muhimbi
Trading as Nutrient

PDF Converter Services - User & Developer Guide

Muhimbi Ltd

Version12.2

Document Control

Draft	Author	Date	Comment
3.0 – 10.1.2	Muhimbi	13/11/2009 – 13/10/2021	Historical versions
10.2	Muhimbi	15/04/2022	Updated for version 10.2
10.2.1	Muhimbi	08/03/2023	Updated for version 10.2.1
10.3.	Muhimbi	14/06/2023	Updated for version 10.3
10.3.1	Muhimbi	22/08/2023	Updated for version 10.3.1
10.4	Stephen Carter	22/09/2023	Updated for version 10.4
11.0	Stephen Carter	13/11/2023	Updated for version 11.0
11.1	Stephen Carter	28/03/2024	Updated for version 11.1
12.0	Stephen Carter	24/11/2024	Updated for version 12.0
12.2	Stephen Carter	12/03/2025	Updated for version 12.2

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.

Rebranding

Muhimbi, Aquaforest, Orpalis, PSPDFKit and Integrify have rebranded as Nutirent.io to align our vision for the future.

Our naming and branding of products is changing to create a cohesive suite of offerings.

This edition will be the last version under the Muhimbi PDF Converter brand.

Disclaimer

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

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, merge files, extract forms data, extract key value pairs, OCR images, apply watermarks or control 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: https://www.muhimbi.com/Products/PDF-Converter-Services
2. Product Overview: https://www.muhimbi.com/products/pdf-converter/
3. Knowledge Base / Frequently Asked Questions: https://www.muhimbi.com/knowledge-base
4. Release Notes: https://www.muhimbi.com/support/documentation/PDF-Converter-Services/Release-Notes
5. Installation & Administration Guide: https://www.muhimbi.com/Products/PDF-Converter-Services/Documentation
6. PDF Converter related content on the Muhimbi Blog: https://www.muhimbi.com/blog/tags/pdf-converter/

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

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.
- Extract forms data from PDF files.
- Extract key value pairs from PDF files.
- 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 PDF and Office 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 for PDF and Office files, 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 here.

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 2024/ 365. 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, dot, dotx, dotm, rtf, txt, wps, xml, odt, ott, mht, html, htm, wpd	
Excel	xls, xlsx, xlsm, xlsb, xlt, xltx, xml, csv, dif, ods, ots, mht, html, htm	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, svg, svgz, vdw, vsdx, vss, vssx, vst, vstx	
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[1]	dwg, dxf	
PDF	pdf, fdf, xfdf, xml	
GdPicture supported formats	dxf, cur, wsq, j2c, webp, jb2, jbig2, jif, jfif, jng, jp2, jpeg, jpg, jpe, koa, lbm, cut, dds, dib, dicom, exif, exr, fax, g3, hdr, heif, heic, iff, ico, j2k, rle, sgi, tga, targa, wbmp, wap, wbm, xbm, xpm	

The PDF Converter also supports output in non-PDF file formats. For details see section 4.6 Cross-Converting between document types.

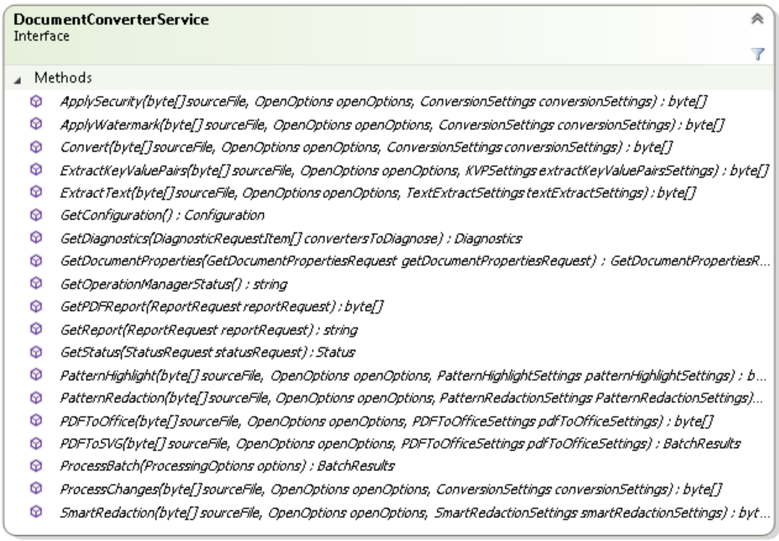
[1] The AutoCAD converter has several automatic recolouring options. For details see *AutoCAD specific switches* in the *Administration Guide*, subsection *Tuning the Document Conversion Service*.

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:



```
DocumentConverterService
Interface

Methods
ApplySecurity(byte[] sourceFile, OpenOptions openOptions, ConversionSettings conversionSettings) : byte[]
ApplyWatermark(byte[] sourceFile, OpenOptions openOptions, ConversionSettings conversionSettings) : byte[]
Convert(byte[] sourceFile, OpenOptions openOptions, ConversionSettings conversionSettings) : byte[]
ExtractKeyValuePairs(byte[] sourceFile, OpenOptions openOptions, KVPSettings extractKeyValuePairsSettings) : byte[]
ExtractText(byte[] sourceFile, OpenOptions openOptions, TextExtractSettings textExtractSettings) : byte[]
GetConfiguration() : Configuration
GetDiagnostics(DiagnosticRequestItem[] convertersToDiagnose) : Diagnostics
GetDocumentProperties(GetDocumentPropertiesRequest getDocumentPropertiesRequest) : GetDocumentPropertiesR...
GetOperationManagerStatus() : string
GetPDFReport(ReportRequest reportRequest) : byte[]
GetReport(ReportRequest reportRequest) : string
GetStatus(StatusRequest statusRequest) : Status
PatternHighlight(byte[] sourceFile, OpenOptions openOptions, PatternHighlightSettings patternHighlightSettings) : b...
PatternRedaction(byte[] sourceFile, OpenOptions openOptions, PatternRedactionSettings PatternRedactionSettings)...
PDFToOffice(byte[] sourceFile, OpenOptions openOptions, PDFToOfficeSettings pdfToOfficeSettings) : byte[]
PDFToSVG(byte[] sourceFile, OpenOptions openOptions, PDFToOfficeSettings pdfToOfficeSettings) : BatchResults
ProcessBatch(ProcessingOptions options) : BatchResults
ProcessChanges(byte[] sourceFile, OpenOptions openOptions, ConversionSettings conversionSettings) : byte[]
SmartRedaction(byte[] sourceFile, OpenOptions openOptions, SmartRedactionSettings smartRedactionSettings) : byt...
```

- **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.
- **ExtractKeyValuePairs**: Extract keys and their associated values from a PDF.
- **ExtractText:** Extract text from a PDF.
- **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 submitted 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.
- **GetPDFReport:** Get a report on usage information in the form of a PDF file. This is part of the Instrumentation process.
- **GetReport:** Get a CSV file on usage. This is part of the Instrumentation process.

- **PatternHighlight:** Highlight text in a document based on regular expression patterns.
- **PatternRedaction:** Redact text in a document based on regular expression patterns.
- **PDFToOffice:** Convert a PDF to an Office type document. This is on preview.
- **PDFToSVG:** Convert a multi-page PDF to a collection of SVG files, one per page.
- **ProcessBatch:** Process multiple files in one call. Currently limited to merge and split operations.
- **SmartRedaction:** Apply Smart Redaction to a PDF.

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.

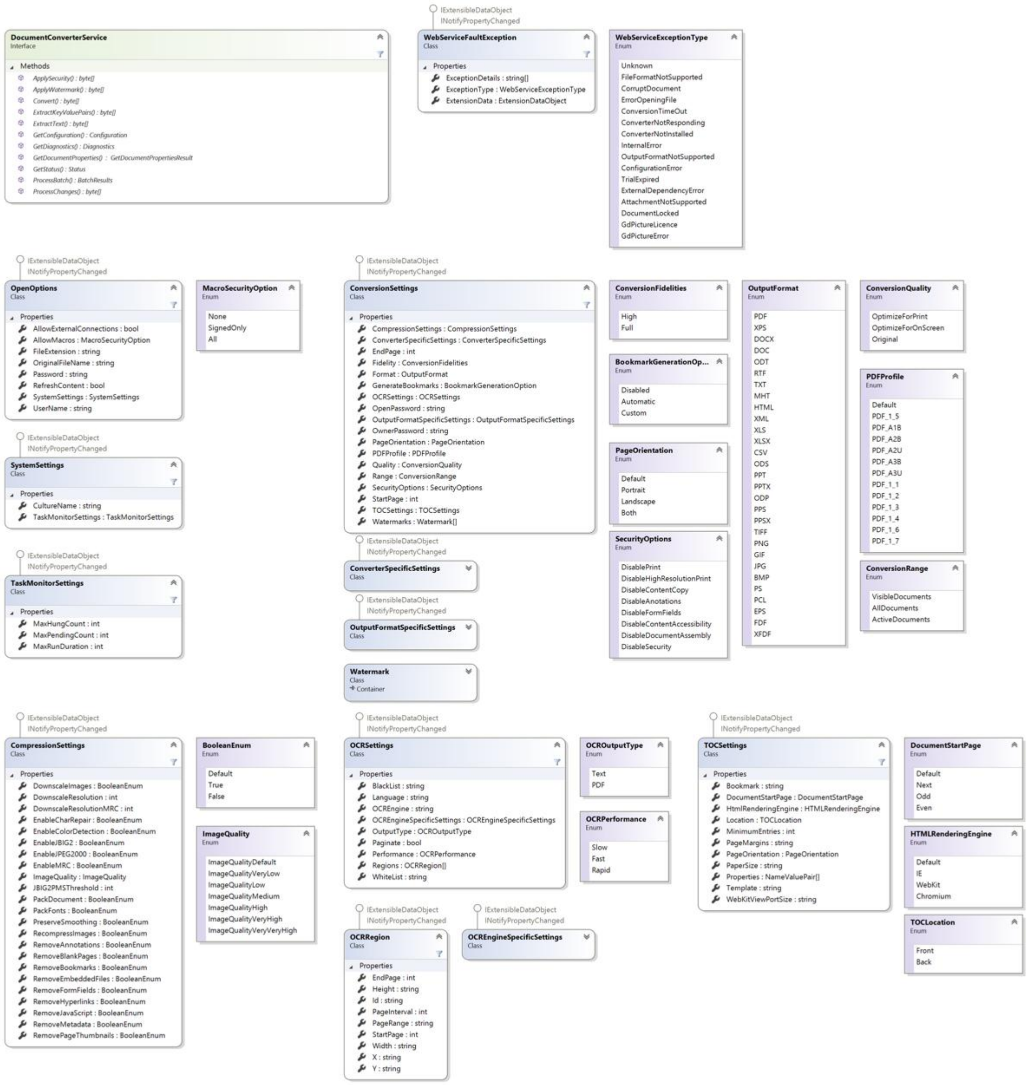
The WSDL can be found at the following location. Change *localhost* to the actual host name if the MDCS is located on a different machine.

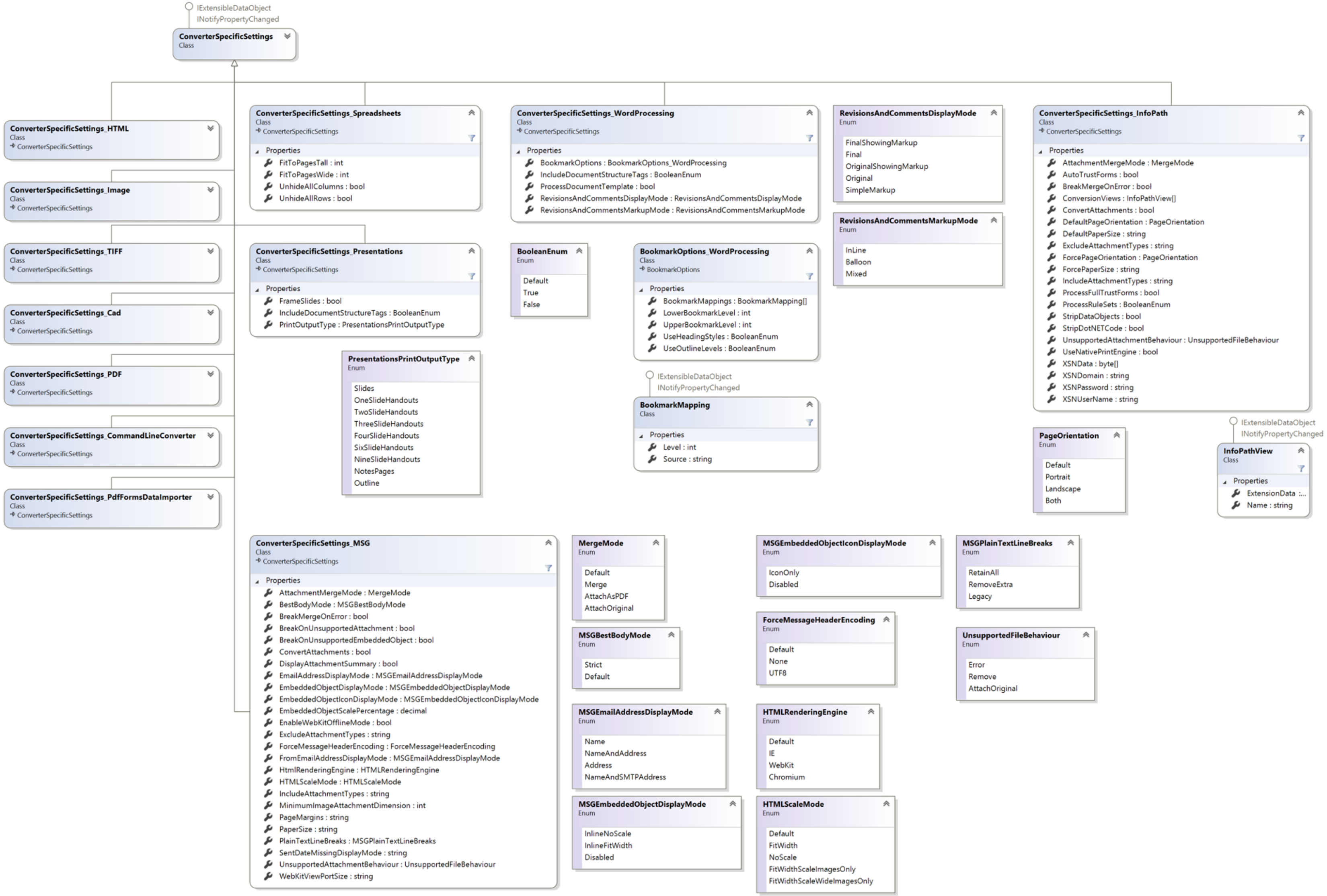
http://localhost:41734/Muhimbi.DocumentConverter.WebService/?wsdl

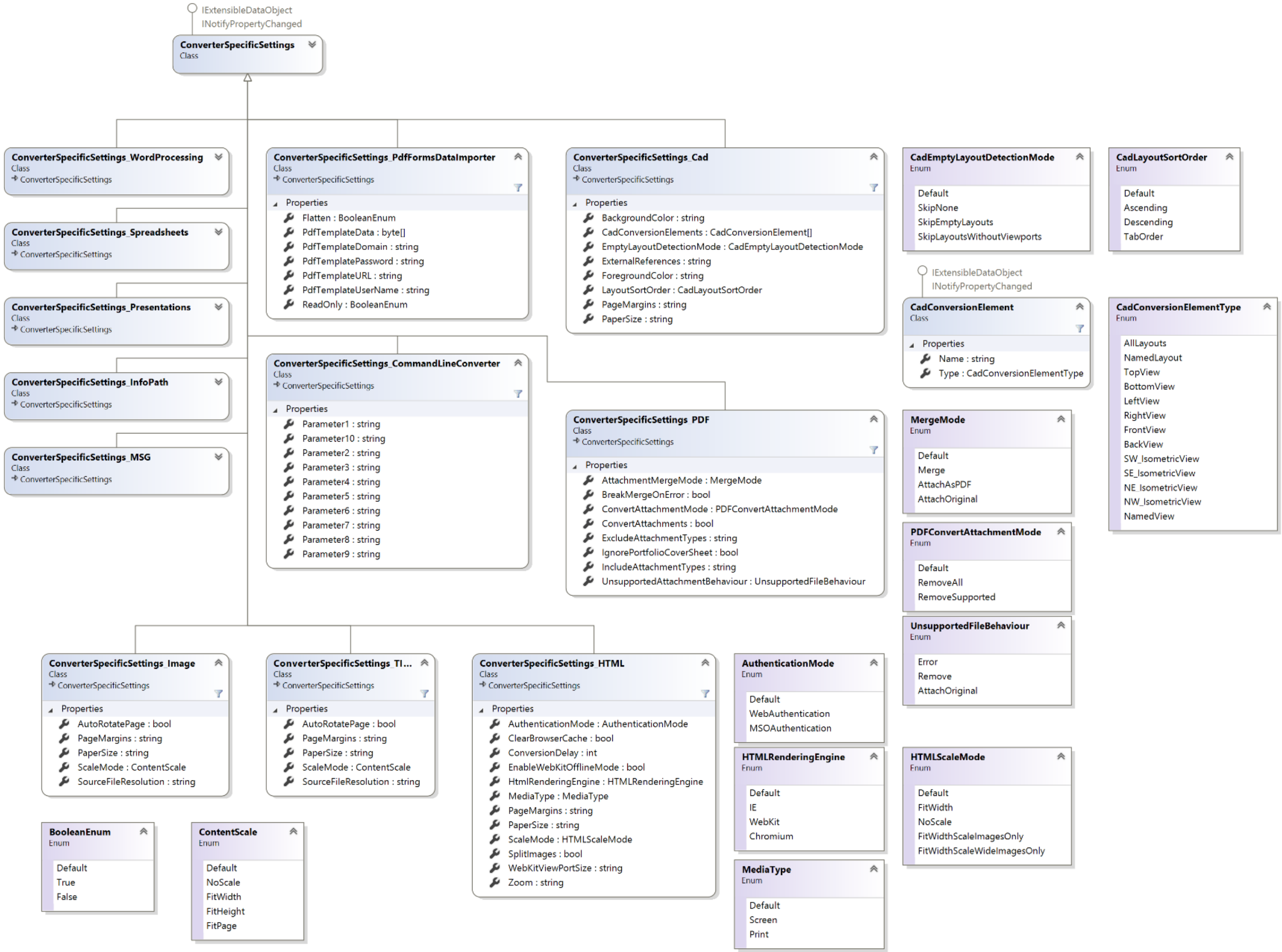
3.2 Conversion

Perhaps not surprisingly, the core of the object model consists of classes and enumerations related to the actual conversion of documents.

This section describes these conversion related classes and methods in detail, the various enumerations are self describing. For code examples see chapters Programatically processing documents and Working with Watermarks.




IExtensibleDataObject
INotifyPropertyChanged
ConverterSpecificSettings
Class
ConverterSpecificSettings_HTML
Class
ConverterSpecificSettings
ConverterSpecificSettings_Image
Class
ConverterSpecificSettings
ConverterSpecificSettings_TIFF
Class
ConverterSpecificSettings
ConverterSpecificSettings_Cad
Class
ConverterSpecificSettings
ConverterSpecificSettings_PDF
Class
ConverterSpecificSettings
ConverterSpecificSettings_CommandLineConverter
Class
ConverterSpecificSettings
ConverterSpecificSettings_PdfFormsDataImporter
Class
ConverterSpecificSettings
ConverterSpecificSettings_Spreadsheets
Class
ConverterSpecificSettings
Properties
FitToPagesTall : int
FitToPagesWide : int
UnhideAllColumns : bool
UnhideAllRows : bool
ConverterSpecificSettings_Presentations
Class
ConverterSpecificSettings
Properties
FrameSlides : bool
IncludeDocumentStructureTags : BooleanEnum
PrintOutputType : PresentationsPrintOutputType
PresentationsPrintOutputType
Enum
Slides
OneSlideHandouts
TwoSlideHandouts
ThreeSlideHandouts
FourSlideHandouts
SixSlideHandouts
NineSlideHandouts
NotesPages
Outline
ConverterSpecificSettings_WordProcessing
Class
ConverterSpecificSettings
Properties
BookmarkOptions : BookmarkOptions_WordProcessing
IncludeDocumentStructureTags : BooleanEnum
ProcessDocumentTemplate : bool
RevisionsAndCommentsDisplayMode : RevisionsAndCommentsDisplayMode
RevisionsAndCommentsMarkupMode : RevisionsAndCommentsMarkupMode
BooleanEnum
Enum
Default
True
False
BookmarkOptions_WordProcessing
Class
BookmarkOptions
Properties
BookmarkMappings : BookmarkMapping[]
LowerBookmarkLevel : int
UpperBookmarkLevel : int
UseHeadingStyles : BooleanEnum
UseOutlineLevels : BooleanEnum
IExtensibleDataObject
INotifyPropertyChanged
BookmarkMapping
Class
Properties
Level : int
Source : string
RevisionsAndCommentsDisplayMode
Enum
FinalShowingMarkup
Final
OriginalShowingMarkup
Original
SimpleMarkup
RevisionsAndCommentsMarkupMode
Enum
InLine
Balloon
Mixed
ConverterSpecificSettings_InfoPath
Class
ConverterSpecificSettings
Properties
AttachmentMergeMode : MergeMode
AutoTrustForms : bool
BreakMergeOnError : bool
ConversionViews : InfoPathView[]
ConvertAttachments : bool
DefaultPageOrientation : PageOrientation
DefaultPaperSize : string
ExcludeAttachmentTypes : string
ForcePageOrientation : PageOrientation
ForcePaperSize : string
IncludeAttachmentTypes : string
ProcessFullTrustForms : bool
ProcessRuleSets : BooleanEnum
StripDataObjects : bool
StripDotNETCode : bool
UnsupportedAttachmentBehaviour : UnsupportedFileBehaviour
UseNativePrintEngine : bool
XSNData : byte[]
XSNDomain : string
XSNPassword : string
XSNUserName : string
PageOrientation
Enum
Default
Portrait
Landscape
Both
IExtensibleDataObject
INotifyPropertyChanged
InfoPathView
Class
Properties
ExtensionData :...
Name : string
ConverterSpecificSettings_MSG
Class
ConverterSpecificSettings
Properties
AttachmentMergeMode : MergeMode
BestBodyMode : MSGBestBodyMode
BreakMergeOnError : bool
BreakOnUnsupportedAttachment : bool
BreakOnUnsupportedEmbeddedObject : bool
ConvertAttachments : bool
DisplayAttachmentSummary : bool
EmailAddressDisplayMode : MSGEmailAddressDisplayMode
EmbeddedObjectDisplayMode : MSGEmbeddedObjectDisplayMode
EmbeddedObjectIconDisplayMode : MSGEmbeddedObjectIconDisplayMode
EmbeddedObjectScalePercentage : decimal
EnableWebKitOfflineMode : bool
ExcludeAttachmentTypes : string
ForceMessageHeaderEncoding : ForceMessageHeaderEncoding
FromEmailAddressDisplayMode : MSGEmailAddressDisplayMode
HtmlRenderingEngine : HTMLRenderingEngine
HTMLScaleMode : HTMLScaleMode
IncludeAttachmentTypes : string
MinimumImageAttachmentDimension : int
PageMargins : string
PaperSize : string
PlainTextLineBreaks : MSGPlainTextLineBreaks
SentDateMissingDisplayMode : string
UnsupportedAttachmentBehaviour : UnsupportedFileBehaviour
WebKitViewPortSize : string
MergeMode
Enum
Default
Merge
AttachAsPDF
AttachOriginal
MSGBestBodyMode
Enum
Strict
Default
MSGEmailAddressDisplayMode
Enum
Name
NameAndAddress
Address
NameAndSMTPAddress
MSGEmbeddedObjectDisplayMode
Enum
InlineNoScale
InlineFitWidth
Disabled
MSGEmbeddedObjectIconDisplayMode
Enum
IconOnly
Disabled
ForceMessageHeaderEncoding
Enum
Default
None
UTF8
HTMLRenderingEngine
Enum
Default
IE
WebKit
Chromium
HTMLScaleMode
Enum
Default
FitWidth
NoScale
FitWidthScaleImagesOnly
FitWidthScaleWideImagesOnly
MSGPlainTextLineBreaks
Enum
RetainAll
RemoveExtra
Legacy
UnsupportedFileBehaviour
Enum
Error
Remove
AttachOriginal


