license in addition to a valid *PDF Converter for SharePoint* or *PDF Converter Services* License in order to use this functionality.

## 6.2 OCR files using Java

The following sample illustrates how to use OCR to convert a file (preferably a scan) into a fully searchable PDF. In this example we use *wsimport* to generate web service proxy classes, but other web service frameworks are supported as well. (See 4.3 for a generic Apache Axis2 sample).

This example is identical to the code provided in 4.2 with the exception that *OCRSettings* are passed into the *ConversionSettings* object. The sample code is also installed alongside the Conversion Service and can be found by opening the *Sample Code* shortcut in the Windows Start Menu

For details on how to setup your Java environment and generate the Web Service proxies see section 4.2. Once the proxy classes have been created add the following sample code to your project. Compile and run the code and make sure the file to OCR is specified on the command line.

```
package com.muhimbi.app;

import com.muhimbi.ws.*;

import java.io.*;
import java.net.URL;
import java.util.List;

import javax.xml.bind.JAXBElement;
import javax.xml.namespace.QName;

public class WsClient {

    private final static String DOCUMENTCONVERTERSERVICE_WSDL_LOCATION =
        "http://localhost:41734/Muhimbi.DocumentConverter.WebService/?wsdl";

    private static ObjectFactory _objectFactory = new ObjectFactory();

    public static void main(String[] args) {
        try {
            if (args.length != 1) {
                System.out.println("Please specify a single file name on the command line.");
            } else {
                // ** Process command line parameters
                String sourceDocumentPath = args[0];
                File file = new File(sourceDocumentPath);
                String fileName = getFileName(file);
                String fileExt = getFileExtension(file);

                System.out.println("Processing file " + sourceDocumentPath);

                // ** Initialise Web Service
                DocumentConverterService_Service dcss = new DocumentConverterService_Service(
                    new URL(DOCUMENTCONVERTERSERVICE_WSDL_LOCATION),
                    new QName("http://tempuri.org/", "DocumentConverterService"));
                DocumentConverterService dcs = dcss.getBasicHttpBindingDocumentConverterService();

                // ** Only call conversion if file extension is supported
                if (isFileExtensionSupported(fileExt, dcs)) {
                    // ** Read source file from disk
                    byte[] fileContent = readFile(sourceDocumentPath);

                    // ** Converting the file
                    OpenOptions openOptions = getOpenOptions(fileName, fileExt);
                    ConversionSettings conversionSettings = getConversionSettings();
                    byte[] convertedFile = dcs.convert(fileContent, openOptions, conversionSettings);
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private static String getFileName(File file) {
        return file.getName();
    }

    private static String getFileExtension(File file) {
        String fileName = file.getName();
        int dotIndex = fileName.lastIndexOf('.');
        if (dotIndex > 0) {
            return fileName.substring(dotIndex + 1, fileName.length());
        }
        return "";
    }

    private static boolean isFileExtensionSupported(String fileExt, DocumentConverterService dcs) {
        List<String> supportedExtensions = dcs.getSupportedExtensions();
        for (String ext : supportedExtensions) {
            if (ext.equals(fileExt)) {
                return true;
            }
        }
        return false;
    }

    private static OpenOptions getOpenOptions(String fileName, String fileExt) {
        OpenOptions openOptions = new OpenOptions();
        openOptions.setFileName(fileName);
        openOptions.setFileExtension(fileExt);
        return openOptions;
    }

    private static ConversionSettings getConversionSettings() {
        ConversionSettings conversionSettings = new ConversionSettings();
        conversionSettings.setUseOCR(true);
        return conversionSettings;
    }

    private static byte[] readFile(String sourceDocumentPath) {
        try {
            File file = new File(sourceDocumentPath);
            byte[] fileContent = new byte[(int) file.length()];
            FileInputStream fis = new FileInputStream(file);
            fis.read(fileContent);
            fis.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return fileContent;
    }
}
```



```
// ** Writing converted file to file system
String destinationDocumentPath = getPDFDocumentPath(file);
writeFile(convertedFile, destinationDocumentPath);
System.out.println("File converted successfully to " + destinationDocumentPath);

    } else {
        System.out.println("The file extension is not supported.");
    }
}

} catch (IOException e) {
    System.out.println(e.getMessage());
} catch (DocumentConverterServiceGetConfigurationWebServiceFaultExceptionFaultFaultMessage e) {
    printException(e.getFaultInfo());
} catch (DocumentConverterServiceConvertWebServiceFaultExceptionFaultFaultMessage e) {
    printException(e.getFaultInfo());
}
}

public static OpenOptions getOpenOptions(String fileName, String fileExtension) {
    OpenOptions openOptions = new OpenOptions();
    // ** Set the minimum required open options. Additional options are available
    openOptions.setOriginalFileName(_objectFactory.createOpenOptionsOriginalFileName(fileName));
    openOptions.setFileExtension(_objectFactory.createOpenOptionsFileExtension(fileExtension));
    return openOptions;
}

public static ConversionSettings getConversionSettings() {
    ConversionSettings conversionSettings = new ConversionSettings();
    // ** Set the minimum required conversion settings. Additional settings are available
    conversionSettings.setQuality(ConversionQuality.OPTIMIZE_FOR_PRINT);
    conversionSettings.setRange(ConversionRange.ALL_DOCUMENTS);
    conversionSettings.getFidelity().add("Full");
    conversionSettings.setFormat(OutputFormat.PDF);
    conversionSettings.setOCRSettings(_objectFactory.createConversionSettingsOCRSettings(
        getOCRSettings()));

    return conversionSettings;
}

public static OCRSettings getOCRSettings() {
    OCRSettings ocrSettings = new OCRSettings();
    ocrSettings.setLanguage(_objectFactory.createOCRSettingsLanguage("eng"));
    ocrSettings.setPerformance(OCRPerformance.SLOW);
    ocrSettings.setWhiteList(_objectFactory.createOCRSettingsWhiteList(""));
    ocrSettings.setBlackList(_objectFactory.createOCRSettingsBlackList(""));
    return ocrSettings;
}

public static String getFileName(File file) {
    String fileName = file.getName();
    return fileName.substring(0, fileName.lastIndexOf('.'));
}

public static String getFileExtension(File file) {
    String fileName = file.getName();
    return fileName.substring(fileName.lastIndexOf('.') + 1, fileName.length());
}

public static String getPDFDocumentPath(File file) {
    String fileName = getFileName(file);
    String folder = file.getParent();
    if (folder == null) {
        folder = new File(file.getAbsolutePath()).getParent();
    }
    return folder + File.separatorChar + fileName + "_ocr."
        + OutputFormat.PDF.value();
}

public static byte[] readFile(String filepath) throws IOException {
    File file = new File(filepath);

```



```

InputStream is = new FileInputStream(file);
long length = file.length();
byte[] bytes = new byte[(int) length];

int offset = 0;
int numRead;
while (offset < bytes.length
    && (numRead = is.read(bytes, offset, bytes.length - offset)) >= 0) {
    offset += numRead;
}

if (offset < bytes.length) {
    throw new IOException("Could not completely read file " + file.getName());
}
is.close();
return bytes;
}

public static void writeFile(byte[] fileContent, String filepath)
    throws IOException {
    OutputStream os = new FileOutputStream(filepath);
    os.write(fileContent);
    os.close();
}

public static boolean isFileExtensionSupported(String extension, DocumentConverterService dcs)
    throws DocumentConverterServiceGetConfigurationWebServiceFaultExceptionFaultFaultMessage
{
    Configuration configuration = dcs.getConfiguration();
    final JAXBElement<ArrayOfConverterConfiguration> converters =
        configuration.getConverters();
    final ArrayOfConverterConfiguration ofConverterConfiguration = converters.getValue();
    final List<ConverterConfiguration> cList =
        ofConverterConfiguration.getConverterConfiguration();

    for (ConverterConfiguration cc : cList) {
        final List<String> supportedExtension = cc.getSupportedFileExtensions()
            .getValue().getString();
        if (supportedExtension.contains(extension)) {
            return true;
        }
    }

    return false;
}

public static void printException(WebServiceFaultException serviceFaultException) {
    System.out.println(serviceFaultException.getExceptionType());
    JAXBElement<ArrayOfstring> element = serviceFaultException.getExceptionDetails();
    ArrayOfstring value = element.getValue();
    for (String msg : value.getString()) {
        System.out.println(msg);
    }
}
}