IExtensibleDataObject
INotifyPropertyChanged
ConverterSpecificSettings
Class
ConverterSpecificSettings_WordProcessing
Class
ConverterSpecificSettings
ConverterSpecificSettings_Spreadsheets
Class
ConverterSpecificSettings
ConverterSpecificSettings_Presentations
Class
ConverterSpecificSettings
ConverterSpecificSettings_InfoPath
Class
ConverterSpecificSettings
ConverterSpecificSettings_MSG
Class
ConverterSpecificSettings
ConverterSpecificSettings_PdfFormsDataImporter
Class
ConverterSpecificSettings
Properties
Flatten : BooleanEnum
PdfTemplateData : byte[]
PdfTemplateDomain : string
PdfTemplatePassword : string
PdfTemplateURL : string
PdfTemplateUserName : string
ReadOnly : BooleanEnum
ConverterSpecificSettings_CommandLineConverter
Class
ConverterSpecificSettings
Properties
Parameter1 : string
Parameter10 : string
Parameter2 : string
Parameter3 : string
Parameter4 : string
Parameter5 : string
Parameter6 : string
Parameter7 : string
Parameter8 : string
Parameter9 : string
ConverterSpecificSettings_Cad
Class
ConverterSpecificSettings
Properties
BackgroundColor : string
CadConversionElements : CadConversionElement[]
EmptyLayoutDetectionMode : CadEmptyLayoutDetectionMode
ExternalReferences : string
ForegroundColor : string
LayoutSortOrder : CadLayoutSortOrder
PageMargins : string
PaperSize : string
ConverterSpecificSettings_PDF
Class
ConverterSpecificSettings
Properties
AttachmentMergeMode : MergeMode
BreakMergeOnError : bool
ConvertAttachmentMode : PDFConvertAttachmentMode
ConvertAttachments : bool
ExcludeAttachmentTypes : string
IgnorePortfolioCoverSheet : bool
IncludeAttachmentTypes : string
UnsupportedAttachmentBehaviour : UnsupportedFileBehaviour
CadEmptyLayoutDetectionMode
Enum
Default
SkipNone
SkipEmptyLayouts
SkipLayoutsWithoutViewports
CadLayoutSortOrder
Enum
Default
Ascending
Descending
TabOrder
IExtensibleDataObject
INotifyPropertyChanged
CadConversionElement
Class
Properties
Name : string
Type : CadConversionElementType
CadConversionElementType
Enum
AllLayouts
NamedLayout
TopView
BottomView
LeftView
RightView
FrontView
BackView
SW_IsometricView
SE_IsometricView
NE_IsometricView
NW_IsometricView
NamedView
MergeMode
Enum
Default
Merge
AttachAsPDF
AttachOriginal
PDFConvertAttachmentMode
Enum
Default
RemoveAll
RemoveSupported
UnsupportedFileBehaviour
Enum
Error
Remove
AttachOriginal
ConverterSpecificSettings_Image
Class
ConverterSpecificSettings
Properties
AutoRotatePage : bool
PageMargins : string
PaperSize : string
ScaleMode : ContentScale
SourceFileResolution : string
ConverterSpecificSettings_TI...
Class
ConverterSpecificSettings
Properties
AutoRotatePage : bool
PageMargins : string
PaperSize : string
ScaleMode : ContentScale
SourceFileResolution : string
ConverterSpecificSettings_HTML
Class
ConverterSpecificSettings
Properties
AuthenticationMode : AuthenticationMode
ClearBrowserCache : bool
ConversionDelay : int
EnableWebKitOfflineMode : bool
HtmlRenderingEngine : HTMLRenderingEngine
MediaType : MediaType
PageMargins : string
PaperSize : string
ScaleMode : HTMLScaleMode
SplitImages : bool
WebKitViewPortSize : string
Zoom : string
AuthenticationMode
Enum
Default
WebAuthentication
MSOAuthentication
HTMLRenderingEngine
Enum
Default
IE
WebKit
Chromium
MediaType
Enum
Default
Screen
Print
HTMLScaleMode
Enum
Default
FitWidth
NoScale
FitWidthScaleImagesOnly
FitWidthScaleWideImagesOnly
BooleanEnum
Enum
Default
True
False
ContentScale
Enum
Default
NoScale
FitWidth
FitHeight
FitPage

3.2.1 The Convert Method

The *Convert* method, part of the *DocumentConverterService* interface, carries out the actual conversion. It accepts 3 parameters:

1. **sourceFile:** A *byte[]* containing the actual file to convert, e.g. an Excel file.
2. **openOptions**: The options to use when opening the source file, e.g. Macro Security settings and credentials. For details see section OpenOptions class.
3. **conversionSettings**: The settings to apply when converting the file to PDF format, e.g. *watermarks*, *outputformat*, *security settings*, etc. For details see section ConversionSettings.

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 to 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.
SubscriptionSettings	SubscriptionSettings	Internal Muhimbi use only, please ignore
UserName	String	Optional username for documents that require both username and password (e.g. certain web pages)

3.2.3 The ConversionSettings class

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

Property	Type	Description
CompressionSettings	CompressionSettings	Enables compression operations and contains compression settings.
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 ConverterSpecificSettings options for an example.
EndPage	Int	The last page to render. Leave blank or specify -1 to ignore this value.
Fidelity	Conversion Fidelities	The type of converter to use. Usually *Full*, but in case of custom converters you may need to use *High*. For details see the Administration Guide.
Format	Output Format	Format to convert the file to. See 4.6 Cross-Converting between document types.
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.
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
OutputFormatSpecific Settings	Output Format Specific Settings	An instance of an object that contains settings specific to the output format, e.g. *ViewerPreferences* or *Font Embedding* settings when the output format is PDF (See

Property	Type	Description
		OutputFormatSpecific Settings class 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.
TOCSettings	TOCSettings	Settings related to automatically creating a table of contents. For details see 3.6.
Watermarks	Watermark[]	Optional array of watermarks to apply to the generated PDF file. For details see 3.5.

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
AttachmentMergeMode	MergeMode	How to deal with attachments if the source InfoPath file contains any. Use *Merge*, *AttachAsPDF* or *AttachOriginal*.
AutoTrustForms	Bool	Automatically trust InfoPath 2010 forms. For details see *Appendix - Using InfoPath with External Data Sources* in the Administration Guide.
BreakMergeOnError	Bool	If an error happens while merging attachments, either fail the operation (true) or insert an error page (false).
ConversionViews	InfoPath View[]	List of view names to convert. See 4.12 for details.
ConvertAttachments	Bool	Enable the conversion of attachments.
DefaultPageOrientation	Page Orientation	The Page orientation for InfoPath views that don't explicitly specify a printer / paper size. Either 'Portrait' or 'Landscape'. Leave empty to let InfoPath decide.
DefaultPaperSize	String	The output paper size for InfoPath views where the printer / paper size is not specified. This does not change the paper size for views where the printer / paper size IS specified. Leave empty to take the value from the default printer. specify a named format such as 'A4' or 'Letter' (See MSDN)
ExcludeAttachmentTypes	String	Control which attachment types to exclude. Specify either an empty value to exclude all or specify values in a comma separated list using standard wildcard expressions (e.g., *.docx, tmp???.xls).
IncludeAttachmentTypes	String	Control which attachment types to include. Specify either an empty value to include all or specify values in a comma separated list using standard wildcard expressions (e.g., *.docx, tmp???.xls).
ForcePageOrientation	Page Orientation	Force the page orientation regardless of the printer / paper size being present or not in the definition of the InfoPath view.
ForcePaperSize	String	Force the paper size regardless of the printer / paper size being present or not in the definition of the InfoPath view.

Property	Type	Description
		Leave empty or specify a named format such as 'A4' or 'Letter' (See MSDN)
ProcessFullTrustForms	Bool	Should InfoPath forms marked as requiring Full Trust be processed based on the other parameters (e.g., StripDotNETCode) or not?
ProcessRuleSets	Boolean Enum	Process any rule sets that may be present.
StripDataObjects	Bool	To allow forms to be converted without extensive server configuration, remove all 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.
UnsupportedAttachment Behaviour	Unsupported FileBehaviour	How to deal with unsupported attachments. Specify *Error*, *Remove* or *AttachOriginal*.
UseNativePrintEngine	Bool	Use the new InfoPath Converter (true) or the legacy one (false)
XSNData	Byte[]	Optional XSN file associated with the form. When specified this file will be used rather than the one specified in the InfoPath XML header.
XSNDomain	String	Optional Domain for fetching XSN
XSNPassword	String	Optional Password for fetching XSN
XSNUserName	String	Optional Username for fetching XSN

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
BookmarkOptions	Bookmark Options_Word Processing	Control how PDF Bookmarks are generated.
IncludeDocumentStructureTags	Boolean Enum	Include document structure tags - for accessibility – when generating PDF.
ProcessDocumentTemplate	Bool	Specify if the MS-Word template will need to be stripped out for DOCX files. Specify *true* 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
AuthenticationMode	Authentication Mode	Authentication mode to use when converting HTML: • WebAuthentication - Standard HTTP authentication • MSOAuthentication - SharePoint Online authentication
ClearBrowserCache	bool	Clear the browser's cache before carrying out the conversion.
ConversionDelay	int	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. Specify -1 to use default from the config file.
EnableWebKitOfflineMode	Bool	Do not resolve external content (e.g. images or css loaded via http) during conversion. This may speed up conversion when the server is not able - or allowed - to communicate with internet-based systems.
HTMLRenderingEngine	HTMLRendering Engine	Specify the rendering engine for converting html content. • IE - Use Internet Explorer based converter (legacy use only) • WebKit - Use WebKit (Chrome like) converter
MediaType	MediaType	The CSS media type to use when converting HTML content to PDF • Screen - Use the 'screen' type • Print - Use the 'print' media type
PageMargins	String	The Margin / border around the generated PDF file. One or four {value}{dim} components separated by commas (,) where: • {value} is a numerical value. • {dim} is the dimension which can be mm, in or inches. (Defaults to inches when nothing is specified) When multiple values are specified then the sequence is: left, top, right and bottom.

Property	Type	Description
		Example: "12mm, 24mm, 12mm, 24mm"
PaperSize	String	Specify the paper size to use for the PDF when converting HTML pages. Either: • 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 (decimal separator must be colon '.') - {dim} is the dimension which can be 'mm', 'in.' or 'inches'. (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."
ScaleMode	HTMLScale Mode	Determine how the HTML will be scaled to the PDF page size: • *FitWidth* - HTML is scaled to fit the width of the paper. • *NoScale* - HTML is not scaled, may result in truncating.
SplitImages	Bool	Split images across page breaks or wrap the complete image to the next page.
WebKitViewPortSize	String	Specify the viewport size (for webkit based converter only) • Paper - Dimensions specified in PaperSize minus the margins in PageMargin. • w x h - In pixels. Example "1280 x 1024"

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	Specify the background color. Accepted values: • *Default* - The default black color is used. • Named color, e.g. 'White' as defined on MSDN. • Web color using the "#aarrggbb" or "#rrggbb" format.
CadConversionElements	CadConversionElement[]	Array of named views, layouts, 3D views to convert to PDF.
EmptyLayoutDetectionMode	CadEmptyLayoutDetectionMode	Specifies how the conversion handles empty or nearly empty layouts. Accepted values are: • *SkipNone* - Every layout will be drawn regardless of whether it has anything in it or not. • *SkipEmptyLayouts* - Layouts will be drawn only if they have entities or valid viewports attached to it • *SkipLayoutsWithoutViewports* - Only layouts with valid viewports are drawn
ForegroundColor	String	• *Default*: Objects are drawn in their original color. • *CorrectForBackground* - Objects are drawn in their own color, but colors matching the background color will be inverted to ensure visibility. • Named color, e.g. 'White' as defined on MSDN. • Web color using the "#aarrggbb" or "#rrggbb" format. • *Greyscale* - All colors are converted to a shade of gray based on luminosity. • *GreyscaleDarken* - All colors are converted to a shade of grey then darkened. • *GreyscaleLighten* - All colors are converted to a shade of grey then lightened. • *Darken* - All colors are darkened. • *Lighten* - All colors are lightened.
PageMargins	String	The Margin / border around the generated PDF file. One or four {value}{dim}

Property	Type	Description
		components separated by commas (,) where: • {value} is a numerical value. • {dim} is the dimension which can be 'mm', 'in.' or 'inches'. (Defaults to inches when nothing is specified) When multiple values are specified then the sequence is: left, top, right and bottom. Example: "12mm, 24mm, 12mm, 24mm"
PaperSize	String	Specify the paper size to use for the PDF when converting CAD files. Either: • 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 (decimal separator must be colon '.') - {dim} is the dimension which can be 'mm', 'in.' or 'inches'. (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."
LayoutSortOrder	CadLayout SortOrder	Specify the sort order for layout names. Accepted values are: • *Default* - Use the order in which the layouts are stored in the source file. • *Ascending* - Sort the layout names from A-Z. • *Descending* - Sort the layout names from Z-A. • *TabOrder* – Sort the layouts as they show up in the CAD editor.
ExternalReferences	String	Optional path for resolving external references in drawings.

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
IncludeDocumentStructureTags	Boolean Enum	Include document structure tags - for accessibility – when generating PDF.
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
AttachmentMergeMode	MergeMode	How to deal with attachments if the source email contains any. *Merge*, *AttachAsPDF* or *AttachOriginal*.
BestBodyMode	MSGBest BodyMode	Determine which email body content (Text / HTML / RTF / RTFHTML) to extract when processing MSG files.
BreakMergeOnError	Bool	If an error happens while merging attachments, either fail the operation (true) or insert an error page (false).
BreakOnUnsupported Attachment	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.
BreakOnUnsupported EmbeddedObject	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.
DisplayAttachment Summary	Bool	Specify whether the attachment filenames are displayed in the email header. This setting works independently of the ConvertAttachments setting.
EmailAddress DisplayMode	MSGEmail Address Display Mode	Determine how the *To, Cc* and *Bcc* email addresses are displayed when processing emails.
ExcludeAttachmentTypes	String	Control which attachment types to exclude. Specify either an empty value to exclude all or specify values in a comma separated list using standard wildcard expressions (e.g. *.docx, tmp???.xls).
EmbeddedObject DisplayMode	MSGEmbedded ObjectDisplay Mode	Determines how embedded objects are displayed. NOTE: Where the embedded object is displayed as an icon, use EmbeddedObjectIconDisplay Mode.

Property	Type	Description
EmbeddedObject IconDisplayMode	MSGEmbedded ObjectIcon DisplayMode	Determines how embedded objects are displayed where they are stored as an icon.
EmbeddedObject ScalePercentage	Decimal	The percentage by which embedded objects are scaled prior to rendering. It defaults to 3.33(%).
EnableWebKitOfflineMode	Bool	For HTML emails, do not resolve external content (e.g., images or css loaded via http). This may speed up conversion when the server is not able - or allowed - to communicate with internet-based systems.
ForceMessageHeader Encoding	ForceMessage HeaderEncoding	Control encoding of the email in case it is not specified, and problematic.
FromEmailAddress DisplayMode	MSGEmail Address DisplayMode	Determine how *From* email addresses are displayed when processing emails.
HTMLRenderingEngine	HTMLRendering Engine	Specify the rendering engine for converting HTML content: • IE - Use Internet Explorer based converter (legacy use only) • WebKit - Use WebKit (Chrome like) converter
HTMLScaleMode	HTMLScale Mode	Scale mode for HTML MSG files. Either *FitWidth*, *FitWidthScaleImages Only* or *NoScale*. Unless there is a good reason to change this, use *FitWidthScaleImagesOnly*.
IncludeAttachmentTypes	String	Control which attachment types to include. Specify either an empty value to include all or specify values in a comma separated list using standard wildcard expressions (e.g., *.docx, tmp???.xls).
MinimumImageAttachment Dimension	Int	Minimum width and height before an attached image will be considered for PDF Conversion. Ideal for filtering out small email images. TIFF attachments are always converted.
PageMargins	String	The Margin / border around the generated PDF file. One or four {value}{dim} components separated by commas (,) where: • {value} is a numerical value. • {dim} is the dimension which can be ‘mm’, ‘in.’ or inches. (Defaults to inches when nothing is specified)

Property	Type	Description
		When multiple values are specified then the sequence is: left, top, right and bottom. Example: "12mm, 24mm, 12mm, 24mm"
PaperSize	String	Specify the paper size to use for the PDF when converting HTML based email. Either: • 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 (decimal separator must be colon '.') - {dim} is the dimension which can be 'mm', 'in.' or 'inches'. (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."
PlainTextLineBreaks	MSGPlain TextLine Breaks	Determine how return characters (new lines) in plain text MSG bodies are handled. Use one of *RetainAll*, *RemoveExtra* or *Legacy* (like in earlier versions of the software).
SentDateMissing DisplayMode	String	Text to display when the email has no ‘sent date’, e.g., when it has never been submitted. Only works with MSG
UnsupportedAttachment Behaviour	Unsupported FileBehaviour	How to deal with unsupported attachments. Specify *Error*, *Remove* or *AttachOriginal*.
WebKitViewPortSize	String	Specify the viewport size for HTML content (for webkit converter only) • Paper - Dimensions specified in PaperSize minus an all around margin of 0.5 inches. • w x h - In pixels. Example "1280 x 1024"

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.
UnhideAllColumns	bool	Attempt to include hidden columns in the destination document. This may fail when the source document contains protected or locked content.
UnhideAllRows	bool	Attempt to include hidden rows.

3.2.11 The ConverterSpecificSettings_Image & …_TIFF classes

An instance of this class is optionally passed in the *ConverterSpecificSettings* property of the *ConversionSettings* class for source documents that represent Image based 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
AutoRotatePage	Bool	Automatically rotate the page to match the orientation of the source file.
PageMargins	String	The Margin / border around the generated PDF file. One or four {value}{dim} components separated by commas (,) where • {value} is a numerical value • {dim} is the dimension which can be ‘mm’, ‘in.’ or ‘inches'. (Defaults to inches when nothing is specified) When multiple values are specified then the sequence is: left, top, right and bottom. Example: "12mm, 24mm, 12mm, 24mm"
PaperSize	String	Specify the paper size to use for the PDF when converting images. Either: • A 'Named' paper size such as *'A4'* or *'Letter'* (See MSDN) • or a custom size in "{width}{dim}{sep}{height}{dim}" format where

Property	Type	Description
		- {width} and {height} are numerical values (decimal separator must be colon '.') - {dim} is the dimension which can be 'mm', 'in.' or 'inches'. (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." • FitImage - The paper will fit the size of the image (taking margins and resolution into account)
ScaleMode	Content Scale	Control how the image will be scaled to the PDF page: • NoScale - The image will be drawn using its original size • FitWidth - When wider than 1 page, the image is scaled to fit the width of the paper. • FitHeight - When higher than 1 page, the image is scaled to fit the height of the paper. • FitPage - When wider or higher than 1 page, the image is scaled to fit one page entirely
SourceFileResolution	String	When the ScaleMode is set to 'NoScale', the following value will optionally override the image's DPI. E.g. a 100DPI image will be rendered half size if this value is set to 200.

3.2.12 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.

3.2.13 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.

Property	Type	Description
AttachmentMergeMode	MergeMode	How to deal with attachments if the source PDF file contains any. Use *Merge*, *AttachAsPDF* or *AttachOriginal*.
BreakMergeOnError	bool	If an error happens while merging attachments, either fail the operation (true) or insert an error page (false).
ConvertAttachments	bool	Convert, and Merge, files attached to PDF files. For details see Programmatically Converting and Merging files attached to PDF Documents (muhimbi.com)
ConvertAttachmentMode	PDFConvert Attachment Mode	**RemoveAll:** 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. **RemoveSupported:** 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.
ExcludeAttachmentTypes	String	Control which attachment types to exclude. Specify either an empty value to exclude all or specify values in a comma separated list using standard wildcard expressions (e.g., *.docx, tmp???.xls).
IgnorePortfolioCover Sheet	bool	When PDF attachments are being converted, and the source file is a portfolio file, then this field determines if the portfolio cover sheet is included or not.
IncludeAttachmentTypes	String	Control which attachment types to include. Specify either an empty value to include all or specify values in a comma separated list using standard wildcard expressions (e.g., *.docx, tmp???.xls).
UnsupportedAttachment Behaviour	Unsupported FileBehaviour	How to deal with unsupported attachments. Specify *Error*, *Remove* or *AttachOriginal*.

3.2.14 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.
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 *Viewer Preferences*. Setting this value to true requires a license for the PDF Converter Professional.
NamedDestination ProcessingMode	NamedDestinationProcessingMode	How to deal with the automatic generation of 'Named Destinations' using the PDF's Bookmarks. The default value is 'None'. • None - Make no change to the named destinations defined in the document. • ClearAll - Remove all named destinations. (All bookmarks pointing to existing named destinations will be fixed up automatically) • Merge - Keep existing named destinations and add new ones based on the PDF's bookmarks. • Replace - Remove all existing named destinations and add new ones based on the PDF's bookmarks.

3.2.15 The CompressionSettings class

An instance of this class is appended to the ConversionSetting object to enable compression operations.

Property	Type	Description
RemoveAnnotations	Bool	Remove annotations.
RemoveBlankPages	Bool	Remove blank pages.
RemoveBookmarks	Bool	Remove bookmarks.
RemoveEmbeddedFiles	Bool	Remove embedded files.
RemoveFormFields	Bool	Remove form fields (does not remove XFA fields/data).
RemoveJavaScript	Bool	Remove JavaScript.
RemoveMetadata	Bool	Remove metadata This only removes XMP metadata in the document. PDF information (title, author, custom pdf info etc.) if present, is not touched.
RemovePageThumbnails	Bool	Remove page thumbnails.
PackFonts	Bool	Pack the PDF’s fonts to reduce their size.
PackDocument	Bool	Pack the PDF to reduce its size.
RecompressImages	Bool	Recompress the PDF’s images.
EnableMRC	Bool	MRC engine will be used for compressing the PDF contents.
DownscaleResolutionMRC	Integer	Resolution (DPI) for downscaling the background layer by the MRC engine. Default value is 100.
PreserveSmoothing	Bool	MRC engine will preserve smoothing between different layers.
ImageQuality	Integer	Image quality to be used for the compression of the images from the PDF.
DownscaleImages	Bool	Images from the PDF will be downscaled.
DownscaleResolution	Integer	Resolution used to downscale images. Default value is 150.
EnableColorDetection	Bool	Color detection will be performed on the images from the PDF.
EnableCharRepair	Bool	Character repairing will be performed during bitonal conversion.
EnableJPEG2000	Bool	Use JPEG2000 compression scheme to compress the images.
EnableJBIG2	Bool	Use JBIG2 compression scheme to compress the bitonal images.
JBIG2PMSThreshold	Int	Threshold value for the JBIG2 encoder pattern matching and substitution. Range 0 to 100, any number lower than 100 may lead to lossy compression. Default value is 85.

3.2.16 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).
CultureName	String	The regional settings to use for the current conversion. Experimental feature only supported for Excel conversions at the time of writing. Contact the Muhimbi support desk for details.

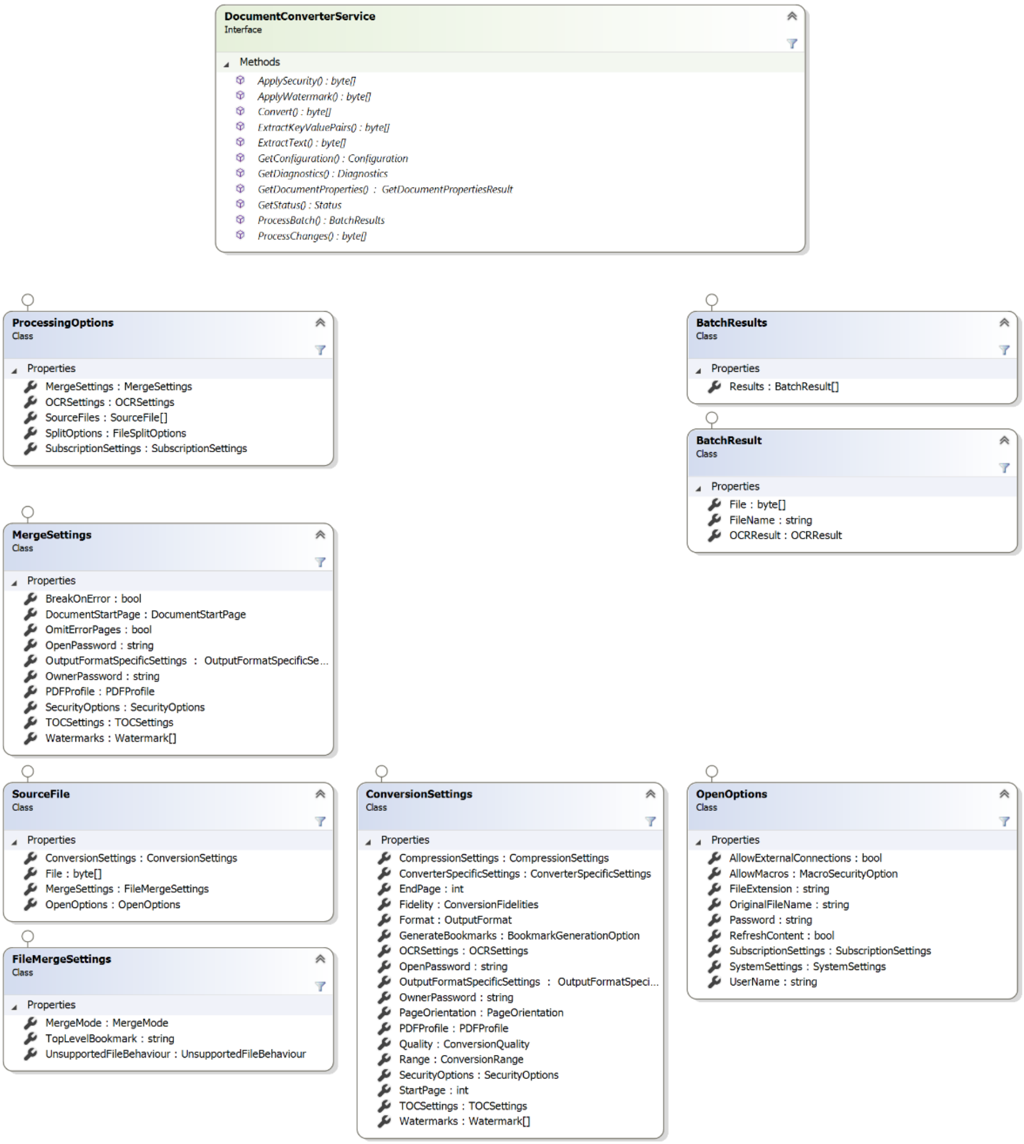
3.2.17 The TaskMonitorSettings class

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.


DocumentConverterService
Interface
Methods
ApplySecurity() : byte[]
ApplyWatermark() : byte[]
Convert() : byte[]
ExtractKeyValuePairs() : byte[]
ExtractText() : byte[]
GetConfiguration() : Configuration
GetDiagnostics() : Diagnostics
GetDocumentProperties() : GetDocumentPropertiesResult
GetStatus() : Status
ProcessBatch() : BatchResults
ProcessChanges() : byte[]
ProcessingOptions
Class
Properties
MergeSettings : MergeSettings
OCRSettings : OCRSettings
SourceFiles : SourceFile[]
SplitOptions : FileSplitOptions
SubscriptionSettings : SubscriptionSettings
BatchResults
Class
Properties
Results : BatchResult[]
BatchResult
Class
Properties
File : byte[]
FileName : string
OCRResult : OCRResult
MergeSettings
Class
Properties
BreakOnError : bool
DocumentStartPage : DocumentStartPage
OmitErrorPages : bool
OpenPassword : string
OutputFormatSpecificSettings : OutputFormatSpecificSe...
OwnerPassword : string
PDFProfile : PDFProfile
SecurityOptions : SecurityOptions
TOCSettings : TOCSettings
Watermarks : Watermark[]
SourceFile
Class
Properties
ConversionSettings : ConversionSettings
File : byte[]
MergeSettings : FileMergeSettings
OpenOptions : OpenOptions
ConversionSettings
Class
Properties
CompressionSettings : CompressionSettings
ConverterSpecificSettings : ConverterSpecificSettings
EndPage : int
Fidelity : ConversionFidelities
Format : OutputFormat
GenerateBookmarks : BookmarkGenerationOption
OCRSettings : OCRSettings
OpenPassword : string
OutputFormatSpecificSettings : OutputFormatSpeci...
OwnerPassword : string
PageOrientation : PageOrientation
PDFProfile : PDFProfile
Quality : ConversionQuality
Range : ConversionRange
SecurityOptions : SecurityOptions
StartPage : int
TOCSettings : TOCSettings
Watermarks : Watermark[]
OpenOptions
Class
Properties
AllowExternalConnections : bool
AllowMacros : MacroSecurityOption
FileExtension : string
OriginalFileName : string
Password : string
RefreshContent : bool
SubscriptionSettings : SubscriptionSettings
SystemSettings : SystemSettings
UserName : string
FileMergeSettings
Class
Properties
MergeMode : MergeMode
TopLevelBookmark : string
UnsupportedFileBehaviour : UnsupportedFileBehaviour

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* properties 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 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 are 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* – The MergeSettings class.
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 *3.3.2*.
SubscriptionSettings	Subscription Settings	Internal Muhimbi use only, please ignore

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.
DocumentStartPage	DocumentStart Page	When printing double sided it is often desirable to let each document in a merged file start on (usually) the right hand page. Behaviour of how documents are aligned in a merge set can be controlled using this property.
OmitErrorPages	Bool	Control if error pages are inserted in the merged document for files that fail to convert. This only has effect if *BreakOnError* is set to 'False'.
OutputFormatSpecific Settings	OutputFormat SpecificSettings	An instance of an object that contains settings specific to the output format, e.g. *ViewerPreferences* or *Font Embedding* settings when the output format is PDF (See section 7 for details).
OpenPassword	String	The ‘open password’ to be applied to the PDF file containing all merged documents.
OwnerPassword	String	The ‘owner password’ to be applied to the PDF file containing all merged documents.
PDFProfile	PDFProfile	The PDF Profile to use for the PDF file containing all merged documents.
SecurityOptions	SecurityOptions	Security restrictions to apply to the PDF file containing all merged documents
TOCSettings	TOCSettings	Settings related to automatically creating a table of contents. For details see 3.6.
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 *SourceFile. ConversionSettings* property.

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: *ByNumberOfPages* or *ByBookmarkLevel*.
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* – The ConversionSettings class for details.
File	Byte[]	The content of the file to process. Leave empty and set *OpenOptions. OriginalFileName* to a URL to convert web pages.
MergeSettings	FileMergeSettings	Settings associated with merging this file. See *3.3.4* – The MergeSettings class 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
MergeMode	MergeMode	How to merge the specified files, Either *Merge*, *AttachAsPDF* or *AttachOriginal*.
TopLevelBookmark	String	The name to use as the 'top level bookmark' in the combined PDF file.
UnsupportedFile Behaviour	UnsupportedFile Behaviour	How to deal with unsupported files. Specify *Error*, *Remove* or *AttachOriginal*.

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 *file merge operation* the merged file is always stored in element 0.

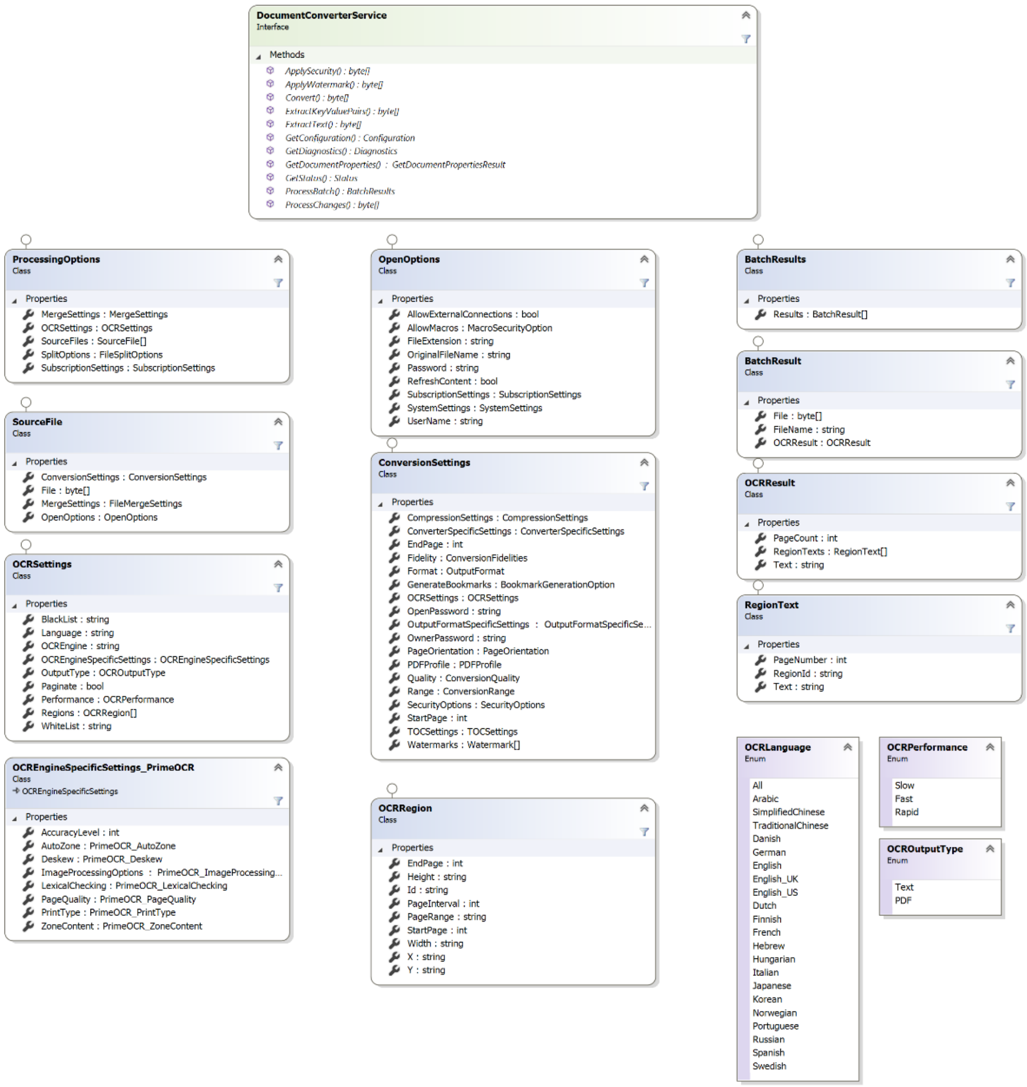
3.3.9 The BatchResult class

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 3.3.2. When OCR Text Extraction is carried out, this field will be empty.
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*.*


DocumentConverterService
Interface
Methods
ApplySecurity() : byte[]
ApplyWatermark() : byte[]
Convert() : byte[]
ExtractKeyValuePairs() : byte[]
ExtractText() : byte[]
GetConfiguration() : Configuration
GetDiagnostics() : Diagnostics
GetDocumentProperties() : GetDocumentPropertiesResult
GetStatus() : Status
ProcessBatch() : BatchResults
ProcessChanges() : byte[]
ProcessingOptions
Class
Properties
MergeSettings : MergeSettings
OCRSettings : OCRSettings
SourceFiles : SourceFile[]
SplitOptions : FileSplitOptions
SubscriptionSettings : SubscriptionSettings
SourceFile
Class
Properties
ConversionSettings : ConversionSettings
File : byte[]
MergeSettings : FileMergeSettings
OpenOptions : OpenOptions
OCRSettings
Class
Properties
BlackList : string
Language : string
OCREngine : string
OCREngineSpecificSettings : OCREngineSpecificSettings
OutputType : OCROutputType
Paginate : bool
Performance : OCRPerformance
Regions : OCRRegion[]
WhiteList : string
OCREngineSpecificSettings_PrimeOCR
Class
→ OCREngineSpecificSettings
Properties
AccuracyLevel : int
AutoZone : PrimeOCR_AutoZone
Deskew : PrimeOCR_Deskew
ImageProcessingOptions : PrimeOCR_ImageProcessing...
LexicalChecking : PrimeOCR_LexicalChecking
PageQuality : PrimeOCR_PageQuality
PrintType : PrimeOCR_PrintType
ZoneContent : PrimeOCR_ZoneContent
OpenOptions
Class
Properties
AllowExternalConnections : bool
AllowMacros : MacroSecurityOption
FileExtension : string
OriginalFileName : string
Password : string
RefreshContent : bool
SubscriptionSettings : SubscriptionSettings
SystemSettings : SystemSettings
UserName : string
ConversionSettings
Class
Properties
CompressionSettings : CompressionSettings
ConverterSpecificSettings : ConverterSpecificSettings
EndPage : int
Fidelity : ConversionFidelities
Format : OutputFormat
GenerateBookmarks : BookmarkGenerationOption
OCRSettings : OCRSettings
OpenPassword : string
OutputFormatSpecificSettings : OutputFormatSpecificSe...
OwnerPassword : string
PageOrientation : PageOrientation
PDFProfile : PDFProfile
Quality : ConversionQuality
Range : ConversionRange
SecurityOptions : SecurityOptions
StartPage : int
TOCSettings : TOCSettings
Watermarks : Watermark[]
OCRRegion
Class
Properties
EndPage : int
Height : string
Id : string
PageInterval : int
PageRange : string
StartPage : int
Width : string
X : string
Y : string
BatchResults
Class
Properties
Results : BatchResult[]
BatchResult
Class
Properties
File : byte[]
FileName : string
OCRResult : OCRResult
OCRResult
Class
Properties
PageCount : int
RegionTexts : RegionText[]
Text : string
RegionText
Class
Properties
PageNumber : int
RegionId : string
Text : string
OCRLanguage
Enum
All
Arabic
SimplifiedChinese
TraditionalChinese
Danish
German
English
English_UK
English_US
Dutch
Finnish
French
Hebrew
Hungarian
Italian
Japanese
Korean
Norwegian
Portuguese
Russian
Spanish
Swedish
OCRPerformance
Enum
Slow
Fast
Rapid
OCROutputType
Enum
Text
PDF

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	OCRPerformance	Indicate what performance / accuracy to use. *Slow* 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 *OCRLanguage* enumeration or custom values if custom character sets / languages have been defined.
WhiteList	String	Optional string of characters to limit recognition to. For example specify *1234567890* to only recognize numbers to prevent a 1 being recognized as *i or l.*
BlackList	String	Optional string of characters to skip recognition for. For example specify *1234567890* 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)?
OutputType	OCROutput Type	The kind of data to return, either extracted **Text**, an OCRed **PDF**, or both (in a single request). Values can be added up to combine.
OCREngine	String	The OCR Engine to use. Leave empty to use the default *Muhimbi* one or enter “GdPictureOCR” to use the GdPicture ORC one (included).
OCREngineSpecific Settings	OCREngine Specific Settings	Settings specific to the OCR Engine. Currently only used by the 3rd party PrimeOCR engine.

3.4.2 OCREngineSpecificSettings_PrimeOCR

Settings for the 3rd party PrimeOCR product, for full details see the PrimeOCR documentation.

Please note that PrimeOCR is not bundled with Muhimbi’s software and only available to customers of PrimeOCR. The following settings are specific to PrimeOCR.

Property	Type	Description
AccuracyLevel	int	The accuracy level of OCR.
Deskew	PrimeOCR_ Deskew	Options for de-skewing images before recognition.
ImagePorcessingOptions	PrimeOCR_ ImageProcessingOptions	Options for image preprocessing.
LexicalChecking	PrimeOCR_ LexicalChecking	Controls lexical checking of OCR results.
PageQuality	PrimeOCR_ PageQuality	Control how to deal with low quality input images.
PrintType	PrimeOCR_ PrintType	Provide details about the type of printer used to create the input file.
AutoZone	PrimeOCR_ AutoZone	Control auto zoning.
ZoneContent	PrimeOCR_ ZoneContent	Restricts the content of zones.

3.4.3 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 *StartPage* and *EndPage* properties.

3.4.4 The OCRResult class

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

Property	Type	Description
PageCount	int	The number of pages OCRed, useful for reporting purposes.
RegionTexts	RegionText[]	Individual results for the various regions specified in *OCRRegion*.
Text	string	The full textual result of OCR processing.

3.4.5 The RegionText class

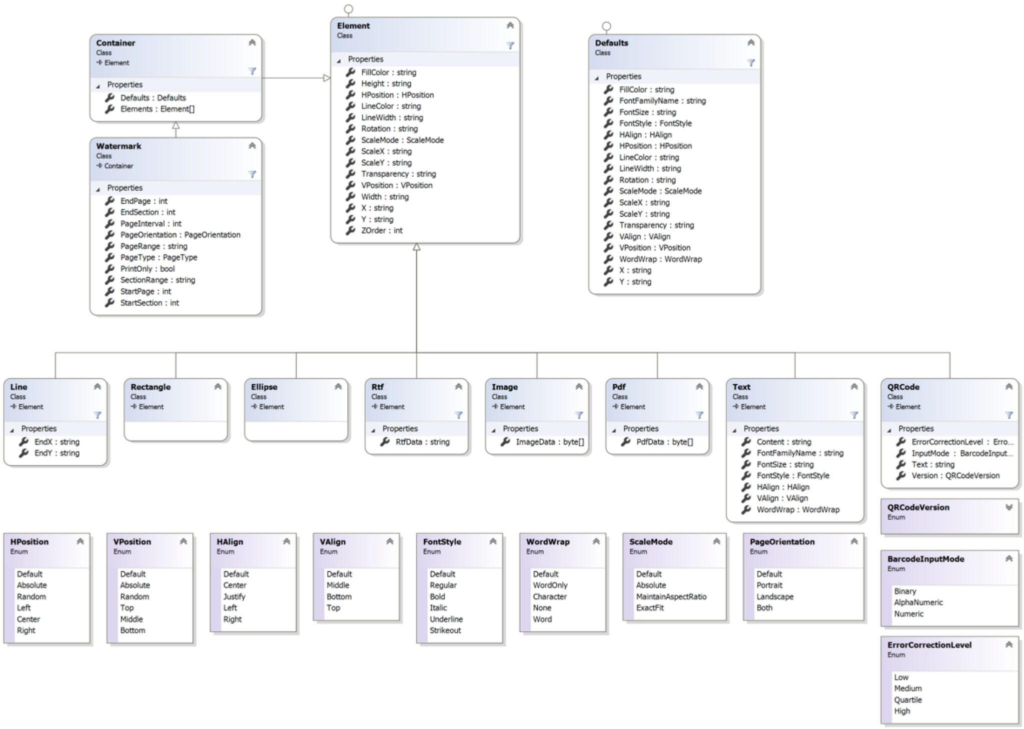
Text associated with a region is returned in an instance of the *RegionText* class.

Property	Type	Description
RegionId	String	The ID of the region the text belongs to, as specified in OCRRegion.Id.
PageNumber	int	The page number the text belongs to.
Text	String	The OCRed 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 the ConversionSettings class, for a code example see chapter 4.11.



3.5.1 The Watermark class

An instance of this class is passed to 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. *LineColor*, *alignment, transparency*, etc. For details see 3.5.4.
Elements	Element[]	A list of elements, e.g. *Text, Line* or *Image* 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. Use negative values to count from the back of the document (e.g -1 is last page, -2 is second to last page)
EndSection	Int	The last section in a Word or Excel document the watermark is visible.
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: *Portrait*, *Landscape* or *Both*.
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 *StartPage* and *EndPage* properties.
PageType	PageType	One or more page types for Word and Excel documents, eg. default|first|even.
PrintOnly	Bool	Should the watermark always be visible (False) or only when printing (True).
SectionRange	String	See StartSection, EndSection
StartPage	Int	The first page of the document the watermark applies to. Defaults to the first page. Use negative values to count from the back of the document (e.g -1 is last page, -2 is second to last page)
StartSection	Int	The first section in a Word or Excel document the watermark is visible.
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 *#rrggbb* or *#aarrggbb* format where *aa* represents the alpha / transparency.
Height	String	The height of the element. Note that this field is of type *string* 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 *#rrggbb* or *#aarrggbb* format where *aa* represents the alpha / transparency.
LineWidth	String	The width of the element's line. Note that this field is of type *string* 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 *string* 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 *string* 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 *string* to allow different scaling units to be specified in a future version.