```

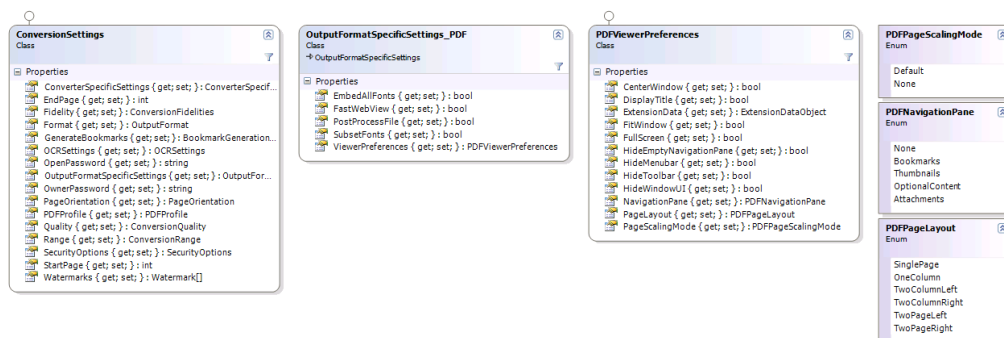
As all this functionality is exposed via a Web Services interface, it works equally well from .NET, PHP, Ruby and other web services enabled environments.

Please note that you need a *PDF Converter Professional* add-on license in addition to a valid *PDF Converter Services* or *PDF Converter for SharePoint* License in order to use this functionality.

## 7 Post processing PDF Files

As of version 7 the PDF Converter adds a number of new post processing facilities, specifically the ability to specify PDF Viewer Preferences, strip or embed fonts, change the PDF Version and linearizing PDFs (a.k.a. *Fast Web View*).

The related classes are as follows:



### 7.1 Specifying PDF Viewer Preferences

Viewer Preferences are *display hints* for the application that is used to view the PDF file, e.g. Adobe Acrobat. These hints are embedded in the PDF file and control such things as the visibility of the Menu and Toolbars, the various panels such as Bookmarks / Attachments, or even viewing the PDF in full screen mode. Please be aware that these are merely hints and not every PDF Reader supports all of them.

The following Viewer Preferences are supported by the Muhimbi PDF Converter:

- **CenterWindow:** Position the document window in the centre of the screen.
- **DisplayTitle:** Display the Document Title in the PDF Reader's window.
- **FitWindow:** Resize the PDF Viewer's window to fit the size of the first page.
- **HideMenubar:** Hide the PDF viewer's menu bar.
- **HideToolBar:** Hide the PDF viewer's tool bars.
- **HideWindowUI:** Hide the user interface elements in the document windows and only display the document's content.
- **PageLayout:** The page layout to use for the document.
- **NavigationPane:** The navigation pane to display when the document is opened.
- **HideEmptyNavigationPane:** If there is no content in the pane then hide it. E.g. when the bookmarks (Outlines) pane is selected, but there are no bookmarks defined, then the pane is hidden.
- **PageScalingMode:** Default scaling option when printing the document.
- **FullScreen:** Display the PDF in full screen mode.

At Muhimbi we pride ourselves at going the extra mile so we have implemented a flag that we believe to be unique. *HideEmptyNavigationPane* checks if any bookmarks or attachments are present and hide those panes if there is no content. This prevents a cluttered user interface when end users view the PDF.

Viewer Preferences can be specified by setting *ConversionSettings.OutputFormatSpecificSettings* (or *MergeSettings.OutputFormatSpecificSettings*) to an instance of *OutputFormatSpecificSettings\_PDF* and populating the *ViewerPreferences* property.

For details see the class diagram in the introduction of this chapter.

When developing in Java please use *Axis2* (See 4.3) as *wsimport* (see 4.2) does not support web service classes that derive from a common base class.

## 7.2 Set PDF Version, Enable Fast Web Views and control Font embedding

Unlike setting the *PDF Viewer Preferences*, the facilities described in this section require a license for the *PDF Converter Professional*, an add-on license for the *PDF Converter Services* and *PDF Converter for SharePoint*.

The following Post Processing settings are available in the *OutputFormatSpecificSettings\_PDF* class:

- **FastWebView:** Enable Fast Web View / Linearization to optimize the PDF for output on the web.
- **EmbedAllFonts:** Embed all fonts into the PDF. Certain fonts may not allow embedding and will therefore never be embedded. Specifying 'false' will remove all fonts from the PDF.
- **SubsetFonts:** Specify if font-subsetting is enabled or not. Font-subsetting embeds only those characters that are used in a document, instead of the entire font. This reduces the size of a PDF file that contains embedded fonts, but may make future content changes problematic.

You can send these settings to the web service by passing a reference to an instance of *OutputFormatSpecificSettings\_PDF* to the *ConversionSettings.OutputFormatSpecificSettings* (or *MergeSettings.OutputFormatSpecificSettings*) property. For details see the class diagram in the introduction of this chapter.

Post processing is enabled by setting the *OutputFormatSpecificSettings\_PDF.PostProcessFile* property to *true*. Please make sure that the Ghostscript prerequisite is installed and configured as described in the Administration Guide. In order to make use of *FastWebView* Ghostscript 9.07 or newer will need to be installed.

When Post processing is enabled the PDF Profile / Version specified in *ConversionSettings.PDFProfile* will automatically be applied to the output file. This includes downgrading the content of the PDF where necessary.

As of Version 7.0 the *PDFProfile* property supports the following PDF Versions and Profiles:

- **Default:** Use whatever PDF version comes out of the underlying converter / source PDF file.
- **PDF\_A1B:** Use the PDF/A1b standard for long term archiving.
- **PDF\_A2B:** Use the PDF/A2b standard for long term archiving.
- **PDF\_1\_1:** PDF 1.1 output (Compatible with Acrobat 2.0 (1994) and later).
- **PDF\_1\_2:** PDF 1.2 output (Compatible with Acrobat 3.0 (1996) and later).
- **PDF\_1\_3:** PDF 1.3 output (Compatible with Acrobat 4.0 (2000) and later).
- **PDF\_1\_4:** PDF 1.4 output (Compatible with Acrobat 5.0 (2001) and later).
- **PDF\_1\_5:** Use PDF Version 1.5. For legacy reasons, out-of-the-box this is treated the same as 'Default', but post processing to 1.5 can be forced using the Config Value "PDF.PostProcessPDF1.5". When post processing is enabled the PDF file will be made compatible with PDF 1.5 (Compatible with Acrobat 6.0 (2003) and later).
- **PDF\_1\_6:** PDF 1.6 output (Compatible with Acrobat 7.0 (2005) and later).
- **PDF\_1\_7:** PDF 1.7 output (Compatible with Acrobat 8.0 (2006) and later).

Please note that when *FastWebView* or PDF/A is enabled you cannot specify any PDF Security settings.