Property	Type	Description
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 *string* 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 *string* 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 *string* 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. Rendered as plain text in PowerPoint.
- **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. This watermark type can only be applied to other PDF files.

- **QRCode:** Adds QRCode based barcodes to a document. The properties are as described below, for more details see this blog post.
    - **Text:** The content to embed in the QR code. This will need to match the specified *InputMode*.
    - **Version:** Over the years many different QR versions have been introduced. Select the one appropriate to your needs, either *Auto* or *Version01 – Version40*.
    - **InputMode:** Specify the appropriate mode for your content:
        - **Binary:** Any value including text, URLs etc.
        - **AlphaNumeric:** Numbers, (Upper case) characters and SPACE, $, %, *, +, -, ., /, :
        - **Numeric:** Numbers only
    - **ErrorCorrectionLevel:** Select the appropriate level for your needs: *Low Medium Quartile High*
- **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.
- **LinearBarcode:** Add traditional barcodes to a document. The key properties are as follows:
    - **Type:** The barcode type including Codabar, Code 11, Code 32, Code 39, Code 93, Code 128 (A/B/C), GS1-128.
    - **Content:** The content for the barcode, please make sure that the specified content is compatible with the data that may be stored in the selected barcode type.
    - **Check digit:** If relevant to the barcode type, calculate and encode the check-digit into the barcode.

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 *#rrggbb* or *#aarrggbb* format where *aa* 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. *FontStyle.Bold* | *FontStyle .Italic*.
HAlign	HAlign	Horizontal alignment of text stored in a *Text* element.
HPosition	HPosition	The horizontal position of the element.
LineColor	String	The color of the element's line in *#rrggbb* or *#aarrggbb* format where *aa* represents the alpha / transparency.
LineWidth	String	The width of the element's line. Note that this field is of type *string* 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 *string* 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 *string* 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.

Property	Type	Description
		Note that this field is of type *string* 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 *Text* element.
VPosition	VPosition	The vertical position of the element.
WordWrap	WordWrap	The word wrapping behaviour of text stored in a *Text* element.
X	String	The x-coordinate of the element. Note that this field is of type *string* 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 *string* 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 allows 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:

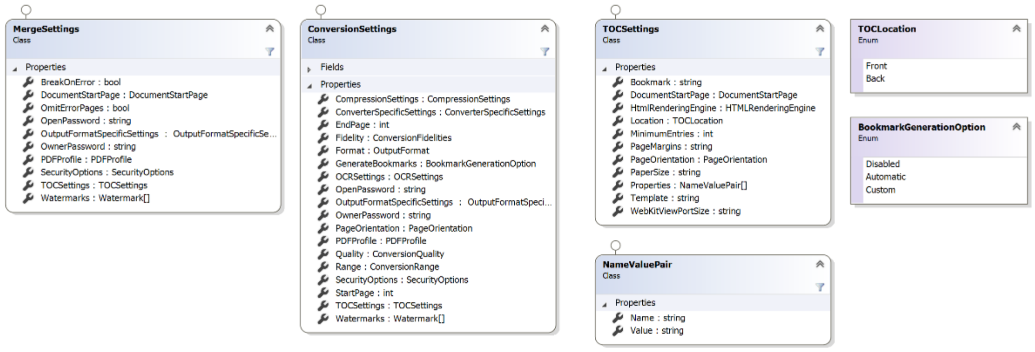
- **{LONG_DATE}:** The long representation of the current date, equivalent in C# to *DateTime.Now.ToLongDateString()*
- **{LONG_TIME}:** The long representation of the current time, equivalent in C# to *DateTime.Now.ToLongTimeString()*
- **{DATE}:** The short representation of the current date, equivalent in C# to *DateTime.Now.ToShortDateString()*
- **{TIME}:** The short representation of the current time, equivalent in C# to *DateTime.Now.ToLongTimeString()*
- **{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.


MergeSettings
Class
Properties
BreakOnError : bool
DocumentStartPage : DocumentStartPage
OmitErrorPages : bool
OpenPassword : string
OutputFormatSpecificSettings : OutputFormatSpecificSe...
OwnerPassword : string
PDFProfile : PDFProfile
SecurityOptions : SecurityOptions
TOCSettings : TOCSettings
Watermarks : Watermark[]
ConversionSettings
Class
Fields
Properties
CompressionSettings : CompressionSettings
ConverterSpecificSettings : ConverterSpecificSettings
EndPage : int
Fidelity : ConversionFidelities
Format : OutputFormat
GenerateBookmarks : BookmarkGenerationOption
OCRSettings : OCRSettings
OpenPassword : string
OutputFormatSpecificSettings : OutputFormatSpeci...
OwnerPassword : string
PageOrientation : PageOrientation
PDFProfile : PDFProfile
Quality : ConversionQuality
Range : ConversionRange
SecurityOptions : SecurityOptions
StartPage : int
TOCSettings : TOCSettings
Watermarks : Watermark[]
TOCSettings
Class
Properties
Bookmark : string
DocumentStartPage : DocumentStartPage
HtmlRenderingEngine : HTMLRenderingEngine
Location : TOCLocation
MinimumEntries : int
PageMargins : string
PageOrientation : PageOrientation
PaperSize : string
Properties : NameValuePair[]
Template : string
WebKitViewPortSize : string
TOCLocation
Enum
Front
Back
BookmarkGenerationOption
Enum
Disabled
Automatic
Custom
NameValuePair
Class
Properties
Name : string
Value : string

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.
DocumentStartPage	Document StartPage	When printing double sided it is often desirable to let the main document start on (usually) the right hand page. Use this property to control on what page the main document (after the TOC) starts.
HTMLRenderingEngine	HTMLRenderingEngine	Specify the rendering engine for converting html content • IE - Use Internet Explorer based converter (legacy use only) • WebKit - Use WebKit (Chrome like) converter
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	The margin / border around the generated TOC. It defaults to a uniform half inch margin. One or four {value}{dim} components separated by commas (,) where • {value} is a numerical value • {dim} is the dimension which can be 'mm', 'in.' or 'inches'. (Defaults to inches when nothing is specified) When multiple values are specified then the sequence is: left, top, right and bottom. Example: "12mm, 24mm, 12mm, 24mm"
PageOrientation	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.
PaperSize	String	The optional paper size to use for the TOC. Either: • 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 (decimal separator must be colon '.') - {dim} is the dimension which can be 'mm', 'in.' or 'inches'. (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	NameValuePair[]	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.
WebKitViewPortSize	String	Specify the viewport size (for webkit based converter only) • Paper - Dimensions specified in PaperSize minus the margins in PageMargin. • w x h - In pixels. Example "1280 x 1024"

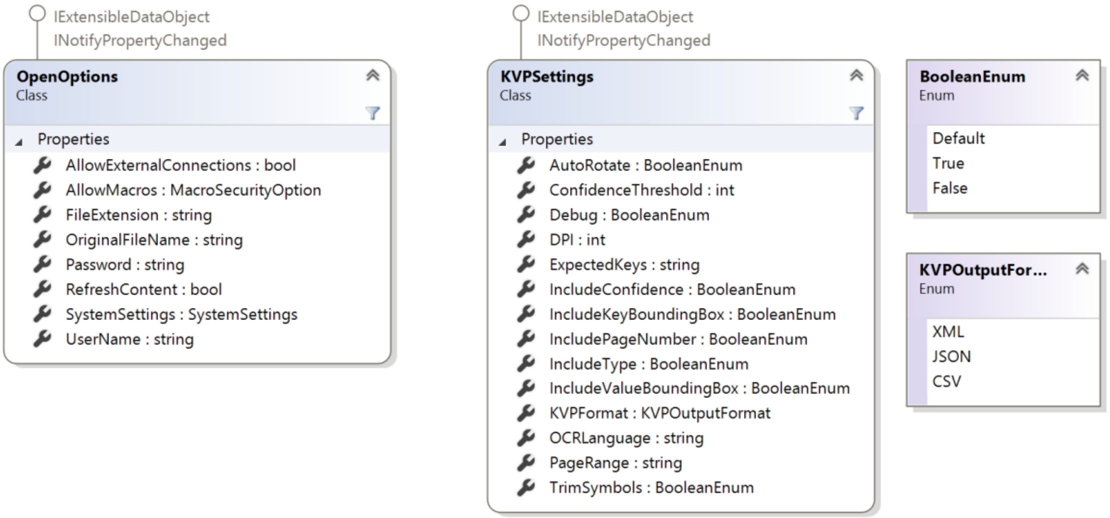
3.6.2 The NameValuePair class

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 Key-Value Pairs extraction

The *Muhimbi Document Conversion Service* can extract Key-Value Pairs (KVP) from PDF documents. It can extract them for specific page ranges and can include the bounding boxes for both the key item and the value item. The extraction returns a file in one of CSV, JSON and XML format.



3.7.1 The KVPSettings class

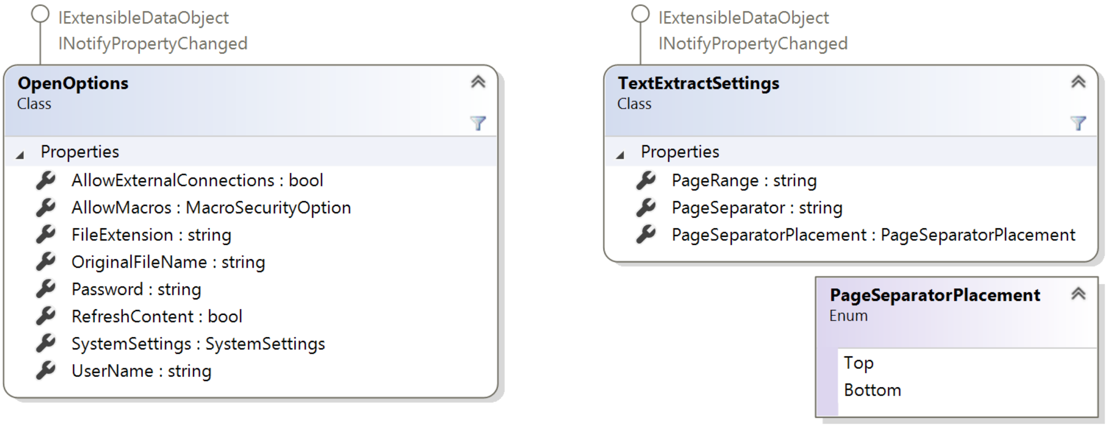
The class contains the KVP specific settings and an instance is passed together with the file and an OpenOptions instance to the ExtractKeyValuePairs method.

Property	Type	Note
Debug	BooleanEnum	Debug mode, provides additional logging
OCRLanguage	string	Three character language codes. Multiple languages separated by '+'. For example *eng* or *eng+jpn*
DPI	int	DPI used for rendering the pages as part of the extraction process.
KVPFormat	KVPOutput Format	Key Value Pair format. It can be XML, JSON or CSV
TrimSymbols	BooleanEnum	Setting this to 'True' will remove any symbols from the start/end of values, with the exception of the hash '#' or period '.' symbols.
IncludeKeyBoundingBox	BooleanEnum	Setting this to True will include the bounding box values for the key in the output.
IncludeValueBoundingBox	BooleanEnum	Setting this to True will include the bounding box values for the value in the output.
IncludePageNumber	BooleanEnum	Setting this to True will include the page number of the key value pair in the output.
IncludeConfidence	BooleanEnum	Setting this to True will include the confidence score of the key value pair in the output. Confidence is measured between 0 (no confidence) and 100 (full confidence).

IncludeType	BooleanEnum	Setting this to True will include the data type of the key value pair in the output.
ConfidenceThreahold	int	The value of confidence (0-100) that a KVP must reach to be included in the output. Results under this confidence threshold will be discarded.
ExpectedKeys	string	JSON string containing the expected keys and their associated synonyms.
AutoRotate	BooleanEnum	Automatically rotate the page if the text does not have the correct orientation.
PageRange	string	Use the string of "1-5" for pages 1 to 5, or use the string of "1,5,6" to specify pages 1 and 5 and 6. You can use the string of "1,5,8-12" to specify pages 1, 5, 8 and all pages from page 8 to page 12, etc.

3.8 Text Extraction

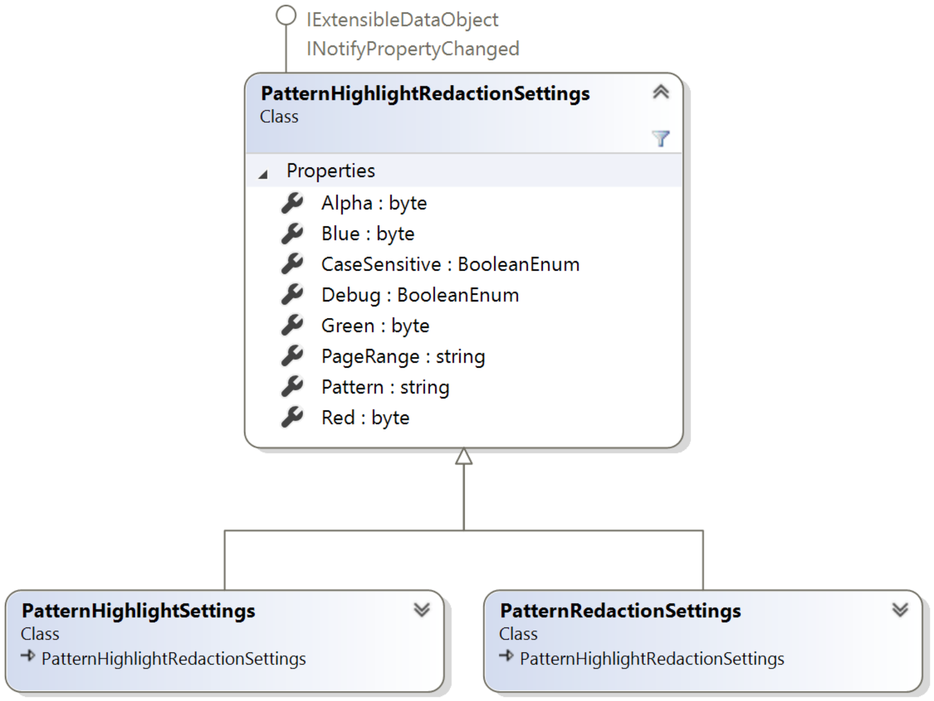
It is possible to extract text from text-based PDF files. The text extraction specific settings (an instance of the TextExtractSettings class) should be passed together with the file and an OpenOptions instance to the ExtractText method.



3.8.1 The TextExtractSettings class

Property	Type	Notes
PageRange	string	Range of pages for text extraction
PageSeparator	string	The text to be added to extracted text as a page separator
PageSeparatorPlacement	PageSeparatorPlacement	Top: adds page separator at the top of the page Bottom: adds page separator at the bottom of the page

3.9 Pattern Redaction and Highlighting



3.9.1 Pattern Redaction

Redact text in a PDF file based on regex patterns.

Property	Type	Notes
Debug	BooleanEnum	Debug mode gives additional logging information
Pattern	string	Regular expression pattern for the text to be redaction
CaseSensitive	BooleanEnum	Is the regular expression case sensitive
Red	byte	Red component of the redaction color. Range 0 - 255.
Green	byte	Green component of the redaction color. Range 0 - 255.
Blue	byte	Blue component of the redaction color. Range 0 - 255.

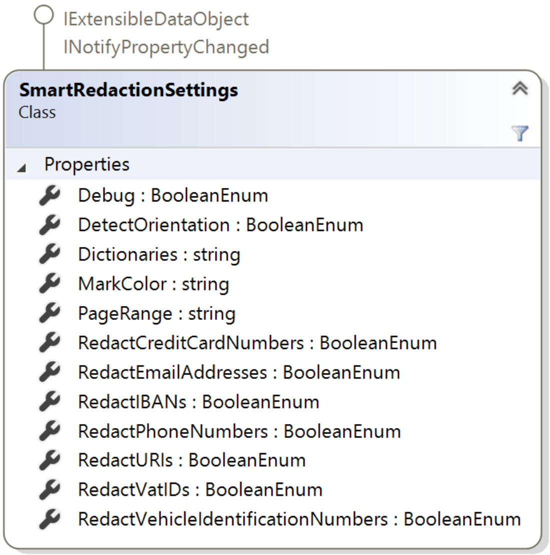
3.9.2 Pattern Highlighting

Highlights text in a PDF file based on regex patterns.

Property	Type	Notes
Debug	BooleanEnum	Debug mode gives additional logging information
Pattern	string	Regular expression pattern for the text to be redaction
CaseSensitive	BooleanEnum	Is the regular expression case sensitive
Red	byte	Red component of the redaction color. Range 0 - 255.
Green	byte	Green component of the redaction color. Range 0 - 255.
Blue	byte	Blue component of the redaction color. Range 0 - 255.
Alpha	Byte	Alpha value, Range 0 – 255

3.10 Smart Redaction

Use Smart Redaction to identify and redact selected sensitive information in the input document.



IExtensibleDataObject
INotifyPropertyChanged

SmartRedactionSettings
Class

Properties

- Debug : BooleanEnum
- DetectOrientation : BooleanEnum
- Dictionaries : string
- MarkColor : string
- PageRange : string
- RedactCreditCardNumbers : BooleanEnum
- RedactEmailAddresses : BooleanEnum
- RedactIBANs : BooleanEnum
- RedactPhoneNumbers : BooleanEnum
- RedactURIs : BooleanEnum
- RedactVatIDs : BooleanEnum
- RedactVehicleIdentificationNumbers : BooleanEnum

Property	Type	Notes
Debug	BooleanEnum	Debug mode gives additional logging information
RedactCreditCardNumbers	BooleanEnum	Specifies whether credit card numbers will be redacted.
RedactEmailAddresses	BooleanEnum	Specifies whether email addresses will be redacted.
RedactIBANs	BooleanEnum	Specifies whether IBANs will be redacted.
RedactPhoneNumbers	BooleanEnum	Specifies whether phone numbers will be redacted.
RedactURIs	BooleanEnum	Specifies whether URIs will be redacted.
RedactVATIDs	BooleanEnum	Specifies whether VAT IDs will be redacted.
RedactVehicle IdentificationNumbers	BooleanEnum	Specifies whether Vehicle Identification Numbers will be redacted.
MarkColor	string	The color used to cover redacted information. The options are: Black, Transparent, Aqua, Teal, Navy, Yellow, Olive, Lime, Blue, Fuchsia, Purple, Red, Maroon, White, Gray, Silver, Green
Dictionaries	string	List of language codes, linked by ‘+’, for example: ”ENG+FRA”
DetectOrientation	BooleanEnum	Specifies whether orientation will not be detected automatically.
PageRange	string	Use the string of "1-5" for pages 1 to 5, or use the string of "1,5,6" to specify pages 1 and 5 and 6. You can use the string of "1,5,8-12" to specify pages 1, 5, 8 and all pages from page 8 to page 12, etc.

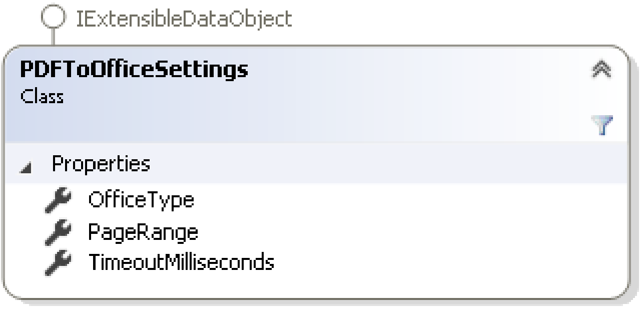
3.11 PDFToOffice (Preview)

Use PDFToOffice to convert a PDF file to an Office format file.

This feature is released as preview. Although most use cases will generate acceptable results there is still on-going development on the GdPicture engine.

This feature is supplied as-is, and we welcome your feedback to improve the conversion fidelity.

NOTE: though SVG is a valid output type, due to SVG format limitations only the first page (see also PageRange) will be converted to SVG. To convert a multi-page PDF to SVG please use the PDFToSVG operation.



IExtensibleDataObject

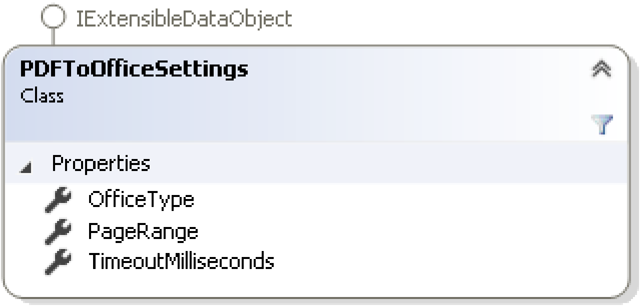
PDFToOfficeSettings
Class

Properties
- OfficeType
- PageRange
- TimeoutMilliseconds

Property	Type	Notes
OfficeType	Office output types	The output type from the list DOCX, PPTX, XLSX and SVG. NOTE: though SVG is a valid output type, due to SVG format limitations only the first page (see also PageRange) will be converted to SVG. To convert a multi-page PDF to SVG please use the PDFToSVG operation.
PageRange	String	Page range to be converted to an Office file.
TimeoutMilliseconds	int	Timeout in milliseconds, if the operation has not been completed after this time, it is assumed to have failed.

3.12 PDFToSVG

Convert a multi-page PDF file to a collection of SVG files.



IExtensibleDataObject

PDFToOfficeSettings
Class

Properties
- OfficeType
- PageRange
- TimeoutMilliseconds

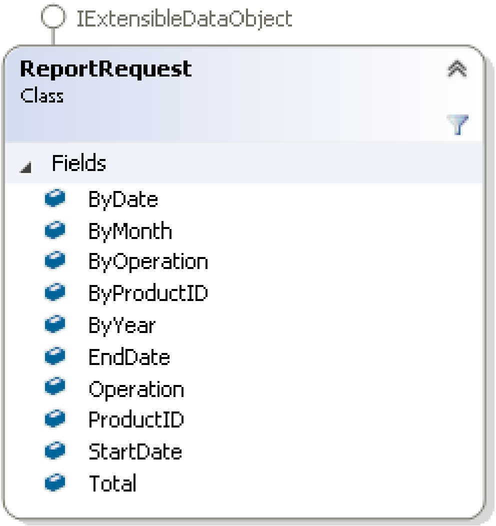
See PDFToOffice for details.

Note: only SVG is a valid OfficeType for this operation.

3.13 Instrumentation (CSV and PDF reports)

From version 12.2 onwards, PDF Converter records usage information. This information is recorded locally.

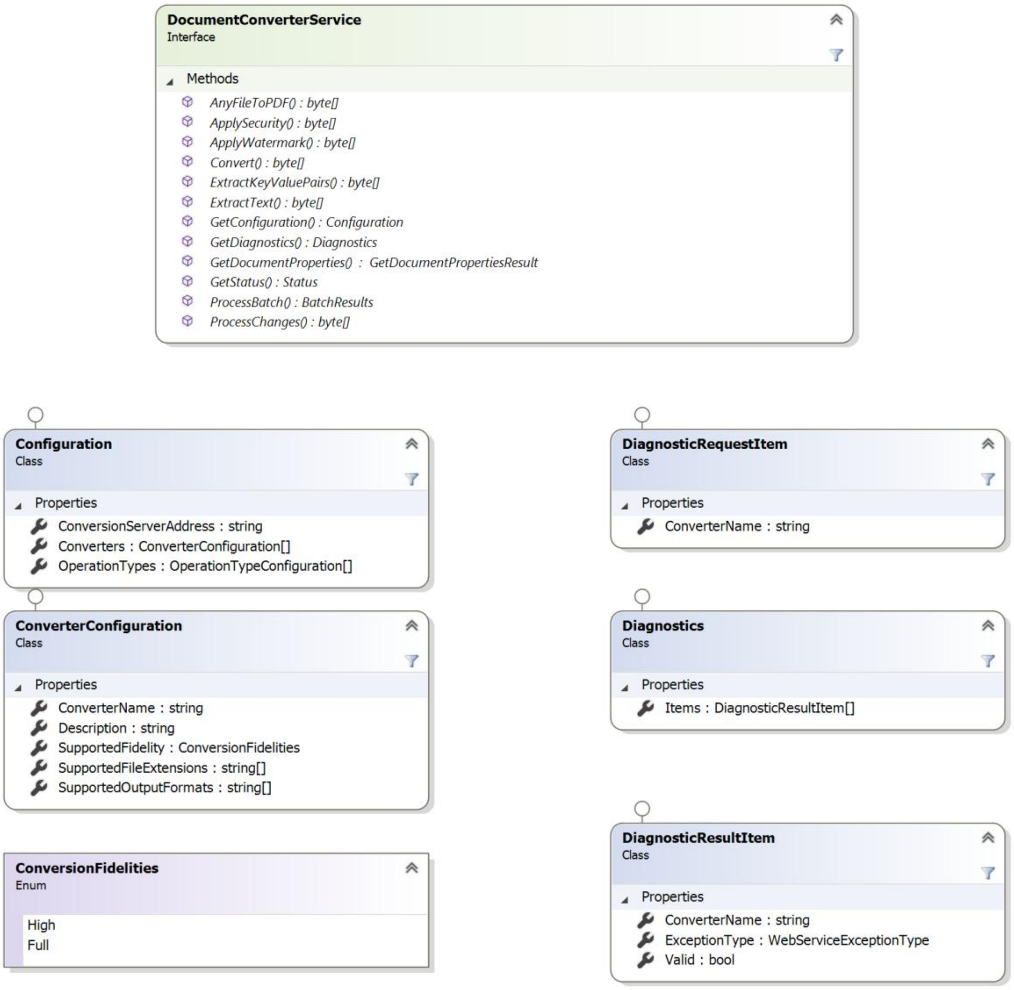
Reports can be requested, either via the Diagnostic Tool or by code.



Option	Description
Total	Group everything
By Year	Group the usage data by year
By Month	Group the usage data by year and month
By Date	Group the usage data by year, month and day
By Operation	Group the usage data by operation
By Product ID	Group the usage data by product id
Start Date	Only return data for operations that start at or after the StartDate value supplied when the check box is ticked
End Date	Only return data for operations that start at or before the EndDate value supplied when the check box is ticked
Operation	Only return data for the specified operation
ProductID	The productID is always 4 for PDF Converter

3.14 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.


DocumentConverterService
Interface
Methods
AnyFileToPDF() : byte[]
ApplySecurity() : byte[]
ApplyWatermark() : byte[]
Convert() : byte[]
ExtractKeyValuePairs() : byte[]
ExtractText() : byte[]
GetConfiguration() : Configuration
GetDiagnostics() : Diagnostics
GetDocumentProperties() : GetDocumentPropertiesResult
GetStatus() : Status
ProcessBatch() : BatchResults
ProcessChanges() : byte[]
Configuration
Class
Properties
ConversionServerAddress : string
Converters : ConverterConfiguration[]
OperationTypes : OperationTypeConfiguration[]
ConverterConfiguration
Class
Properties
ConverterName : string
Description : string
SupportedFidelity : ConversionFidelities
SupportedFileExtensions : string[]
SupportedOutputFormats : string[]
ConversionFidelities
Enum
High
Full
DiagnosticRequestItem
Class
Properties
ConverterName : string
Diagnostics
Class
Properties
Items : DiagnosticResultItem[]
DiagnosticResultItem
Class
Properties
ConverterName : string
ExceptionType : WebServiceExceptionType
Valid : bool

3.14.1 Retrieving Configuration settings

The *GetConfguration* 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 supplied with the product as well as any custom converters.
OperationTypes	OperationType Configuration	DO NOT USE, For Muhimbi internal use only.

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.

3.14.2 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:

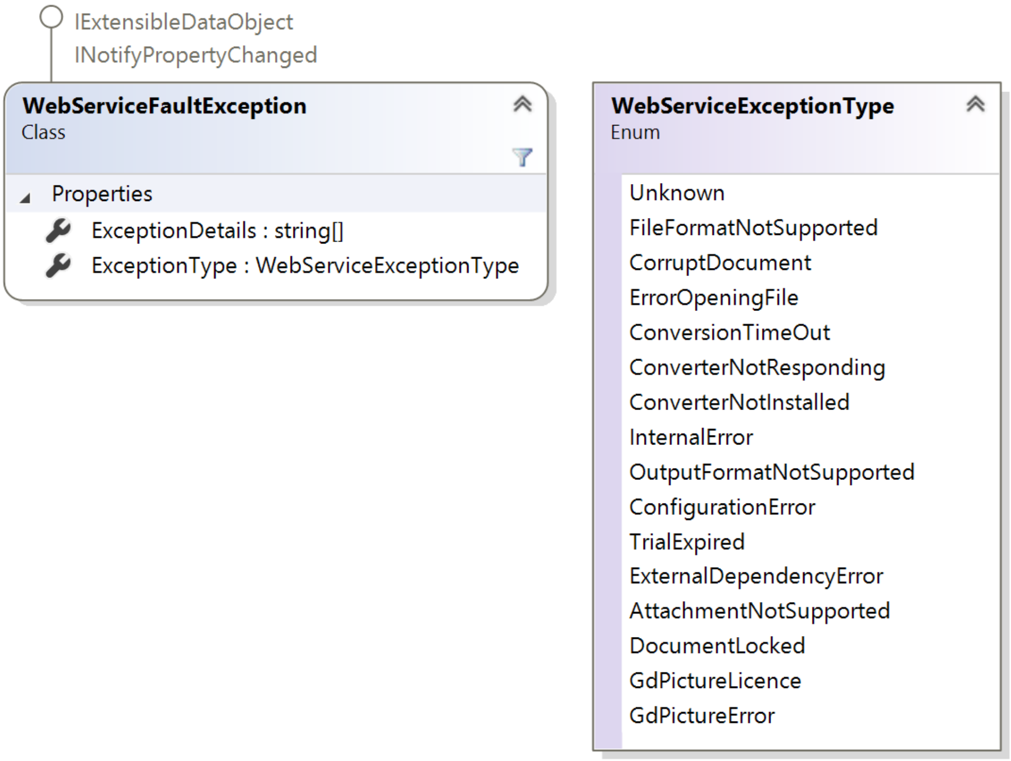
Property	Type	Description
Items	Diagnostic ResultItem[]	An array of items that holds the results.

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. *ConverterNotInstalled*)
Valid	Bool	A flag indicating if the converter is valid (No errors encountered during diagnosis).

3.15 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

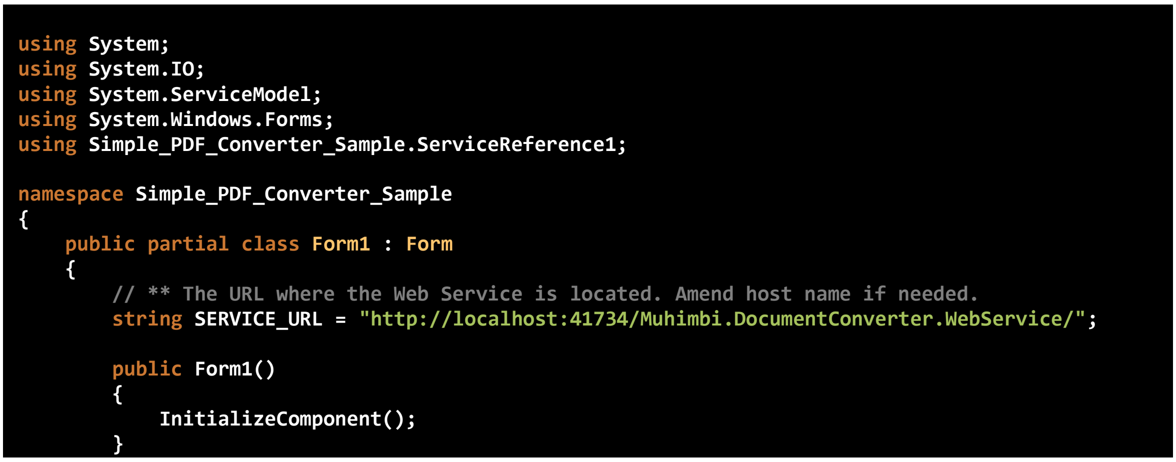
Listed below is a basic example of how to convert a document to PDF format using a simple *WinForms* application. For a more comprehensive example see the .NET Sample code installed alongside each copy of the MDCS. Use the Start Menu to open the appropriate folder or visit our GitHub area.

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

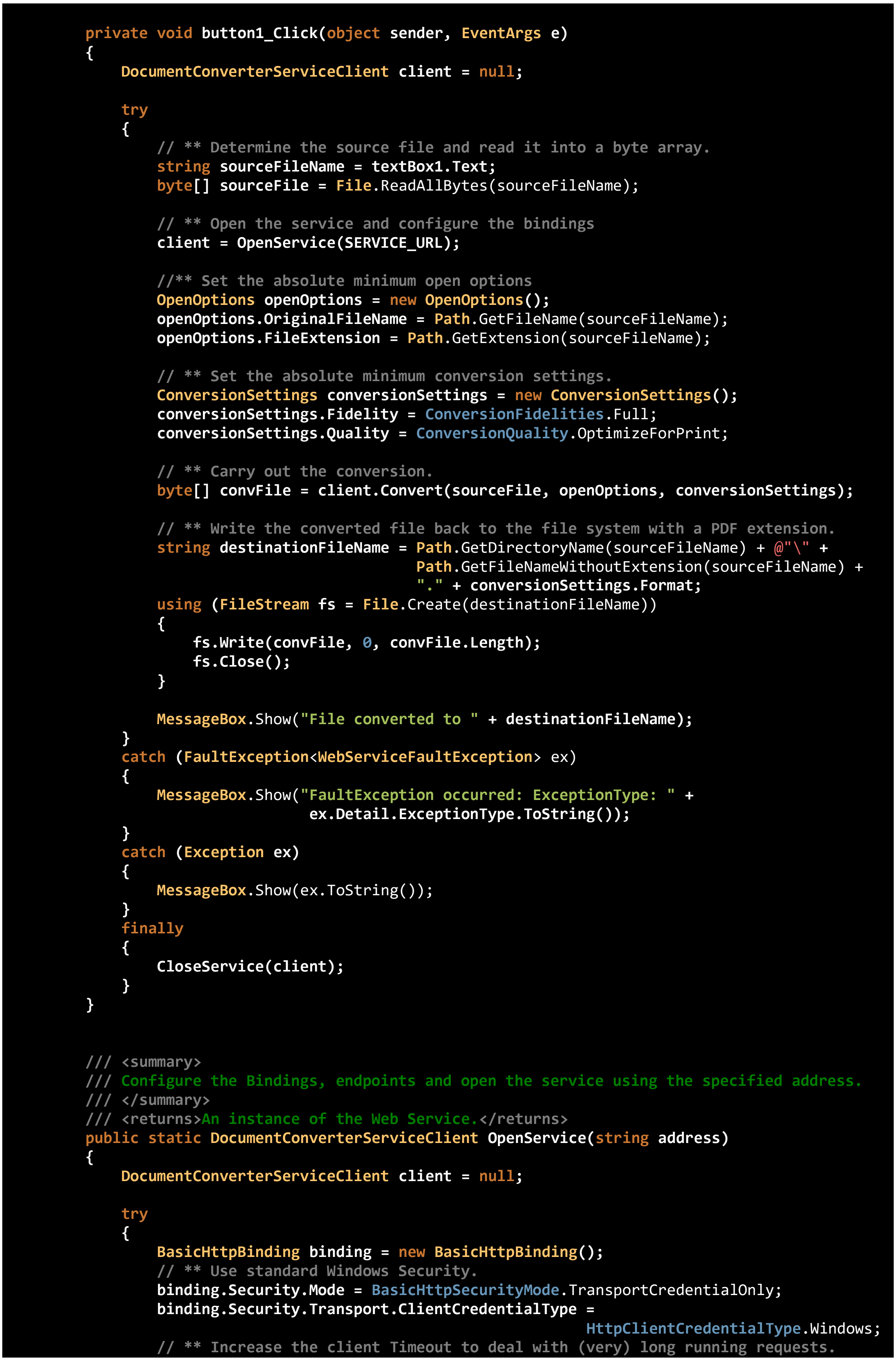
https://www.muhimbi.com/blog/converting-office-files-to-pdf-format-using-a-web-services-based-interface/

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.MaxStringContentLength = 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 https://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 or visit our GitHub area.

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.

```java
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 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();
  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 and on GitHub .

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.Document Converter.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,
```

```java
    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("." + 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

To compile your code does depend 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 (<install location>\Muhimbi Document Converter\Sample Code).

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 HTM:

```
<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, Compress, 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
<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
<?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:

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

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.

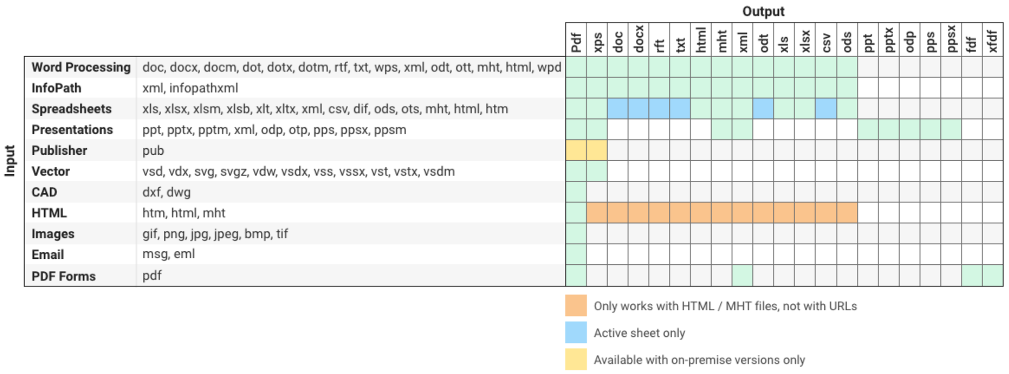
4.6 Cross-Converting between document types

Although the product names 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 number of legacy .DOC (Office 97-2003 format) files, but your company now requires all files to be saved in the more modern, and open, Office Open XML .DOCX (Office 2007+) formats. By using the Muhimbi PDF Converter you can convert between these formats automatically using a simple web service call using Java or .NET.

Conversion in the other direction is possible as well. 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	rft	txt	html	mht	xml	odt	xls	xlsx	csv	ods	ppt	pptx	odp	pps	ppsx	fdf	xfdf
	Word Processing	doc, docx, docm, dot, dotx, dotm, rtf, txt, wps, xml, odt, ott, mht, html, wpd	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩							
	InfoPath	xml, infopathxml	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩							
	Spreadsheets	xls, xlsx, xlsm, xlsb, xlt, xltx, xml, csv, dif, ods, ots, mht, html, htm	🟩	🟩	🟦	🟦	🟦	🟦	🟩	🟩	🟩	🟦	🟩	🟩	🟦	🟩							
	Presentations	ppt, pptx, pptm, xml, odp, otp, pps, ppsx, ppsm	🟩	🟩						🟩	🟩						🟩	🟩	🟩	🟩	🟩		
	Publisher	pub	🟨	🟨																			
	Vector	vsd, vdx, svg, svgz, vdw, vsdx, vss, vssx, vst, vstx, vsdm	🟩	🟩																			
	CAD	dxf, dwg	🟩																				
	HTML	htm, html, mht	🟩	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧							
	Images	gif, png, jpg, jpeg, bmp, tif	🟩																				
	Email	msg, eml	🟩																				
	PDF Forms	pdf	🟩								🟩											🟩	🟩

🟧 Only works with HTML / MHT files, not with URLs

🟦 Active sheet only

🟨 Available with on-premise versions 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. PDF forms data can be extracted by converting PDF to *fdf, xfdf,* and *xml*. For details see this blog post.
4. 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:

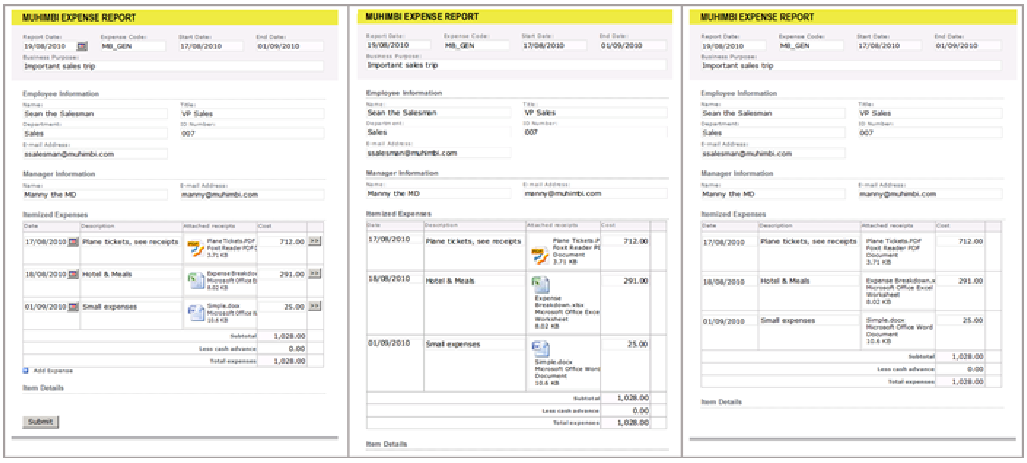
1. **Attachments**: When converting an InfoPath form to PDF the software also converts all attachments and merges them into the main PDF. This is possible because you can represent almost any file format in PDF and merge them together. Unfortunately this is not possible when converting to HTML, MS-Word 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.



MUHIMBI EXPENSE REPORT

Report Date:	Expense Code:	Start Date:	End Date:
19/08/2010	MB_GEN	17/08/2010	01/09/2010

Business Purpose: Important sales trip

Employee Information

Name: Sean the Salesman	Title: VP Sales
Department: Sales	ID Number: 007
E-mail Address: ssalesman@muhimbi.com	

Manager Information

Name: Manny the MD	E-mail Address: manny@muhimbi.com

Itemized Expenses

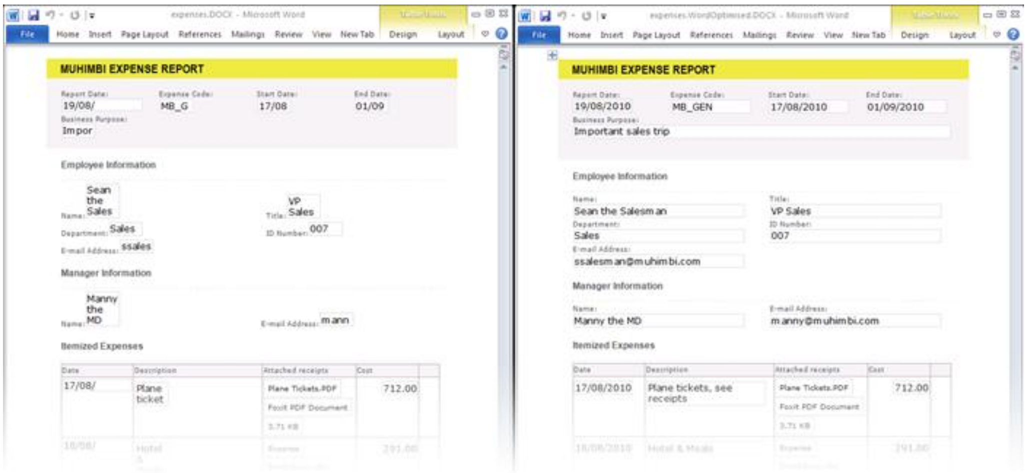
Date	Description	Attached receipts	Cost
17/08/2010	Plane tickets, see receipts	Plane Tickets.PDF Foxit Reader PDF Document 3.71 KB	712.00
18/08/2010	Hotel & Meals	Expense Breakdown.xlsx Microsoft Office Excel Worksheet 8.02 KB	291.00
01/09/2010	Small expenses	Simple.docx Microsoft Office Word Document 10.6 KB	25.00
		Subtotal	1,028.00
		Less cash advance	0.00
		Total expenses	1,028.00

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.