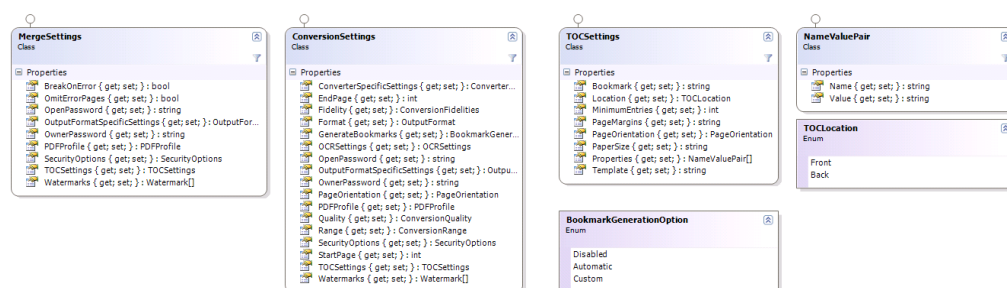
When developing in Java please use *Axis2* (See 4.3) as *wsimport* (see 4.2) does not support web service classes that derive from a common base class.

## 8 Building a Table Of Contents

One of the more popular features provided by the PDF Converter is the ability to convert and merge multiple documents into a single PDF, all in one operation (See section 3.3.1 for details). Although this facility works very well, and even includes the ability to generate PDF bookmarks to aid with navigation inside the merged document, a common request is to add a full Table Of Contents (TOC) to the merged document as well. Read on for details and an example. A copy of this chapter can also be found [on our Blog](#).

### 8.1 Object Model

The classes relevant to dealing with TOCs are as follows. For details see 3.6.



- **MergeSettings:** When merging multiple files and generating a single table of contents, follow the normal procedure for merging files ([sample code](#)) and populate the *MergeSettings.TOCSettings* property as per the sample code below.
- **ConversionSettings:** To generate a table of contents for a single document (so not as part of a merge operation), follow the normal procedure for converting or processing a single file ([sample code](#)) and populate *ConversionSettings.TOCSettings* as per the sample code below.
- **TOCSettings:** All settings related to the generation of the TOC can be found in this class. The available properties are as follows:
  - **Bookmark:** The TOC itself can have its own PDF bookmark to aid with navigation. Specify the text in this property.
  - **Location:** TOCs can be added to the *Front* or *Back* of the document. Enter the relevant option here.
  - **MinimumEntries:** For certain, simple, documents that only have one or 2 bookmarks, it may not make sense to add a table of contents. Use this property to specify the minimum number of entries before a TOC is generated. The default value is '0', which will always create a TOC regardless of the number of entries.
  - **PageMargins:** Page margins in the format set out below. It defaults to a uniform half inch margin.
    - "#{dim}" - for a uniform margin or
    - "#{dim},#{dim},#{dim},#{dim}" - for individual margins
 where  
 - # is numeric value

- {dim} is dimension. Either empty (meaning inches) or "mm", "in", "in.", "inch" or "inches".
- **PageOrientation:** The orientation used by the TOC. *Portrait*, *Landscape* or *Default*. The *Default* option uses the same orientation as the page following (or preceding) the TOC depending on the value specified in *Location*.
- **PaperSize:** A *named* paper size such as *A4* or *Letter* (See [MSDN](#)) or a custom size in "{width}{dim}{sep}{height}{dim}" format where:
  - {width} and {height} are numerical values (please use a colon '.' as the decimal separator) .
  - {dim} is the dimension which can be 'mm', 'in.' or 'inches'. (It defaults to inches when nothing is specified)
  - {sep} separates the width and the height, either 'by', comma (,) or the letter 'x' Example: "8.5 in. by 6 in."
- **Properties:** Optional properties to pass to the XSL template for display or processing purposes. For details see below.
- **Template:** The XSL template (See 8.3) to use for formatting purposes. This can either be a string containing all the XSL, a path - **local** to the server running the conversion service - to the location of the XSL file, or a URL to the XSL file on a web (or SharePoint) server.
- **NameValuePair:** A single value that can be passed into the XSL using *TOCSettings.Properties*.
- **TOCLocation:** Used by *TOCSettings.Location* to determine where the TOC should go.
- **BookmarkGenerationOption:** As explained in *XML Source Data* (8.2), the TOC system is based on the content and structure of PDF Bookmarks. It is therefore essential that during the conversion of the source documents *ConversionSettings.GenerateBookmarks* is set to *Automatic*.

Based on the previously described list of classes and properties, adding a TOC may sound complex, but nothing could be further from the truth. The easiest way to get started is to take our [sample code](#), add the following code and then pass *tocSettings* into either *ConversionSettings.TOCSettings* or *MergeSettings.TOCSettings*.

```
/** Create any custom properties that need to be passed into the TOC.
NameValuePair[] properties = new NameValuePair[2];
properties[0] = new NameValuePair() { Name = "title", Value = "Development Guide" };
properties[1] = new NameValuePair() { Name = "status", Value = "Draft" };

// ** Specify the various TOCSettings
TOCSettings tocSettings = new TOCSettings
{
    MinimumEntries = 0,
    Bookmark = "Table Of Contents",
    Location = TOCLocation.Front,
    Properties = properties,
    Template = @"C:\templates\toc.xsl",
};

// ** Pass the TOC Settings into the conversion
conversionSettings.TOCSettings = tocSettings;
```



You are not limited to our sample code, but it is a good starting point. It is even possible to pass the *tocSettings* to both *ConversionSettings*.*TOCSettings* AND *MergeSettings*.*TOCSettings* to generate TOCs for each individual document in a merge operation, and then add an overall TOC for the entire merged document.

The big question is what to specify in the *Template* property. Read on for details.

## 8.2 XML Source Data

To determine what entries to include in the TOC, the conversion service looks at the Bookmarks present in the PDF file. If the source file is not already in PDF format, it will be converted to PDF and – where possible – generate PDF bookmarks based on the internal structure of the document. For example, when converting an MS-Word file the various headings determine the structure of the PDF Bookmarks.

Although in most cases it is not important for our customers to have any knowledge about the internals of the Muhimbi Conversion Service, in this particular case - and by design - it is. Internally, an XML document is generated that represents the content and structure of the PDF Bookmarks, this XML document is then transformed using XSL into HTML. It is this HTML – the language that underpins every website on the internet – that determines the formatting of the TOC. Developers have full control over the XSL, providing an enormous amount of flexibility.

Let's take our Administration Guide as an example. When converted to PDF a set of nested PDF bookmarks are created, which internally generates the following XML (*truncated as it is several pages long*).

```
<?xml version="1.0" encoding="utf-8"?>
<toc>
  <topics>
    <topic title="Administration Guide - TOC" target="[GUID]" level="0" page="1" />
    <topic title="1 Introduction" target="[GUID]" level="0" page="8">
      <topic title="1.1 Prerequisites" target="[GUID]" level="1" page="10" />
      <topic title="1.2 Solution architecture" target="[GUID]" level="1" page="11" />
    </topic>
    ....
    <topic title="Appendix - Licensing" target="[GUID]" level="0" page="69" />
  </topics>
  <properties>
    <property name="title">Some Document Title</property>
  </properties>
</toc>
```

The generated XML is fairly straight forward, a number of nested *topic* elements make up the structure. Each element has a descriptive *title* attribute, a *level* attribute (which matches the nesting level), a *page* attribute containing the page number, and a *target* attribute which is used for internal processing purposes (this example shows [GUID] as it is not relevant).

Please note: All page numbers in the TOC reflect the physical page number of that page in the generated PDF, including the addition of the TOC page itself. If the source document(s) already display page numbers, then these may no longer be the same as the page number listed in the TOC or their actual page

number in the generated PDF. If you wish to change the page numbers displayed in the footer of a document then please use our watermarking facilities (see chapter 3.5).

The list of *topic* elements is followed by a *properties* section. This section, and its contents, consists of a number of optional values that may have been passed into the request. This allows, for example, the addition of information to the TOC to display the document's status, author, title or any other kind of information. In this example we are passing in the title of the document.