MUHIMBI EXPENSE REPORT

Report Date: 19/08/ Expense Code: MB_G Start Date: 17/08 End Date: 01/09
Business Purpose: Impor
Employee Information
Name: Sean the Sales Title: VP Sales
Department: Sales ID Number: 007
E-mail Address: ssales
Manager Information
Name: Manny the MD E-mail Address: mann
Itemized Expenses

Date	Description	Attached receipts	Cost
17/08/	Plane ticket	Plane Tickets.PDF Foxit PDF Document 3.71 KB	712.00

MUHIMBI EXPENSE REPORT

Report Date: 19/08/2010 Expense Code: MB_GEN Start Date: 17/08/2010 End Date: 01/09/2010
Business Purpose: Important sales trip
Employee Information
Name: Sean the Salesman Title: VP Sales
Department: Sales ID Number: 007
E-mail Address: ssalesman@muhimbi.com
Manager Information
Name: Manny the MD E-mail Address: manny@muhimbi.com
Itemized Expenses

Date	Description	Attached receipts	Cost
17/08/2010	Plane tickets, see receipts	Plane Tickets.PDF Foxit PDF Document 3.71 KB	712.00

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

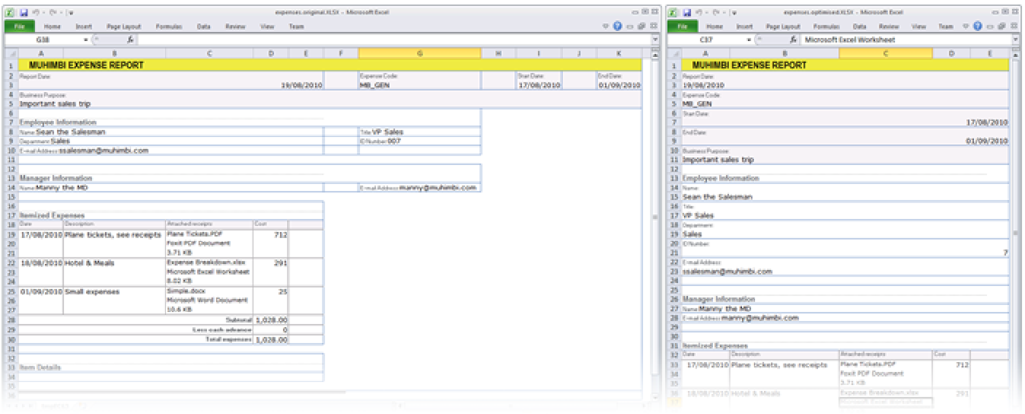
Looking at the 'before optimisation' conversion results in the image displayed above, there are 2 things that stand out:

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.


expenses.original.XLSX - Microsoft Excel
File
Home
Insert
Page Layout
Formulas
Data
Review
View
Team
G38
expenses.optimised.XLSX - Microsoft Excel
C37
Microsoft Excel Worksheet

MUHIMBI EXPENSE REPORT

Report Date 19/08/2010	Expense Code MB_GEN	Start Date 17/08/2010	End Date 01/09/2010

Business Purpose
Important sales trip

Employee Information

Name Sean the Salesman	Title VP Sales
Department Sales	ID Number 007
E-mail Address ssalesman@muhimbi.com	

Manager Information

Name Manny the MD	E-mail Address manny@muhimbi.com

Itemized Expenses

Date	Description	Attached receipts	Cost
17/08/2010	Plane tickets, see receipts	Plane Tickets.PDF Foxit PDF Document 3.71 KB	712
18/08/2010	Hotel & Meals	Expense Breakdown.xlsx Microsoft Excel Worksheet 8.02 KB	291
01/09/2010	Small expenses	Simple.docx Microsoft Word Document 10.6 KB	25
		Subtotal	1,028.00
		Less cash advance	0
		Total expenses	1,028.00

Item Details

MUHIMBI EXPENSE REPORT
Report Date
19/08/2010
Expense Code
MB_GEN
Start Date
17/08/2010
End Date
01/09/2010
Business Purpose
Important sales trip

Employee Information
Name
Sean the Salesman
Title
VP Sales
Department
Sales
ID Number
7
E-mail Address
ssalesman@muhimbi.com

Manager Information
Name Manny the MD
E-mail Address manny@muhimbi.com

Itemized Expenses

Date	Description	Attached receipts	Cost
17/08/2010	Plane tickets, see receipts	Plane Tickets.PDF Foxit PDF Document 3.71 KB	712
18/08/2010	Hotel & Meals	Expense Breakdown.xlsx Microsoft Excel Worksheet	291

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:

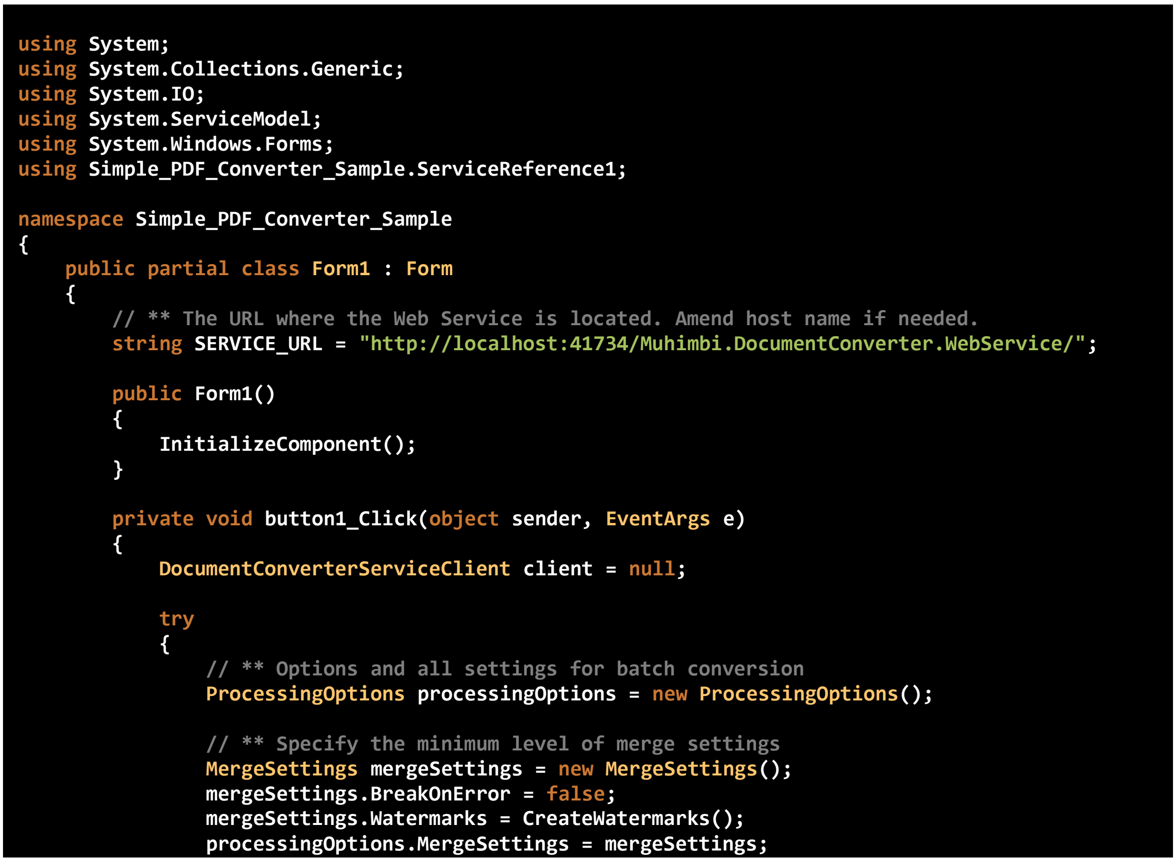
1. **Column Widths**: As Excel uses a cell / grid based approach it is not possible to mix different column widths. The information in the form's header requires different column width and spans than the columns used in the repeating table further down the page. By changing the horizontally oriented 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.

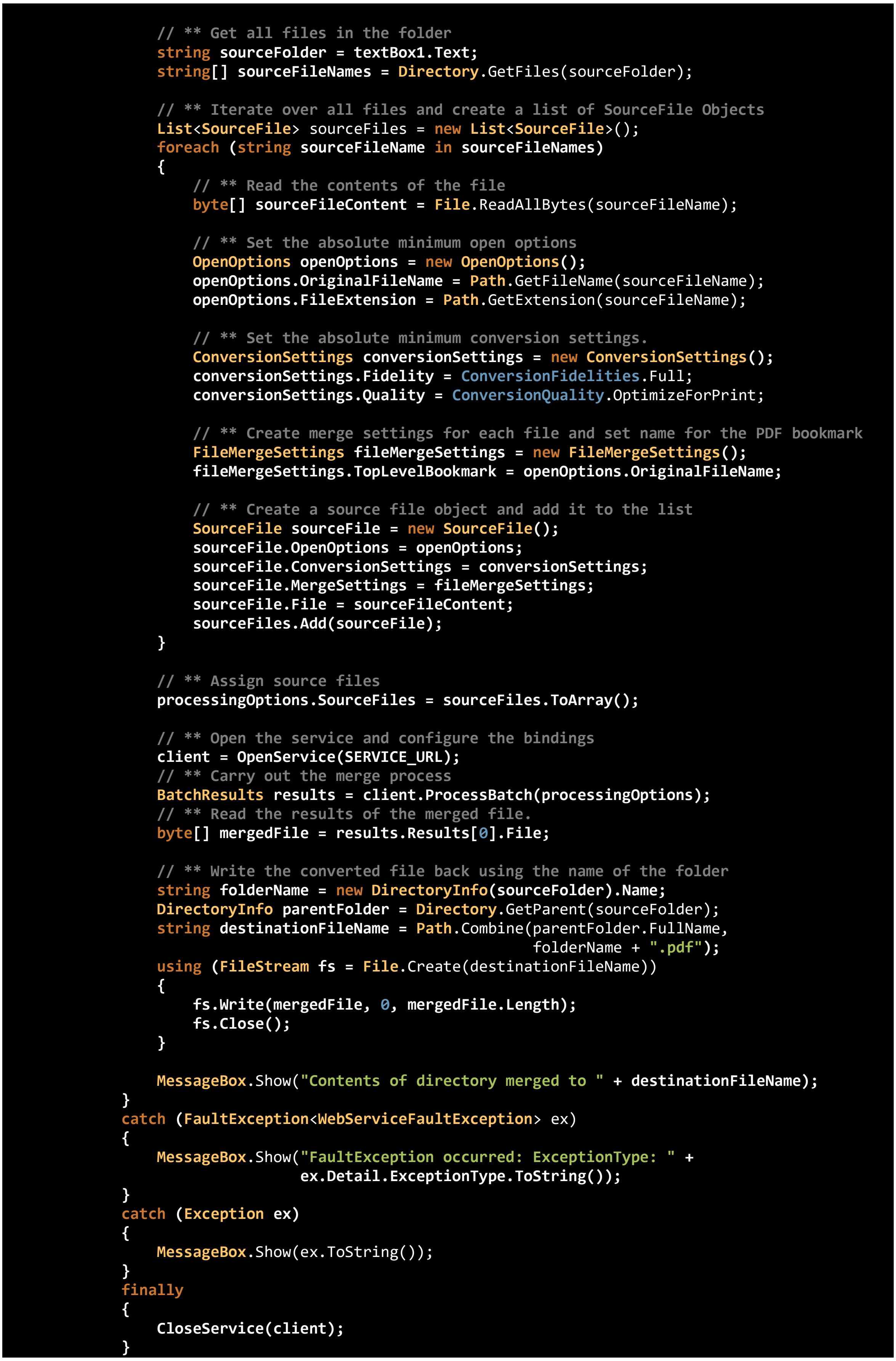
4.7 Merging multiple files into a single PDF using .NET

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*.

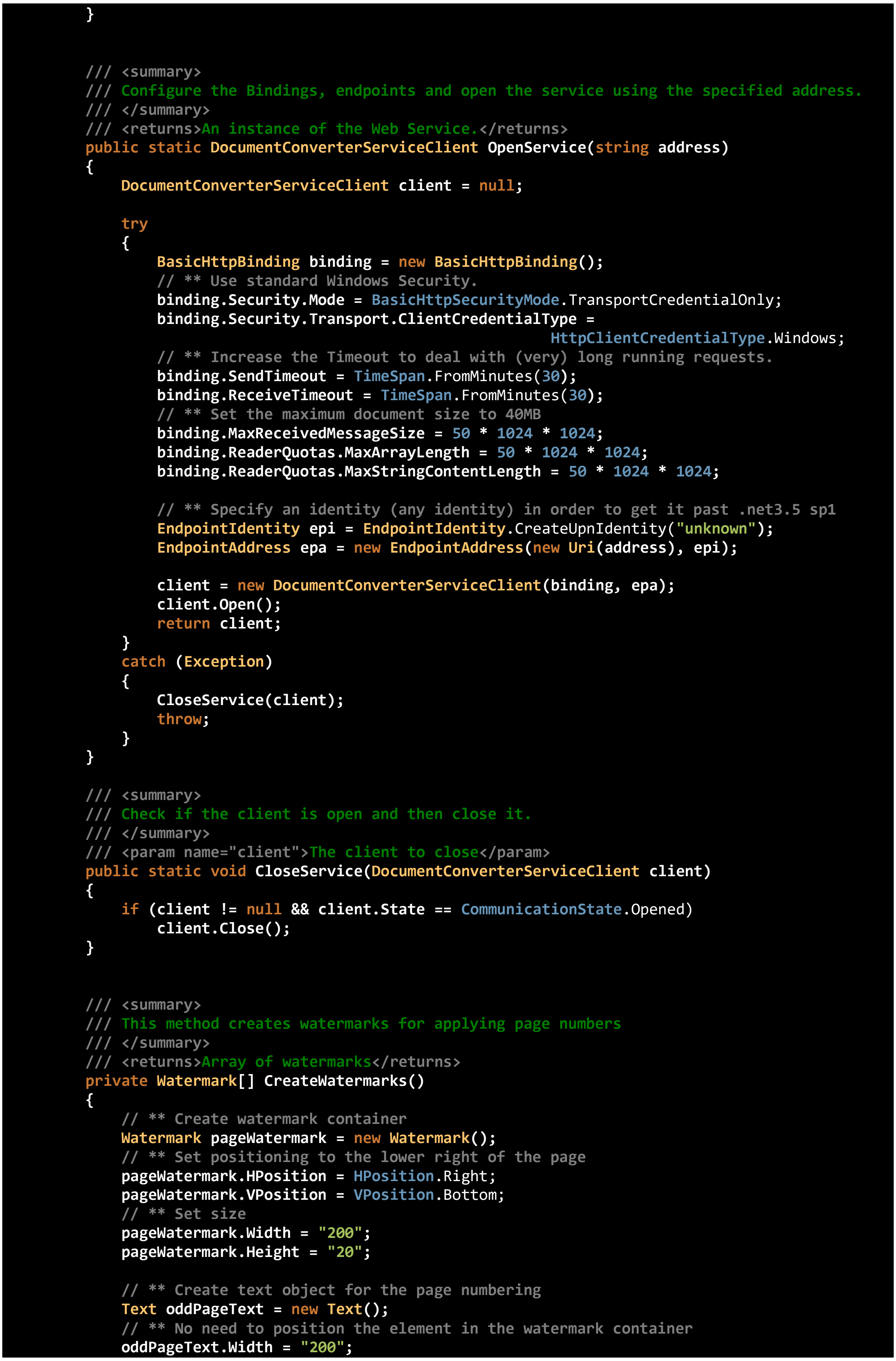
1. Start a new Visual Studio project and create the project type of your choice. In this example we are using a standard *.net 3.0* project of type *Windows Forms Application*. Name it ‘Simple PDF Converter Sample’.
2. Add a *TextBox* and *Button* control to the form. Accept the default names of *textBox1* 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:



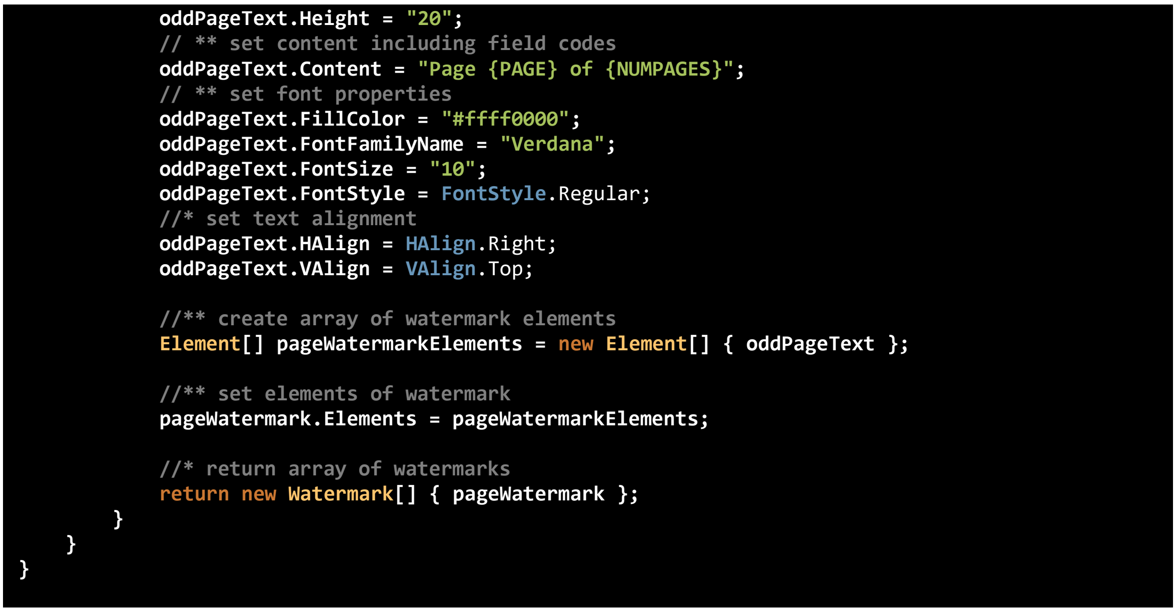
```
using System;
using System.Collections.Generic;
using System.IO;
using System.ServiceModel;
using System.Windows.Forms;
using Simple_PDF_Converter_Sample.ServiceReference1;

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.MaxStringContentLength = 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 https://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 (<install location>\Muhimbi Document Converter\Sample 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 How to use Java to Combine Multiple PDF Files (muhimbi.com) as well as on GitHub.

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

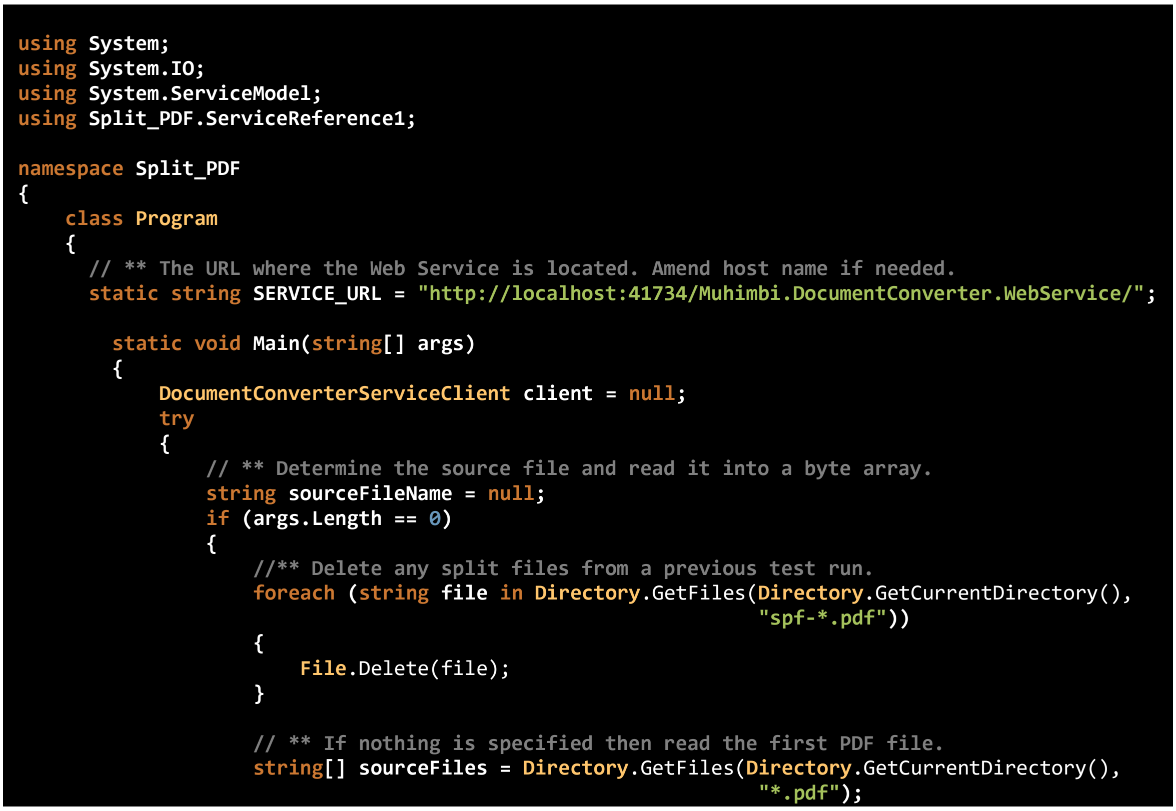
```java
package com.muhimbi.app;

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");
        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

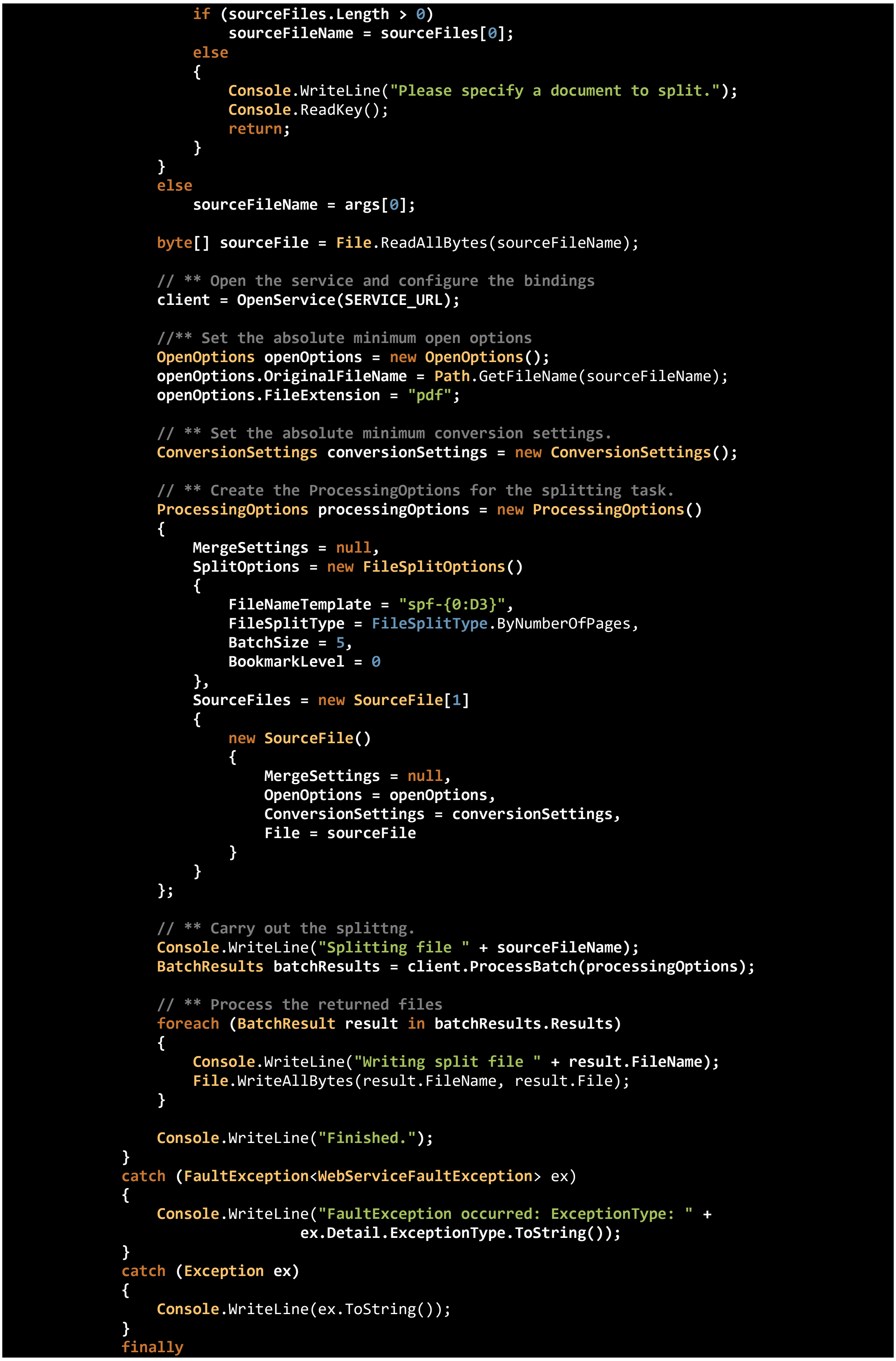
The following sample describes the steps needed to split up a single PDF file based on the number of pages. We are using Visual Studio and C#, but any environment that can invoke web services should be able to access this functionality. Note that the WSDL can be found at *http://localhost:41734 /Muhimbi.DocumentConverter.WebService/?wsdl*.

The source code for this example can be found in the folder <install location>\Muhimbi Document Converter\Sample Code. For more details see How to Split PDF Pages & Files using C# (muhimbi.com).

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 splittng.
                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.MaxStringContentLength = 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 https://msdn.microsoft.com/en-us/library/ms735119(v=VS.90).aspx.

4.10 Converting HTML / web pages using a Web Service call

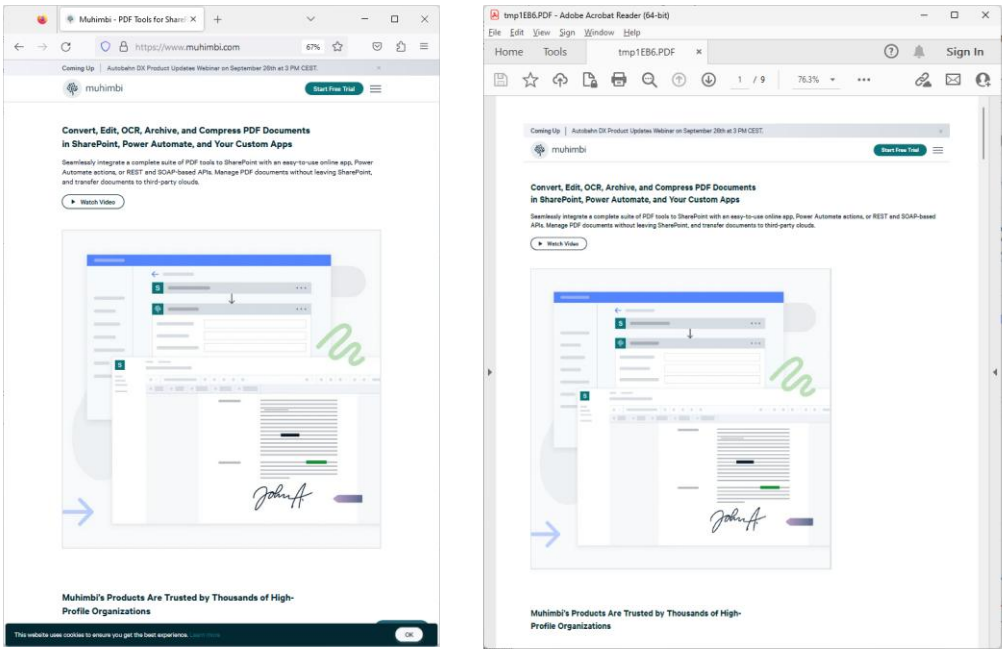
The Muhimbi PDF Converter comes with 3 different HTML to PDF Conversion engines. Legacy ones, based on Internet Explorer and Webkit, and a separate high-fidelity converter based on the Chromium framework. The Chromium converter is enabled by default, switching back to the legacy converters is possible, but discouraged.

Please keep in mind that HTML is not particularly well suited for printing or PDF output, however our software generally generates good results, especially with guidance provided in the following Knowledge Base articles:

- Converting HTML - Empty page / Authentication problems.
- Solving formatting issues when converting HTML to PDF.

The conversions service's config file provides a high level of control over the HTML Converter. Please consult section 2.6.10 of the Administration Guide as well as the Conversion Service config file's in-line documentation for more details.

Behaviour can be controlled on a request-by-request basis by passing in an instance of ConverterSpecificSettings_HTML in the ConversionSettings.ConverterSpecificSettings property. For details see section 3.2.6.


Muhimbi - PDF Tools for ShareP
https://www.muhimbi.com
67%
tmp1EB6.PDF - Adobe Acrobat Reader (64-bit)
File Edit View Sign Window Help
Home Tools tmp1EB6.PDF
Sign In
1 / 9
76.3%
Coming Up
Autobahn DX Product Updates Webinar on September 26th at 3 PM CEST.
muhimbi
Start Free Trial
Watch Video
OK

Convert, Edit, OCR, Archive, and Compress PDF Documents in SharePoint, Power Automate, and Your Custom Apps

Seamlessly integrate a complete suite of PDF tools to SharePoint with an easy-to-use online app, Power Automate actions, or REST and SOAP-based APIs. Manage PDF documents without leaving SharePoint, and transfer documents to third-party clouds.

Muhimbi's Products Are Trusted by Thousands of High-Profile Organizations

This website uses cookies to ensure you get the best experience. Learn more

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

4.10.1 Converting HTML in .NET

Listed below is a simple C# example showing 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("https://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 = "https://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);
    }
}
```

4.10.2 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, PDF/A2b or PDF/A3b

Using the PDF Converter Professional add-on license, 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*.

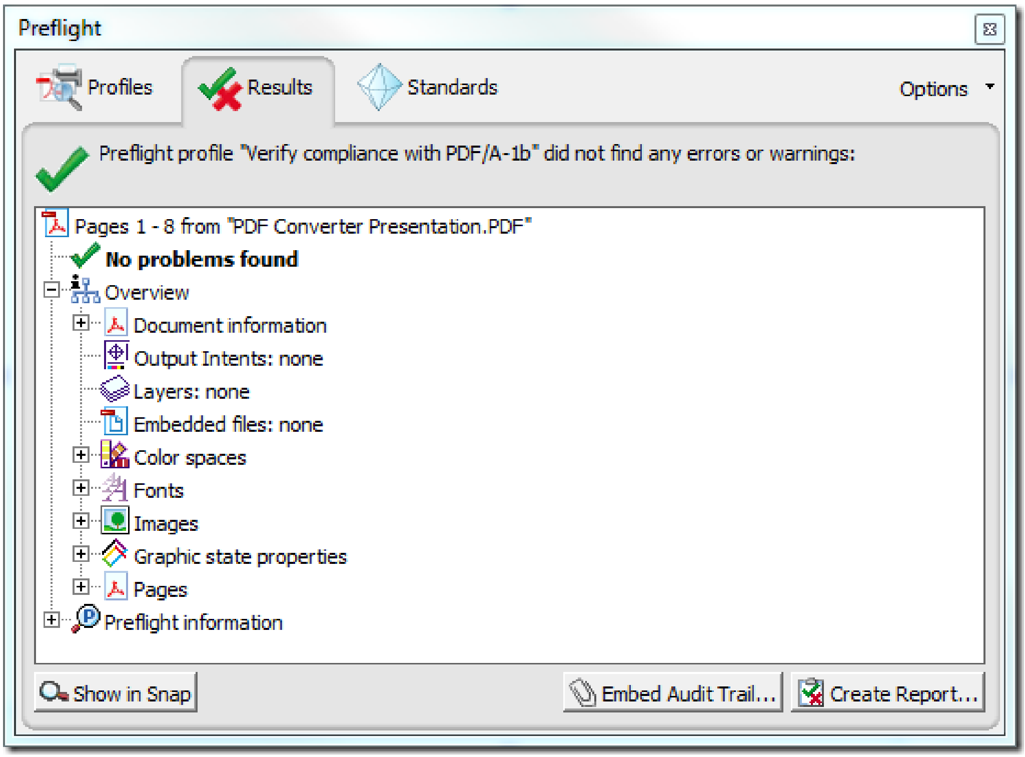
The on-line equivalent of this section can be found in the following blog post [Converting PDF document to PDF/A1b using the Muhimbi PDF Converter Web Service]().

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.



Preflight profile "Verify compliance with PDF/A-1b" did not find any errors or warnings:

Pages 1 - 8 from "PDF Converter Presentation.PDF"
No problems found
Overview
Document information
Output Intents: none
Layers: none
Embedded files: none
Color spaces
Fonts
Images
Graphic state properties
Pages
Preflight information

Sample Code

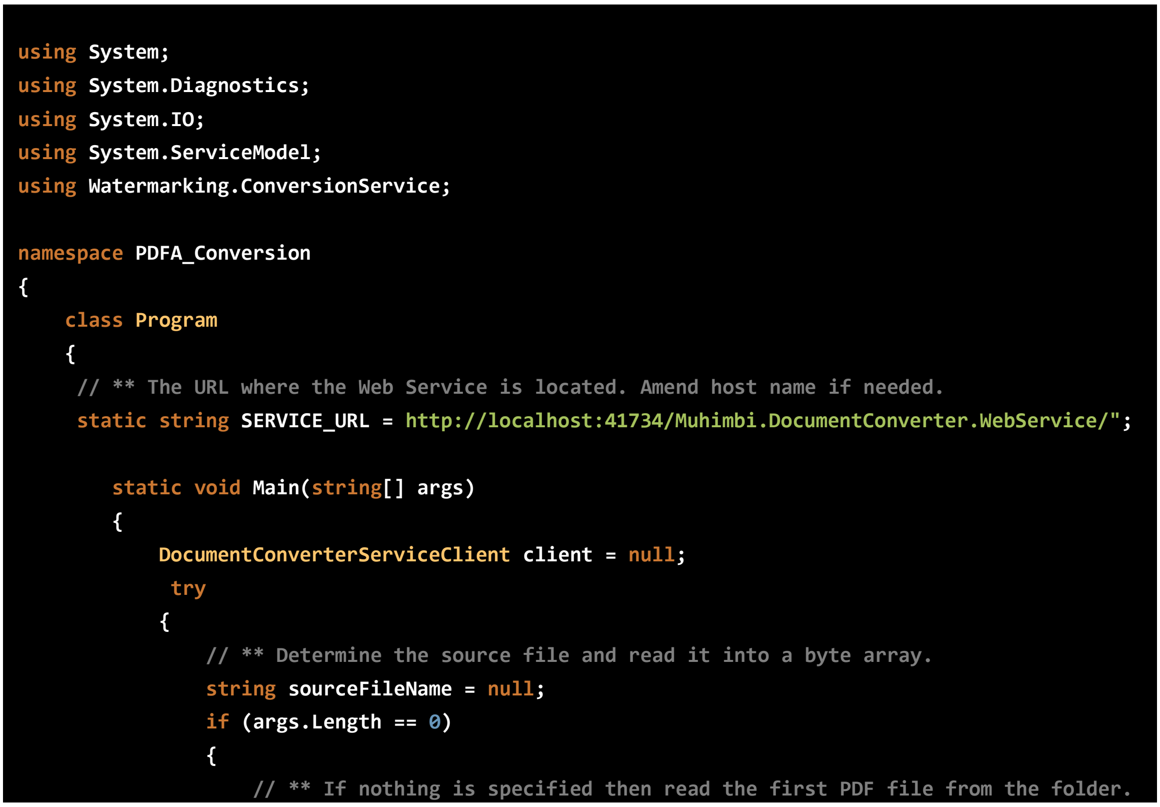
The sample code listed below converts PDF files to PDF/A files. You can either copy the code or open the VS project from the *Sample Code* folder in the *Start Menu* or visit our GitHub area.

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.

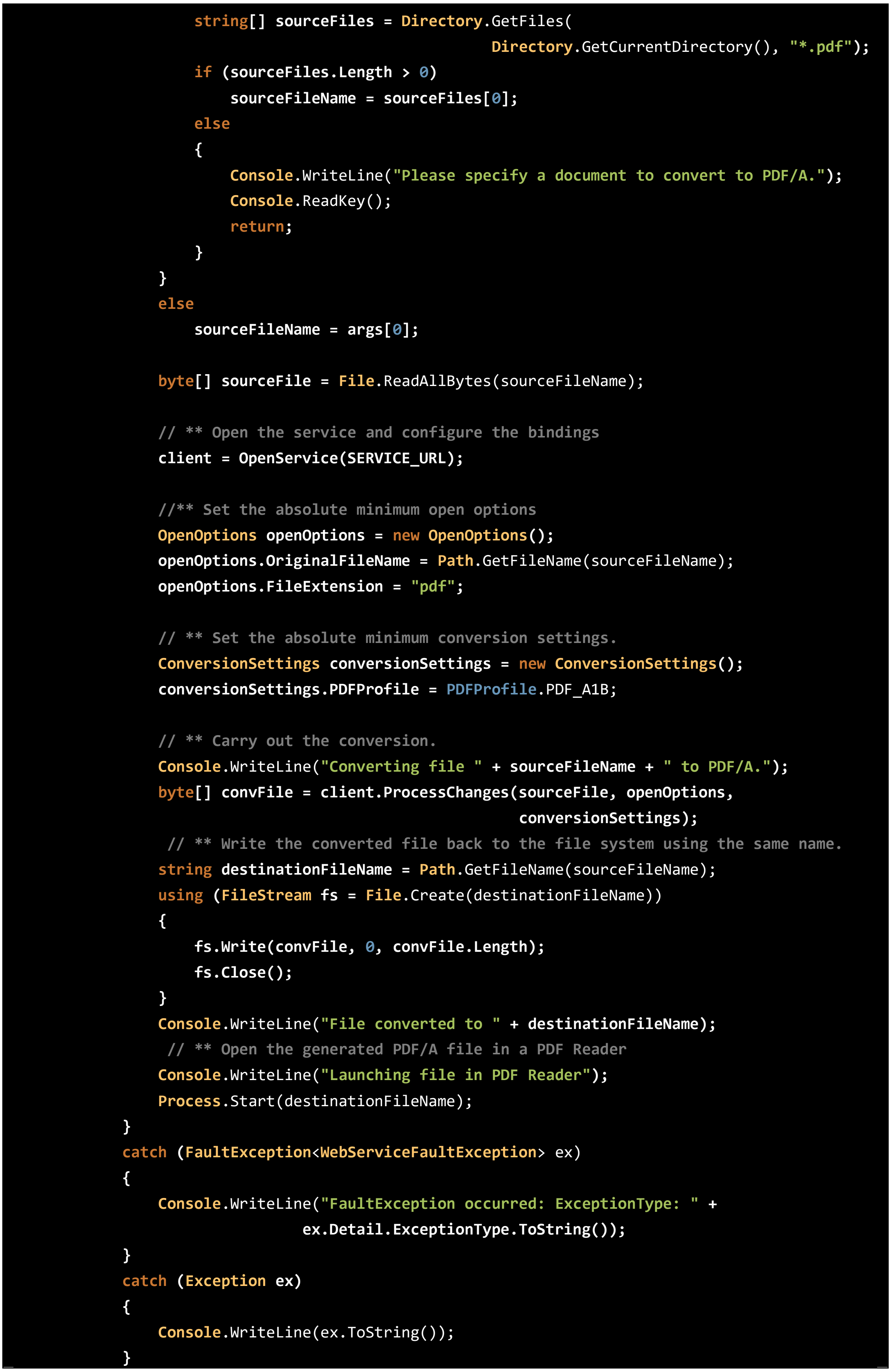
Please note that you need the PDF Converter Professional add-on license in addition to a valid PDF *Converter for SharePoint* or *PDF Converter Services* License in order to use this functionality.

1. Download and install 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

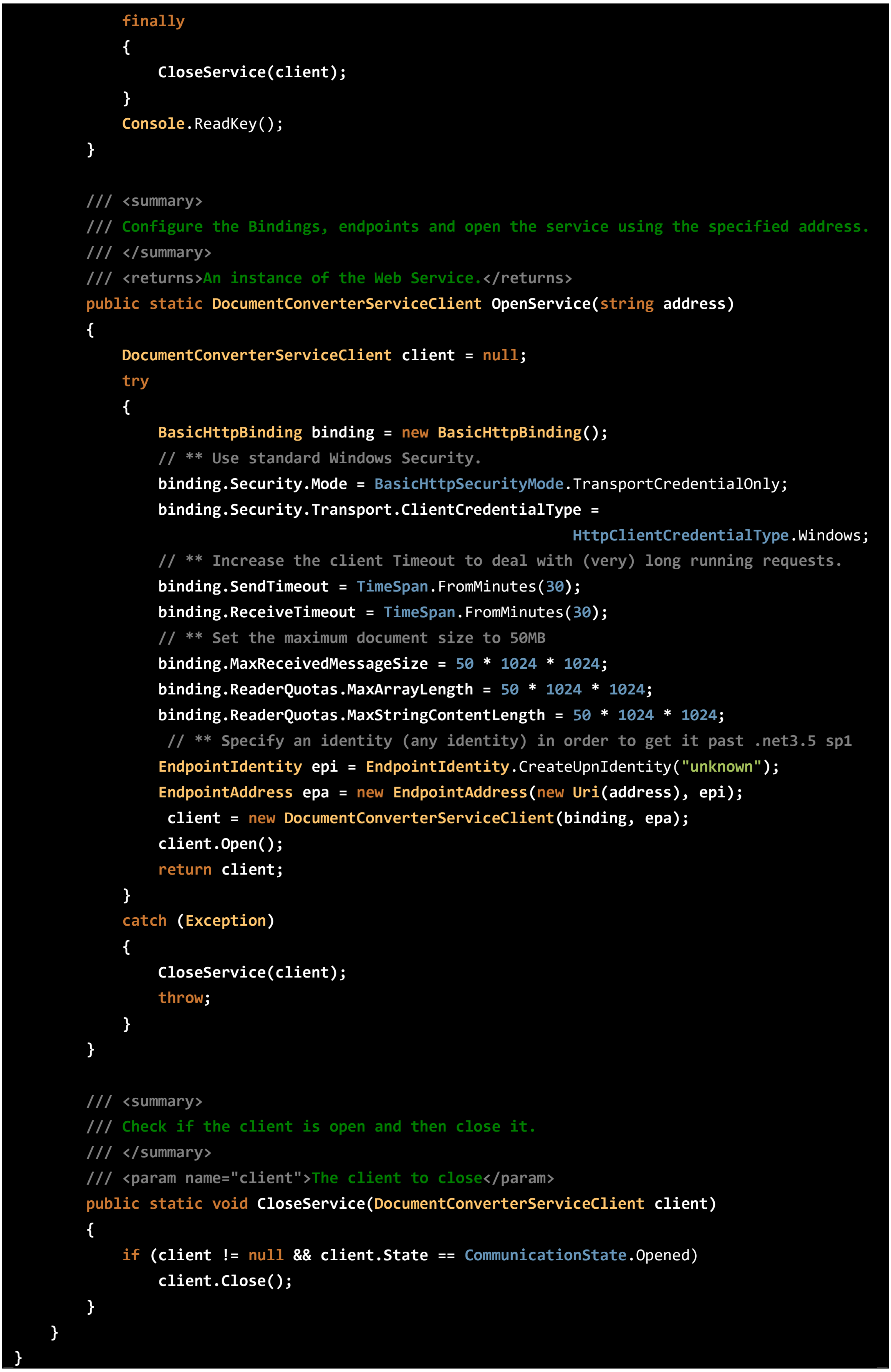
   https://localhost:41734/Muhimbi.DocumentConverter.WebService/?wsdl
5. Paste the following code into *Program.cs*.
6. Make sure the source folder contains a PDF file.
7. Compile and execute the application. The converted PDF/A file will automatically be opened in your system's PDF reader.



```
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.MaxStringContentLength = 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.

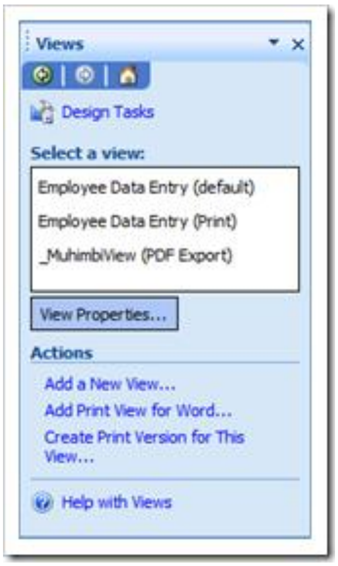
As always, the best way to illustrate this is by example. The latest version of this tutorial is available on the Muhimbi Blog at the following URL: Controlling which views to export to PDF format in InfoPath (muhimbi.com).

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.



Employee Data Entry (default)
Employee Data Entry (Print)
_MuhimbiView (PDF Export)
View Properties...
Actions
Add a New View...
Add Print View for Word...
Create Print Version for This View...
Help with Views

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 than 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.

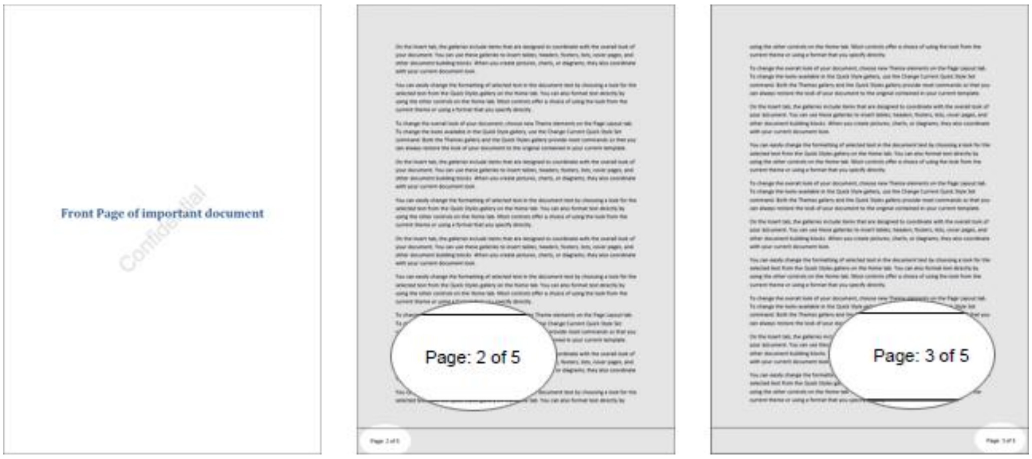
5 Working with watermarks

As described in chapter 3.5 Watermarking, 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. This works in PDF, DOCX, XLSX and PPTX files.

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.