### 8.3 XSL Transformation

Although the XML document's content may differ between requests, the structure is always the same. As a result we can use the [XSL industry standard](#) to convert the XML into an attractive looking HTML document. Although XSL may look daunting to the uninitiated, the following sample ([download](#)) is a good starting point and can be amended to suit your particular needs (or used as is).

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <xsl:stylesheet version="1.0"
3     xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
4     xmlns:msxsl="urn:schemas-microsoft-com:xslt"
5     exclude-result-prefixes="msxsl">
6
7   <xsl:output method="html" indent="yes"/>
8
9   <xsl:template match="/toc">
10     <html>
11       <head>
12         <style type="text/css">
13           ul.toc
14             {
15               margin: 0;
16               padding: 0;
17               list-style: none;
18             }
19           ol.toc
20             {
21               margin: 0;
22               padding: 0;
23               margin-left: 10px;
24               list-style: none;
25             }
26           ul.toc li
27             {
28               clear: both;
29               overflow: hidden;
30             }
31           ol.toc li
32             {
33               overflow: hidden;
34             }
35           span.title
36             {
37               float: left;
38               padding-right: 4px;
39             }
40           span.page
41             {
42               float: right;
43               padding-left: 4px;
44             }
45           span.dots
46             {
47               font-size: 0px;
48               width:100%;

```



```

49         border-bottom: 2px dotted black;
50     }
51     a.toc
52     {
53         text-decoration: none;
54         color: #000;
55     }
56 </style>
57 </head>
58 <body>
59 <h1>
60     <xsl:value-of select="properties/property[@name='title']"/>
61 </h1>
62 <br/>
63 <br/>
64 <xsl:apply-templates/>
65 </body>
66 </html>
67 </xsl:template>
68
69 <xsl:template match="topics">
70     <ul class="toc">
71         <xsl:apply-templates/>
72     </ul>
73 </xsl:template>
74
75 <!-- Empty template so properties are not appearing -->
76 <xsl:template match="properties"></xsl:template>
77
78 <xsl:template match="topic[@level='0']">
79     <li>
80         <xsl:element name="a">
81             <xsl:attribute name="href">
82                 <xsl:value-of select="@target"/>
83             </xsl:attribute>
84             <xsl:attribute name="class">toc</xsl:attribute>
85             <span class="title" style="font-weight: 900;">
86                 <xsl:value-of select="@title"/>
87             </span>
88             <span class="page">
89                 <xsl:value-of select="@page"/>
90             </span>
91             <span class="dots"></span>
92         </xsl:element>
93     </li>
94     <ol class="toc">
95         <xsl:apply-templates/>
96     </ol>
97 </xsl:template>
98
99 <xsl:template match="topic">
100     <li>
101         <xsl:element name="a">
102             <xsl:attribute name="href">
103                 <xsl:value-of select="@target"/>
104             </xsl:attribute>
105             <xsl:attribute name="class">toc</xsl:attribute>
106             <span class="title">
107                 <xsl:value-of select="@title"/>
108             </span>
109             <span class="page">
110                 <xsl:value-of select="@page"/>
111             </span>
112             <span class="dots"></span>
113         </xsl:element>
114     </li>
115     <ol class="toc">
116         <xsl:apply-templates/>
117     </ol>
118 </xsl:template>
119 </xsl:stylesheet>

```

Although this is a standard XSL file, the following sections are of particular interest:

- **Lines 12-56:** Standard HTML CSS style sheet which controls the look of the generated HTML.
- **Line 60:** Insert a custom *property* passed into the conversion request. In our example the document's title.
- **Line 76:** An empty template for the *properties* element to prevent this information from being displayed as a plain list.
- **Lines 78-97:** XSL template for generating HTML associated with all *Level 0* topics. If you wish to control the generated HTML for a specific level then copy the *topic[@level='0']* template and change the level number to match to appropriate nesting level.
- **Lines 99-118:** XSL Template for all topic levels that do not have an explicit template defined.

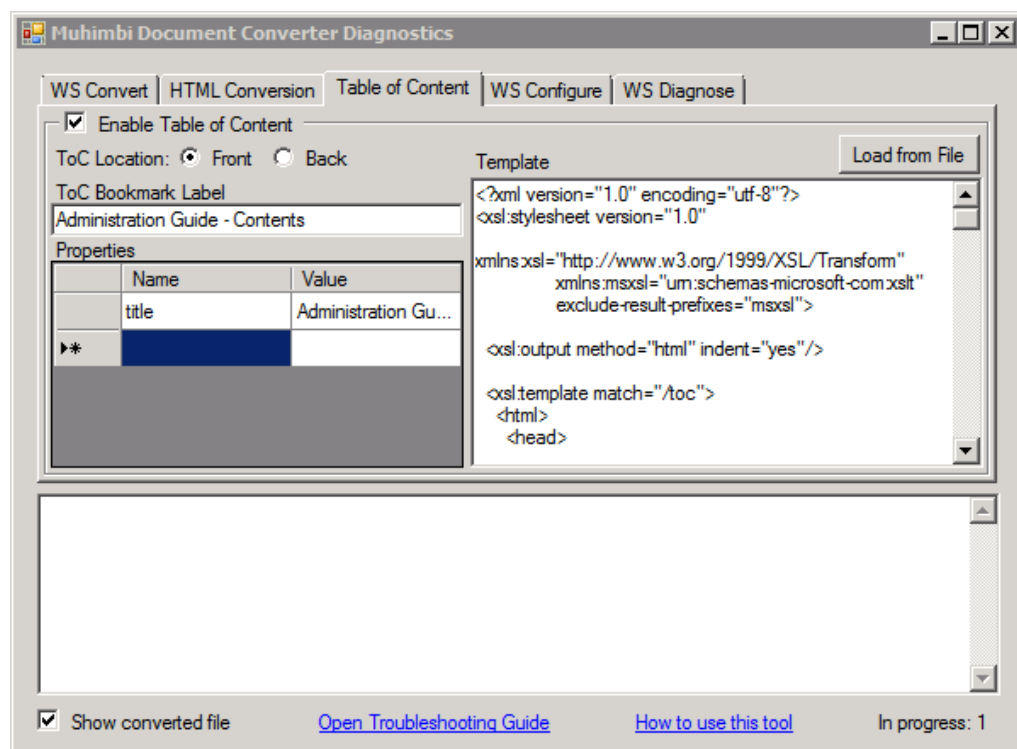
If your experience with XML and XSL is limited then we recommend using the XSL sample provided above. As can be seen below, the results look very good.

### Administration Guide - Contents

<b>Administration Guide - Contents</b>	1
<b>1 Introduction</b>	8
1.1 Relevant articles on the Muhimbi Blog	9
1.2 Prerequisites	10
1.3 Solution architecture	11
<b>2 Deployment</b>	12
2.1 Quick start	12
2.1.1 Installing the Windows Service	12
2.1.2 Installing the SharePoint Front End & Workflow actions	13
2.2 Detailed Installation instructions	14
2.2.1 Installing the Windows Service	14
2.2.2 Installing the SharePoint Front End	15
2.2.3 Feature Activation / Deactivation	16
2.2.4 Installing the License	17
2.3 Post Installation configuration	18
2.3.1 Enabling converters / Specifying location of Conversion Service	18
2.3.2 Tuning the Document Conversion service	19
2.4 Un-installation	32
2.4.1 Un-installing the Document converter service	32
2.4.2 Un-installing the SharePoint Front End via the command line	32
2.4.3 Un-installing the SharePoint Front End via Central Administration	33
2.5 Upgrading from a previous version	33
<b>3 Troubleshooting &amp; Other common tasks</b>	35
3.1 Windows Event Log	35
3.2 SharePoint Trace Log	35
3.3 Document Converter Trace Log	36
3.4 SharePoint audit log	36
3.5 Common issues & Errors	36
3.5.1 Your account is not allowed to deploy SharePoint Solutions	36
3.5.2 Errors on newly added servers	36
3.5.3 An evaluation message is displayed in the UI and converted documents	37
3.5.4 'Unknown Error' or 'resource object not found'	38
3.5.5 Documents using non standard fonts (e.g. Japanese) are not converted properly / The fonts in the destination document are not correct	39
3.5.6 Error messages related to printer drivers or the printer spooler are logged	39
3.5.7 The PDF Converter functionality is not visible in a Document Library	40
3.5.8 Problems converting InfoPath forms without a shared XSN file	40
3.5.9 The 'Convert to PDF' context menu is displayed twice	41
3.5.10 InfoPath forms using Ink controls fail to convert	41
3.5.11 Error 403 (Forbidden) when converting InfoPath forms	41
3.5.12 InfoPath files are converted using an old version of the XSN template	41
3.5.13 Nintex Workflow Activities are not working as expected after upgrading	42
3.5.14 Event Manager error after uninstallation	42
3.5.15 Files uploaded via Windows Explorer do not trigger 'Insert' watermarks	42
3.5.16 'Watermark on Open' does not show watermarks	42
3.5.17 Problems with HTML to PDF Conversion of SharePoint 2010 pages	43
3.5.18 Changing the default bookmark and sort fields when merging files	43
3.5.19 Deploying the Conversion Service on Windows Server 2012 and later	43

## 8.4 Testing & Troubleshooting

Although it is only a basic application, the PDF Converter comes with a handy Diagnostics Tool (including full source code) to test the Table Of Contents facility. While this might be merely a handy test tool, not the official user interface for the TOC facility, it can be incredibly helpful in quickly testing various XSL template designs before integrating them into your solution.



To test the XSL and TOC output, enable the Table of Content as per the screenshot above, modify the XSL template if needed, specify any optional properties, select a file or folder in the *WS Convert* tab and choose either the *Convert* or *Merge* button.

## 9 Troubleshooting

Although the MDCS is a robust and efficient solution, some questions may arise during the day to day operation of the software. This section provides some pointers to answer common questions.

If you still have questions after reading this chapter then please check out the links in chapter 1 *Introduction* as well as our [comprehensive Knowledge Base](#)..

### 9.1 Problems parsing the WSDL

By default the MDCS uses *localhost* as the base address. Most web service client libraries deal with this correctly, however if the service is exposed using a different machine name then you may need to update the *base address*. For details see section 3.3.4 in the Administration Guide.

### 9.2 Converting documents takes a long time

In general the PDF Converter performs extremely well. However, depending on the size and complexity of the documents that are being converted, the conversion process may take some time to execute.

If conversion requests timeout then please have a look at the Administration Guide, section 2.4.4.

### 9.3 The PDF file does not look the same as the source file

Although the MDCS converts documents with very high fidelity and reliability, there are some situations that may cause the converted documents to look different from the source files. The main reasons for this are as follows:

1. One or more fonts used by the document are not installed on the Document Conversion Server. Ask your Administrator to install the correct fonts.
2. The spacing of the characters in InfoPath documents doesn't look correct. Unfortunately InfoPath 2007 does not deal well with certain fonts, even when these fonts have been installed on the server. Try using a different font, creating a separate InfoPath *Print View* or switching to InfoPath 2010.

### 9.4 An evaluation message is displayed in each converted document

When an *evaluation* message is displayed in each converted document then something may be wrong with your license or your license has not been installed. Please see section 2.3 of the Administration Guide for more details about installing the license.

### 9.5 InfoPath Forms fail to convert

When InfoPath documents fail to convert then please consult *Appendix - Using InfoPath with External Data Sources* in the Administration Guide or visit <http://support.muhimbi.com/entries/21278696-my-infopath-form-fails-to-convert-how-can-i-troubleshoot-this>

### 9.6 Converting non supported files

The PDF Converter supports a large number of source file formats. Support for additional formats can be added by following the instructions in the Administration Guide under *Appendix - Creating Custom Converters*.

### Appendix - Relevant articles on the Muhimbi Blog

The [Muhimbi Blog](#) is updated frequently with new articles related to this product. The following posts are relevant to readers of this Developer Guide..

- [Converting files to PDF Format using a Web Services based interface](#) (.NET)
- [Convert files to PDF Format from Java using Web Services](#) (WSImport)
- [Convert files to PDF Format from Java using Web Services](#) (Axis2)
- [Convert files to PDF Format from PHP using a Web Services based interface](#)
- [Convert files to PDF Format from Ruby using a Web Services based interface](#)
- [Invoking the PDF Converter Web Service from Visual Studio 2005 using vb.net](#)
- [Set PDF Version, enable Fast Web Views, embed / strip fonts](#)
- [Specifying PDF Viewer Preferences](#)
- [Converting PDF files to PDF/A1b using a Web Services based interface](#)
- [OCR Facilities provided by Muhimbi's server based PDF Conversion products](#)
- [OCR Scans and Images using a Web Services based Interface](#) (WSImport)
- [OCR Scans and Images using a Web Services based Interface](#) (.NET)
- [Using Windows Azure to convert documents to PDF format](#)
- [Muhimbi PDF Converter Deployment scenarios](#)
- [Performance metrics for the Muhimbi PDF Converter](#)
- [Troubleshooting steps for the PDF Converter.](#)
- [Troubleshooting InfoPath to PDF Conversion / Document Converter Architecture](#)
- [Adding custom Converters to Muhimbi's range of PDF Conversion products](#)
- [Using the Watermarking features of the Muhimbi PDF Converter Services](#)
- [Using the PDF Watermarking features from Java based environments](#)
- [Converting InfoPath forms including all attachments to a single PDF file](#)
- [Convert InfoPath to MS-Word, Excel, XPS and PDF](#)
- [Controlling which views to export to PDF format in InfoPath](#)
- [Dealing with hyperlinks when converting InfoPath files to PDF format](#)
- [Cross-Convert document types \(xls to xlsx, doc to docx\)](#)
- [Programmatically Convert HTML pages to PDF format](#)
- [Converting and merging multiple files using a Web Services based interface](#) (.NET)
- [Converting and merging multiple files using a Web Services based interface](#) (Java)
- [Splitting PDF Files using the PDF Converter Web Service](#) (.NET)
- [Converting AutoCAD \(DXF, DWG\) files to PDF](#)
- [Using Third Party CAD Converters with the Muhimbi PDF Converter](#)
- [Converting TIFF files to PDF](#)
- [Converting Outlook MSG files to PDF including attachments](#)
- [PDF/A Support in the Muhimbi PDF Converter Services & SharePoint](#)
- [Reduce PDF Converter Web Service message size using MTOM](#)

A number of articles written specifically for SharePoint based environments are available as well.

- [Using the PDF Converter from a SharePoint Designer workflow](#)
- [Convert and merge multiple PDF files using a SharePoint Designer workflow](#)
- [Converting multiple SharePoint files to PDF Format using Nintex workflow](#)
- [Watermark PDFs using Nintex Workflow](#)
- [Secure PDFs using Nintex Workflow](#)
- [Convert and Merge PDFs using Nintex Workflow](#)
- [Convert HTML to PDF using Nintex Workflow](#)
- [Copy Meta-Data and set content types using a SharePoint Designer Workflow](#)
- [Convert SharePoint documents to PDF using K2 workflows](#)
- [Splitting PDF Files using a SharePoint workflow](#)
- [Inserting SharePoint List data into a PDF document using a workflow](#)
- [Configure PDF Security from a SharePoint Workflow](#)
- [Watermarking features of the Muhimbi PDF Converter for SharePoint](#)
- [Applying user specific watermarks when a PDF document is opened](#)
- [Convert and merge files to PDF using the SharePoint User Interface](#)
- [Using the PDF Converter for SharePoint from your own code](#)
- [Automatically convert files to PDF using an e-mail enabled Document Library](#)
- [Batch print InfoPath Forms using the PDF Converter for SharePoint](#)
- [Using SharePoint Forms Services to convert InfoPath forms to PDF format](#)
- [Convert HTML pages to PDF format using the SharePoint User Interface](#)
- [Converting SharePoint Lists to PDF format using a SharePoint Designer Workflow](#)
- [Embedding SharePoint Document IDs in PDF files and generating Short URLs](#)
- [Mirror / sync a SharePoint folder structure using the Workflow Power Pack](#)

## Appendix - Licensing

All Muhimbi products are licensed in a way that allows maximum flexibility. Please familiarise yourself with the licensing agreement, particularly section 3 – *Grant of License*, before purchasing our software.

For details see:

1. How we license:  
<http://blog.muhimbi.com/2010/01/how-we-license-our-products-make-sure.html>
2. The License Agreement:  
<http://www.muhimbi.com/Software-License-Agreement.aspx>
3. Licensing FAQ:  
<http://www.muhimbi.com/support/faqs.aspx#Licensing>

In summary we support the following license types. Please refer to the resources above for exact details:

	Number of servers	Unlimited Web Applications	Unlimited Web Farms	Unlimited Locations	Redis-tributable	Includes Source code
Server License	1	<input checked="" type="checkbox"/>				
Enterprise License	Unlimited	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
OEM License	Unlimited	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
Source Code License	Unlimited	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Please note that some older SharePoint specific license types have been discontinued. However, these are still valid for those customers that have purchased them in the past. Please see the License Agreement for details about these old style licenses.

1. **Free evaluation version:** If you install the software without a license you are using the evaluation version. The software is fully functional without any time limits, but an evaluation message will be displayed on most screens, in the workflow history and in any generated document. Please do not use any evaluation software in your production environment. You can get support using any of the means in the Support area on our site.
2. **Server License:** The easiest way to license our software is to buy a Server License for each of your servers, virtual or physical, that runs our software. This could be all SharePoint servers in your farm or non SharePoint based servers that run our software.



3. **Enterprise License:** If you are running our software on more than a handful of servers then it may be more economical to purchase an Enterprise License. This allows the software to be installed on an unlimited number of servers in the organisation.
4. **OEM License:** If you wish to bundle our software with your own solution and redistribute it to 3<sup>rd</sup> parties then you require an OEM License. Please read the details in the Software License Agreement for more information.
5. **OEM License + Source Code:** If you need all the benefits of the OEM License and / or you need access to the source code to make modifications specific to your organisation, then this license type is the best option. Note that we do not provide support for our software once changes have been made to the source code. Please read the details in the Software License Agreement if you want to bundle our software with your own solution.

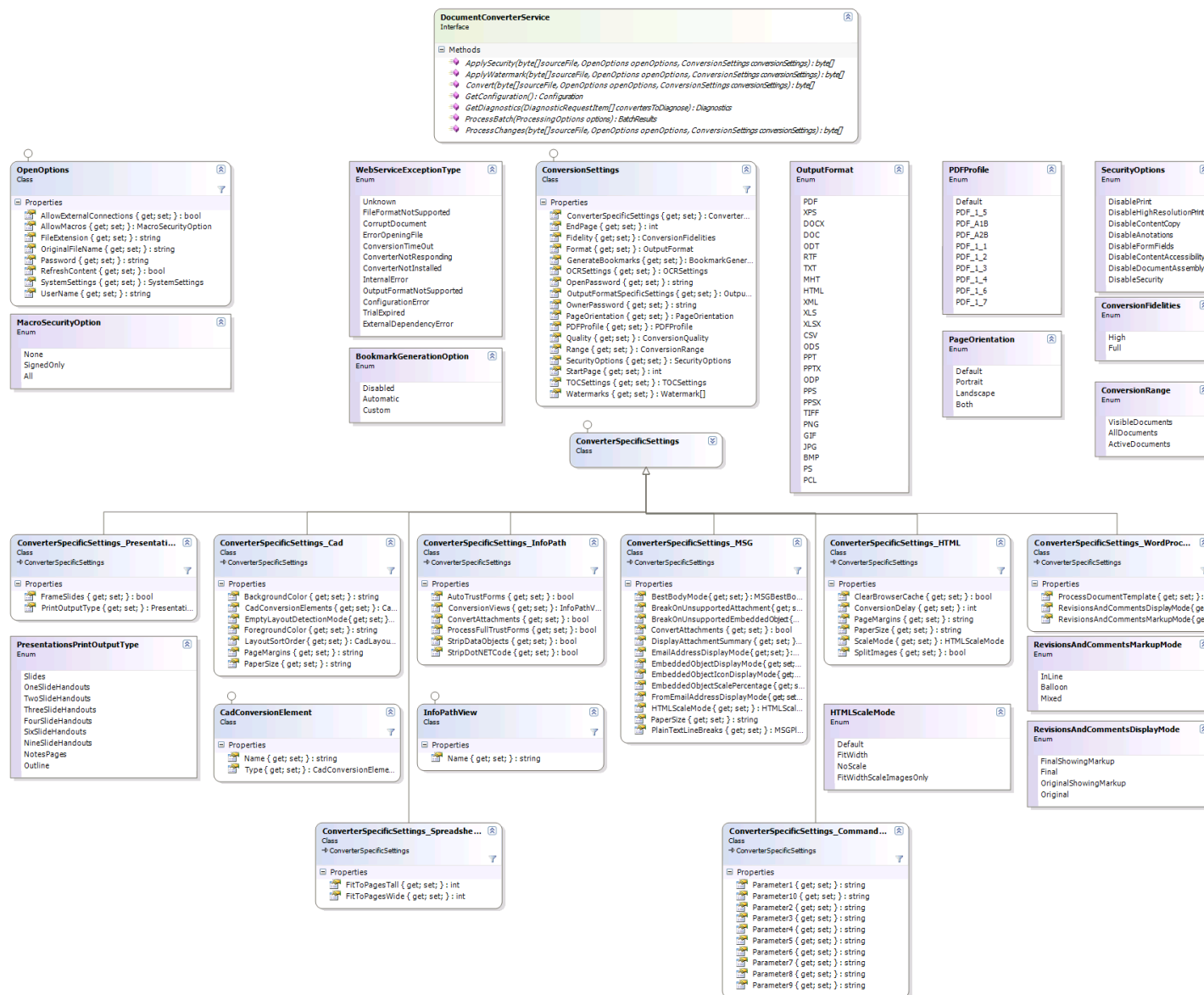
*The PDF Converter for SharePoint license is limited to use from SharePoint environments only. If you wish to invoke the PDF Conversion Service from a non-SharePoint based environment, e.g. Java, .NET or any other Web Services capable system then you will need to purchase a license for the PDF Converter Services.*

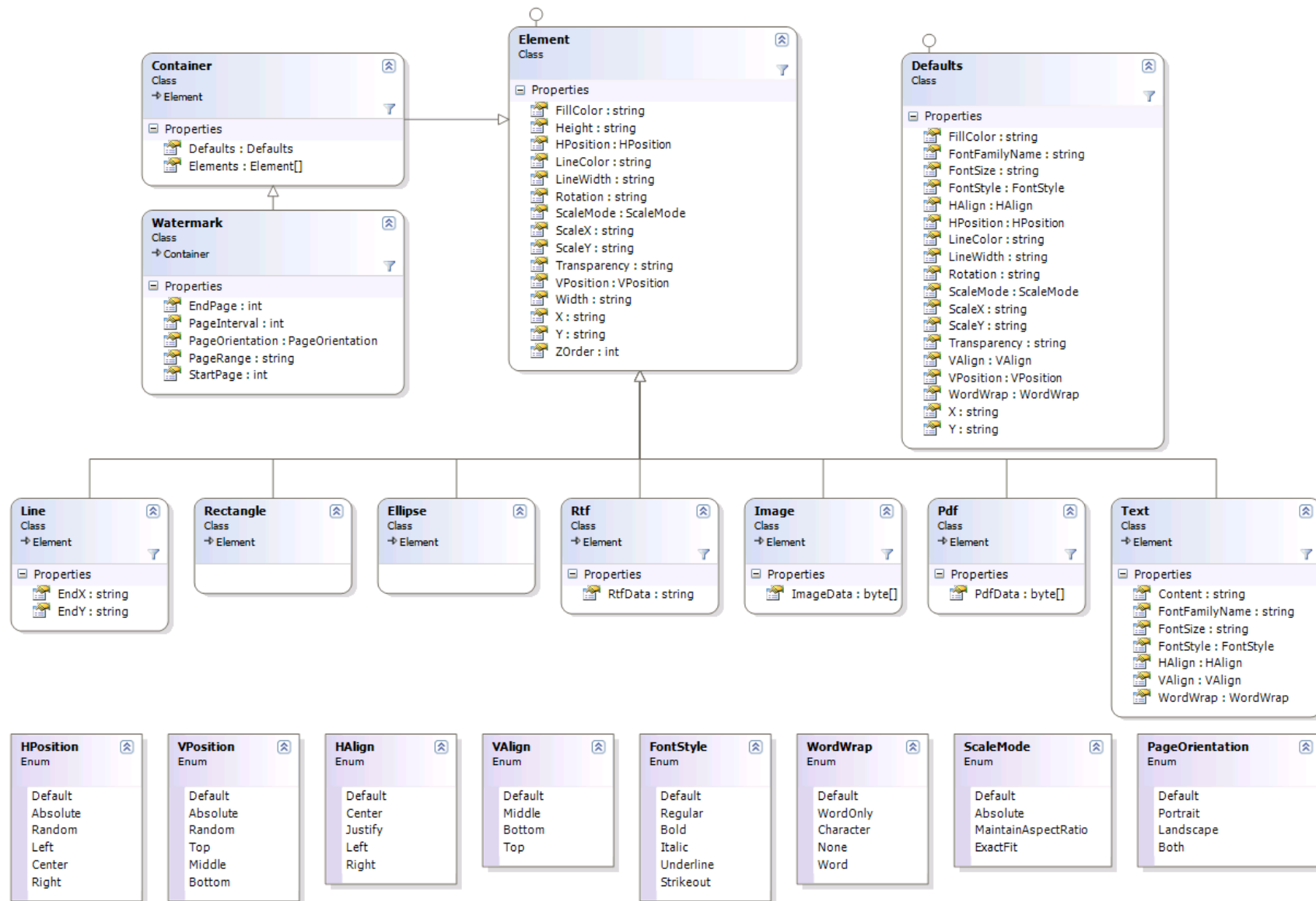
*The PDF Converter Professional license is an add-on that adds additional functionality to either the PDF Converter for SharePoint or the PDF Converter Services. This functionality, e.g. PDF/A post-processing and OCR, is usually associated with more complex environments and has therefore been separated from the main product. Please note that the PDF Converter Professional is a license that must be applied alongside a valid license of the PDF Converter for SharePoint or PDF Converter services. A separate download of the Professional version of the software is not needed, the license unlocks all functionality.*

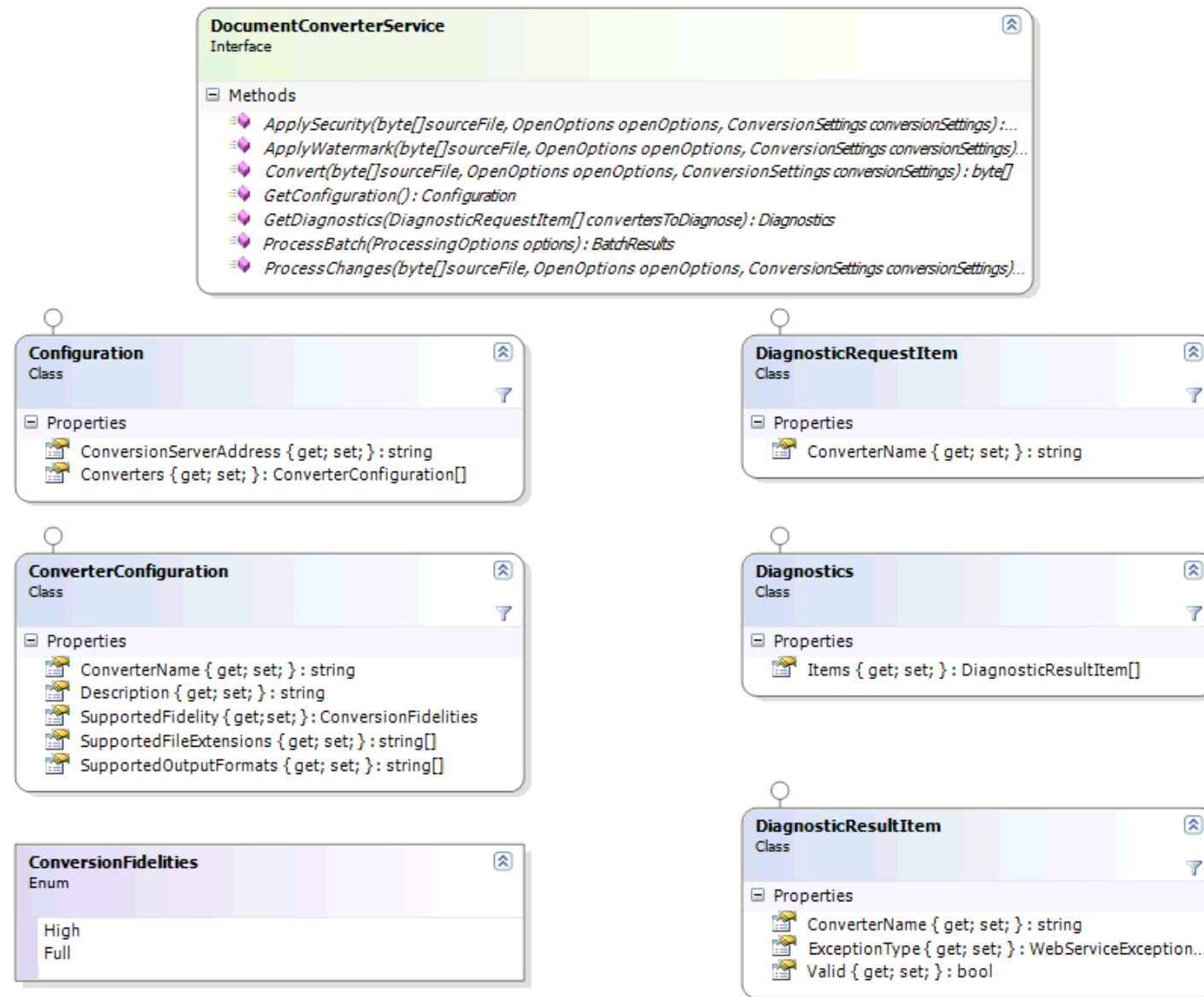
## Appendix – Class Diagrams

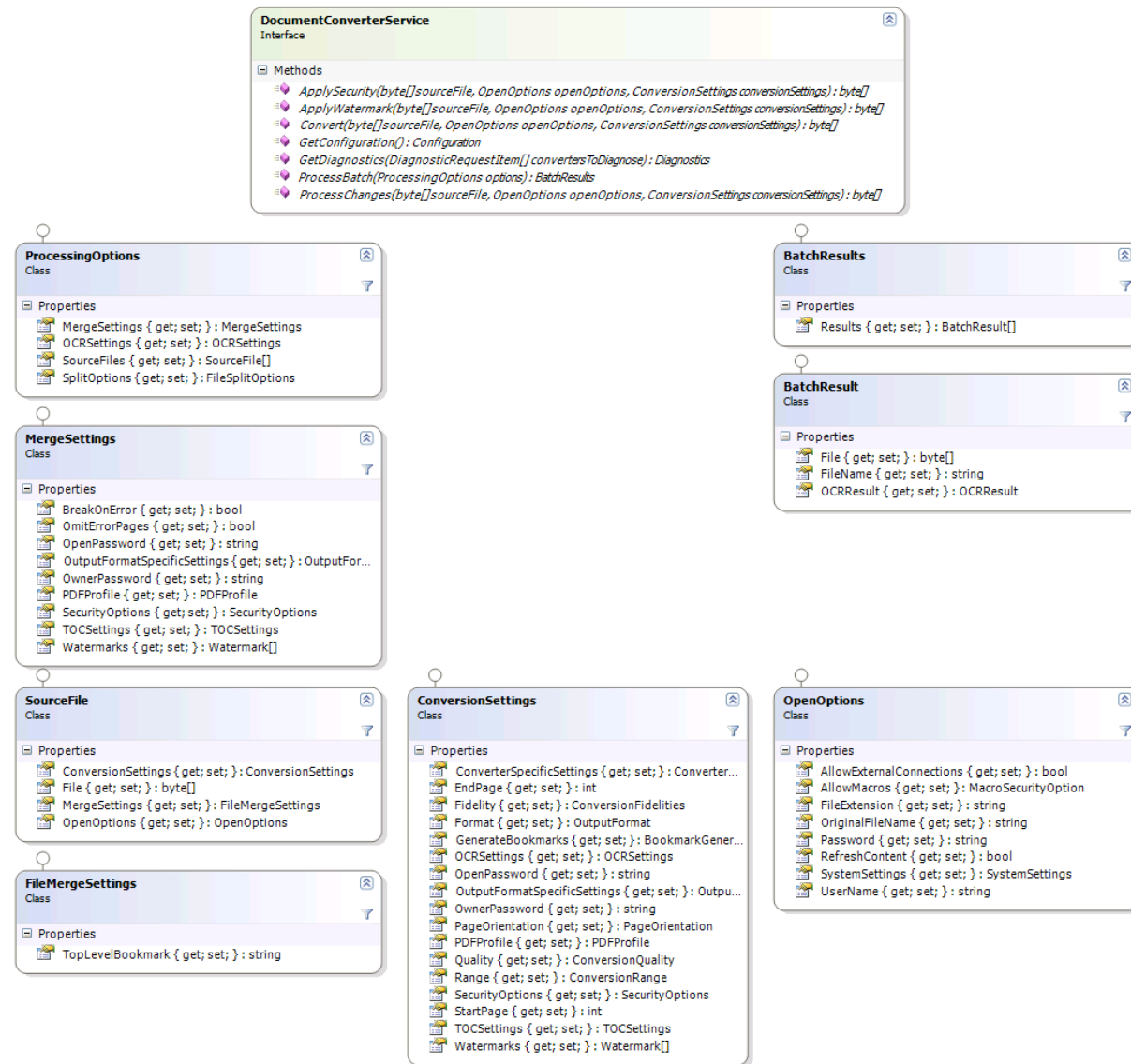
## PDF Converter Services - User & Developer Guide















**ConversionSettings**  
Class

Properties

- ConverterSpecificSettings { get; set; } : ConverterSpecif...
- EndPage { get; set; } : int
- Fidelity { get; set; } : ConversionFidelities
- Format { get; set; } : OutputFormat
- GenerateBookmarks { get; set; } : BookmarkGeneration...
- OCRSettings { get; set; } : OCRSettings
- OpenPassword { get; set; } : string
- OutputFormatSpecificSettings { get; set; } : OutputFor...
- OwnerPassword { get; set; } : string
- PageOrientation { get; set; } : PageOrientation
- PDFProfile { get; set; } : PDFProfile
- Quality { get; set; } : ConversionQuality
- Range { get; set; } : ConversionRange
- SecurityOptions { get; set; } : SecurityOptions
- StartPage { get; set; } : int
- Watermarks { get; set; } : Watermark[]

**OutputFormatSpecificSettings\_PDF**  
Class

→ OutputFormatSpecificSettings

Properties

- EmbedAllFonts { get; set; } : bool
- FastWebView { get; set; } : bool
- PostProcessFile { get; set; } : bool
- SubsetFonts { get; set; } : bool
- ViewerPreferences { get; set; } : PDFViewerPreferences

**PDFViewerPreferences**  
Class

Properties

- CenterWindow { get; set; } : bool
- DisplayTitle { get; set; } : bool
- ExtensionData { get; set; } : ExtensionDataObject
- FitWindow { get; set; } : bool
- FullScreen { get; set; } : bool
- HideEmptyNavigationPane { get; set; } : bool
- HideMenubar { get; set; } : bool
- HideToolBar { get; set; } : bool
- HideWindowUI { get; set; } : bool
- NavigationPane { get; set; } : PDFNavigationPane
- PageLayout { get; set; } : PDFPageLayout
- PageScalingMode { get; set; } : PDFPageScalingMode

**PDFPageScalingMode**  
Enum

- Default
- None

**PDFNavigationPane**  
Enum

- None
- Bookmarks
- Thumbnails
- OptionalContent
- Attachments

**PDFPageLayout**  
Enum

- SinglePage
- OneColumn
- TwoColumnLeft
- TwoColumnRight
- TwoPageLeft
- TwoPageRight

**MergeSettings**  
Class

Properties

- BreakOnError { get; set; } : bool
- OmitErrorPages { get; set; } : bool
- OpenPassword { get; set; } : string
- OutputFormatSpecificSettings { get; set; } : OutputFor...
- OwnerPassword { get; set; } : string
- PDFProfile { get; set; } : PDFProfile
- SecurityOptions { get; set; } : SecurityOptions
- TOCSettings { get; set; } : TOCSettings
- Watermarks { get; set; } : Watermark[]

**ConversionSettings**  
Class

Properties

- ConverterSpecificSettings { get; set; } : Converter...
- EndPage { get; set; } : int
- Fidelity { get; set; } : ConversionFidelities
- Format { get; set; } : OutputFormat
- GenerateBookmarks { get; set; } : BookmarkGener...
- OCRSettings { get; set; } : OCRSettings
- OpenPassword { get; set; } : string
- OutputFormatSpecificSettings { get; set; } : Outpu...
- OwnerPassword { get; set; } : string
- PageOrientation { get; set; } : PageOrientation
- PDFProfile { get; set; } : PDFProfile
- Quality { get; set; } : ConversionQuality
- Range { get; set; } : ConversionRange
- SecurityOptions { get; set; } : SecurityOptions
- StartPage { get; set; } : int
- TOCSettings { get; set; } : TOCSettings
- Watermarks { get; set; } : Watermark[]

**TOCSettings**  
Class

Properties

- Bookmark { get; set; } : string
- Location { get; set; } : TOCLocation
- MinimumEntries { get; set; } : int
- PageMargins { get; set; } : string
- PageOrientation { get; set; } : PageOrientation
- PaperSize { get; set; } : string
- Properties { get; set; } : NameValuePair[]
- Template { get; set; } : string

**NameValuePair**  
Class

Properties

- Name { get; set; } : string
- Value { get; set; } : string

**TOCLocation**  
Enum

- Front
- Back

**BookmarkGenerationOption**  
Enum

- Disabled
- Automatic
- Custom