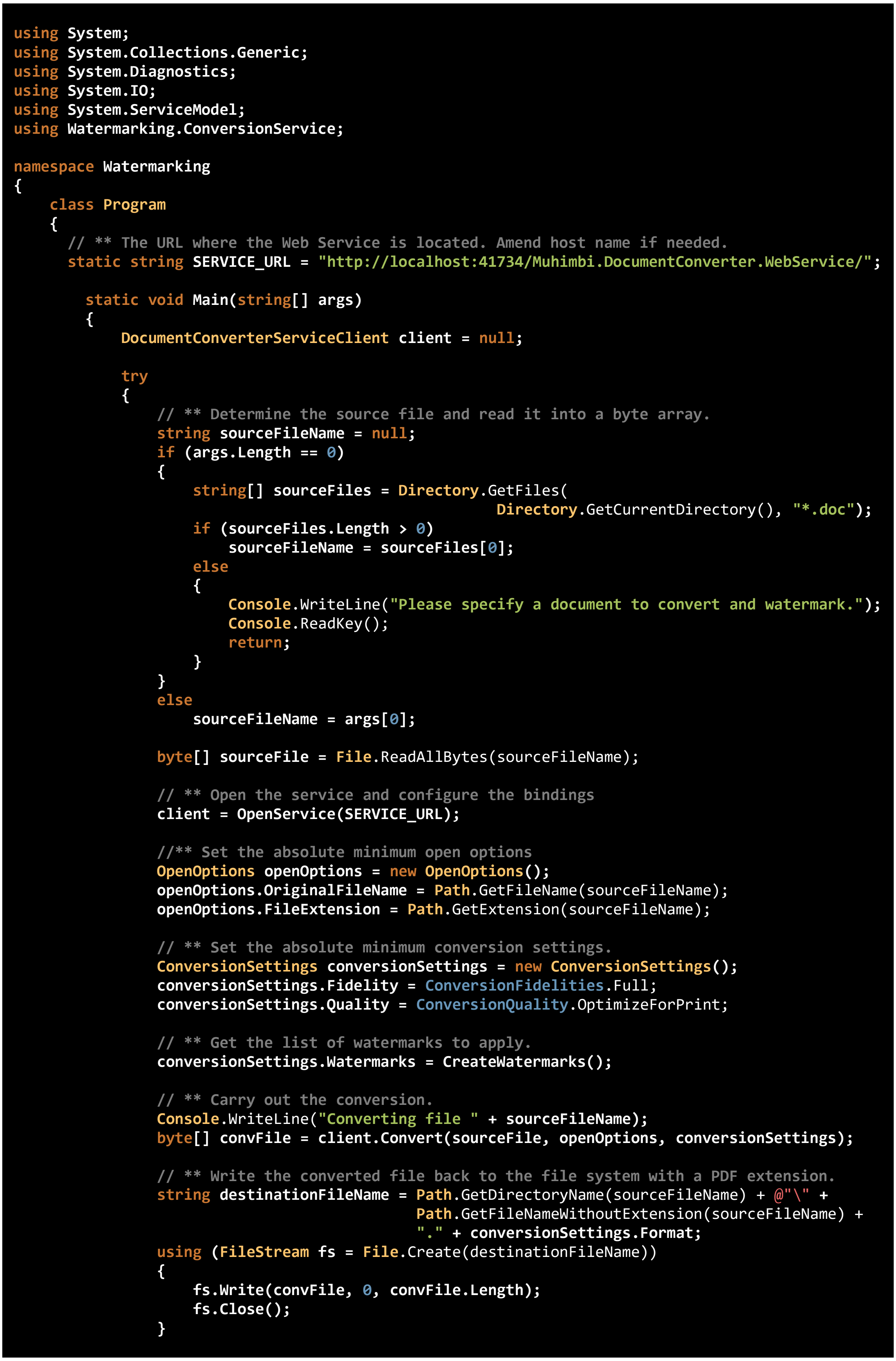
The sample code expects the path of the PDF file on the command line. If the path is omitted then the first MS-Word file found in the current directory will be used.

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

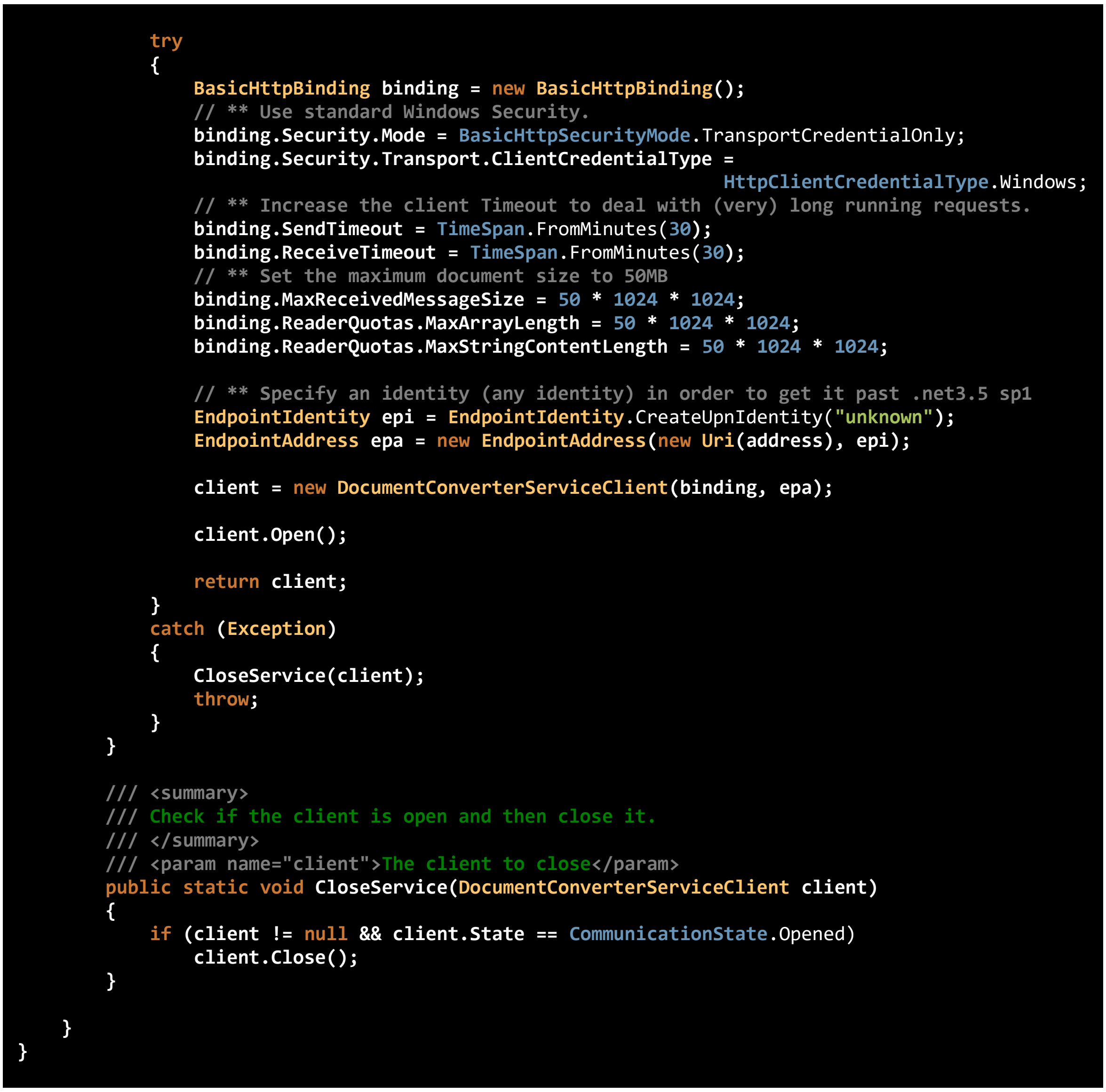
1. Create a new Visual Studio C# Console application named *Watermarking*.
2. Add a *Service Reference* to the following URL and specify *ConversionService* as the namespace

   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.GetDirectoryName(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.MaxStringContentLength = 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 source 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);
    }
  }
}
```

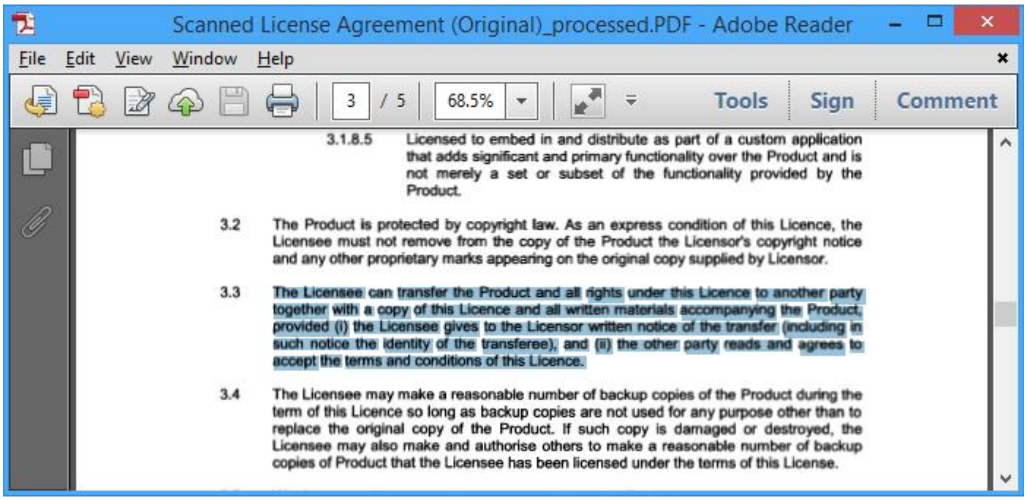
6 Carry out OCR (Optical Character Recognition)

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 License Agreement (Original)_processed.PDF - Adobe Reader
File
Edit
View
Window
Help
3
/ 5
68.5%
Tools
Sign
Comment

3.1.8.5 Licensed to embed in and distribute as part of a custom application that adds significant and primary functionality over the Product and is not merely a set or subset of the functionality provided by the Product.

3.2 The Product is protected by copyright law. As an express condition of this Licence, the Licensee must not remove from the copy of the Product the Licensor's copyright notice and any other proprietary marks appearing on the original copy supplied by Licensor.

3.3 The Licensee can transfer the Product and all rights under this Licence to another party together with a copy of this Licence and all written materials accompanying the Product, provided (i) the Licensee gives to the Licensor written notice of the transfer (including in such notice the identity of the transferee), and (ii) the other party reads and agrees to accept the terms and conditions of this Licence.

3.4 The Licensee may make a reasonable number of backup copies of the Product during the term of this Licence so long as backup copies are not used for any purpose other than to replace the original copy of the Product. If such copy is damaged or destroyed, the Licensee may also make and authorise others to make a reasonable number of backup copies of Product that the Licensee has been licensed under the terms of this License.

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.
- Integrate with 3rd party OCR Engines such as PrimeOCR.

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* or on GitHub.

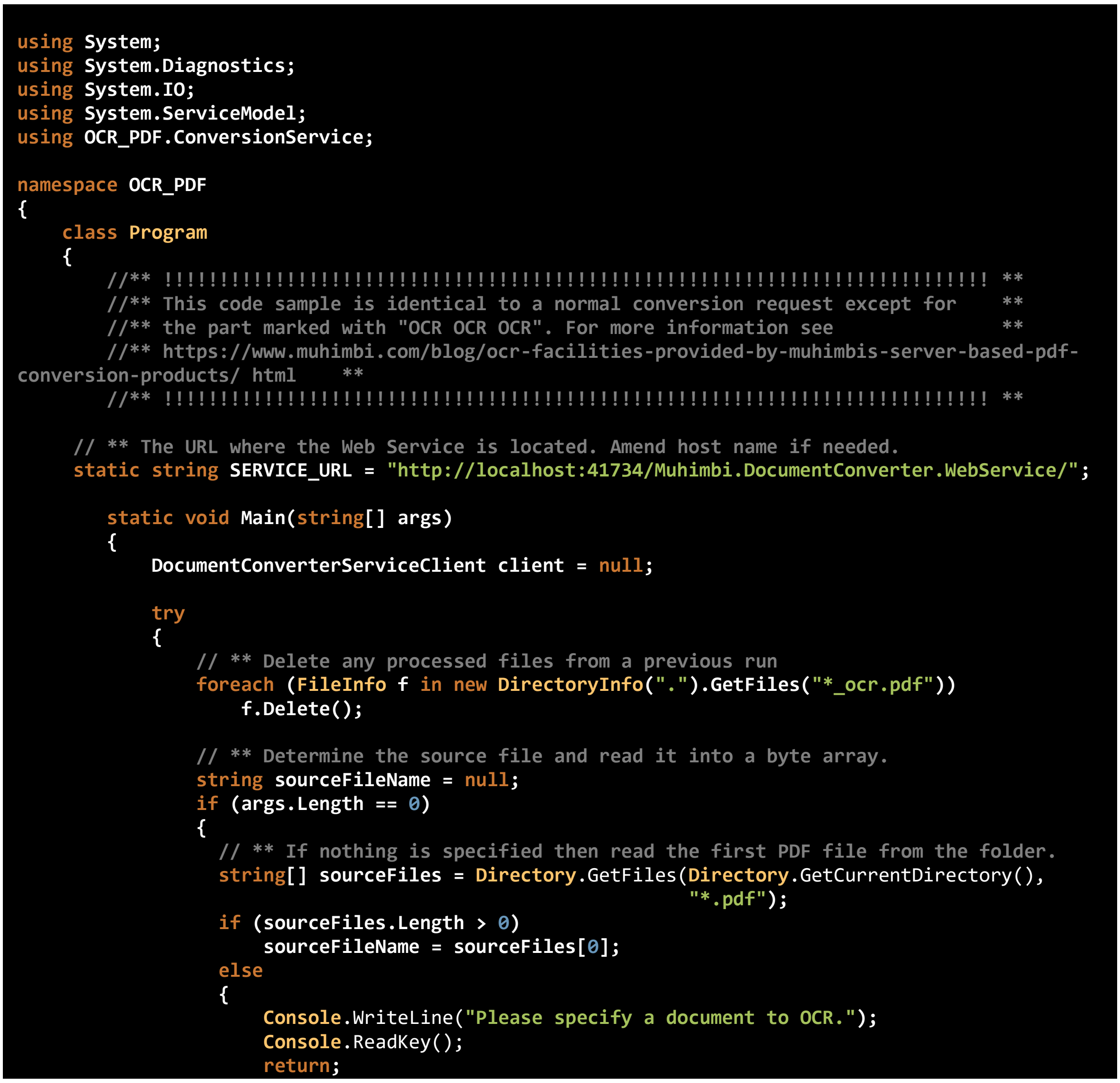
This example expects the path of the source PDF file on the command line. If the path is omitted, then the first PDF file found in the current directory will be used.

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

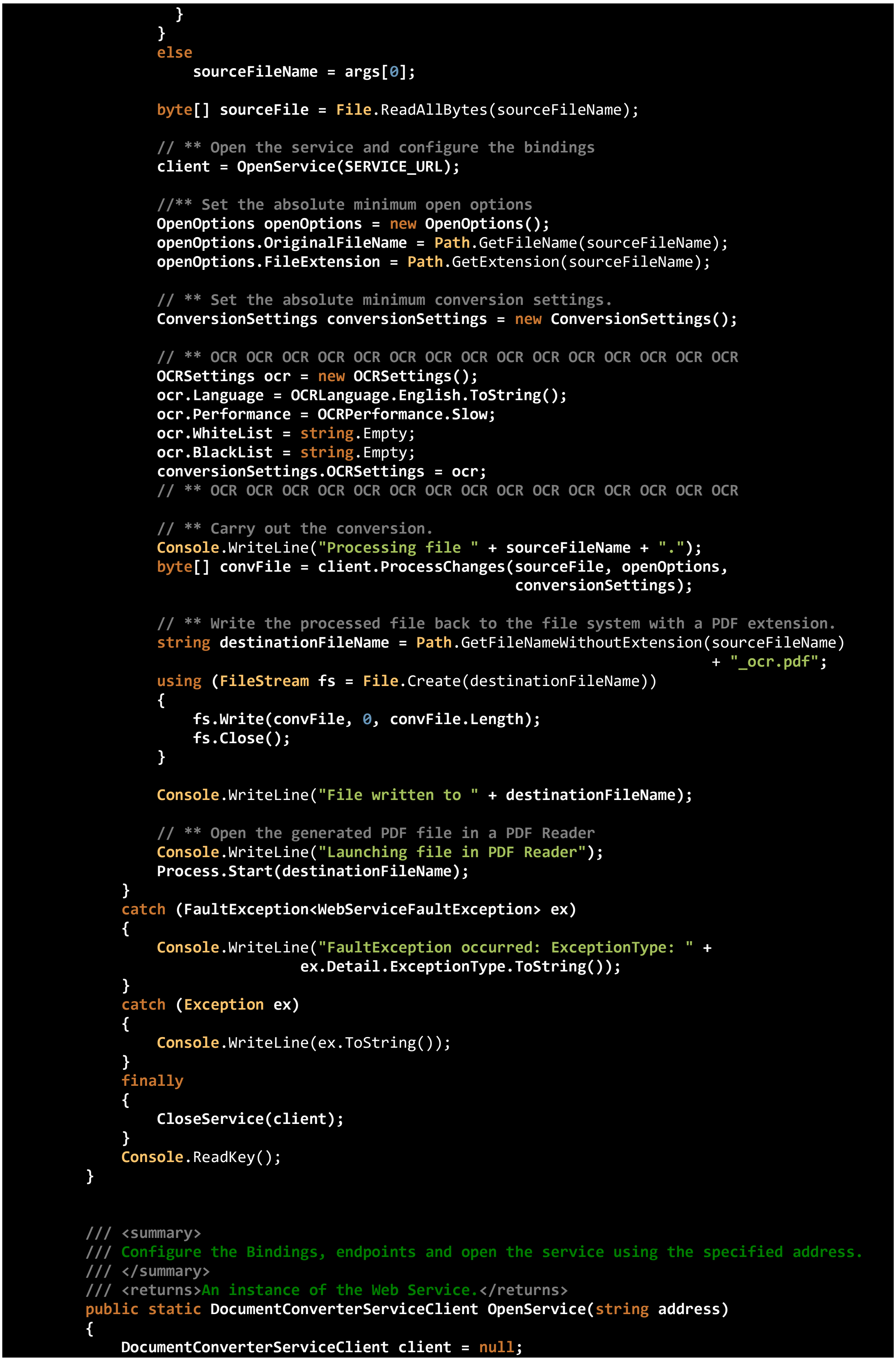
1. Install version 7.1 (or newer) of the *Muhimbi PDF Converter Services* or *PDF Converter for SharePoint*.
2. Create a new Visual Studio C# Console application named *OCR_PDF*.
3. Add a Service Reference to the following URL and specify *ConversionService* as the namespace. If you are developing on 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

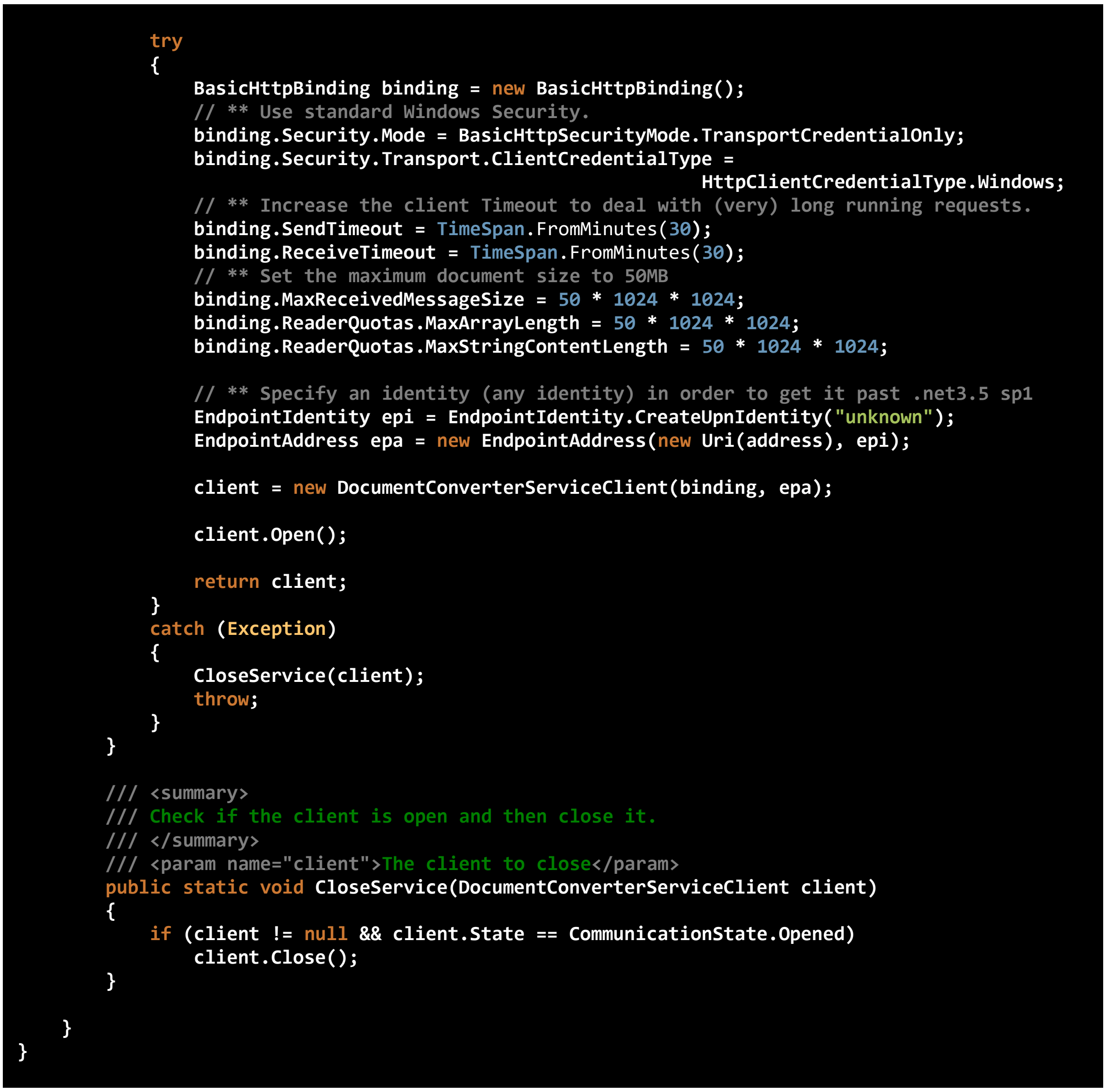
4. Paste the following code into *Program.cs*.



```
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               **
        //** https://www.muhimbi.com/blog/ocr-facilities-provided-by-muhimbis-server-based-pdf-
conversion-products/ 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
                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

                // ** 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.MaxStringContentLength = 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 source 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 OCRed 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 or visiting our GitHub area.

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.

```java
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);
          // ** 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);
    }
  }

}
```

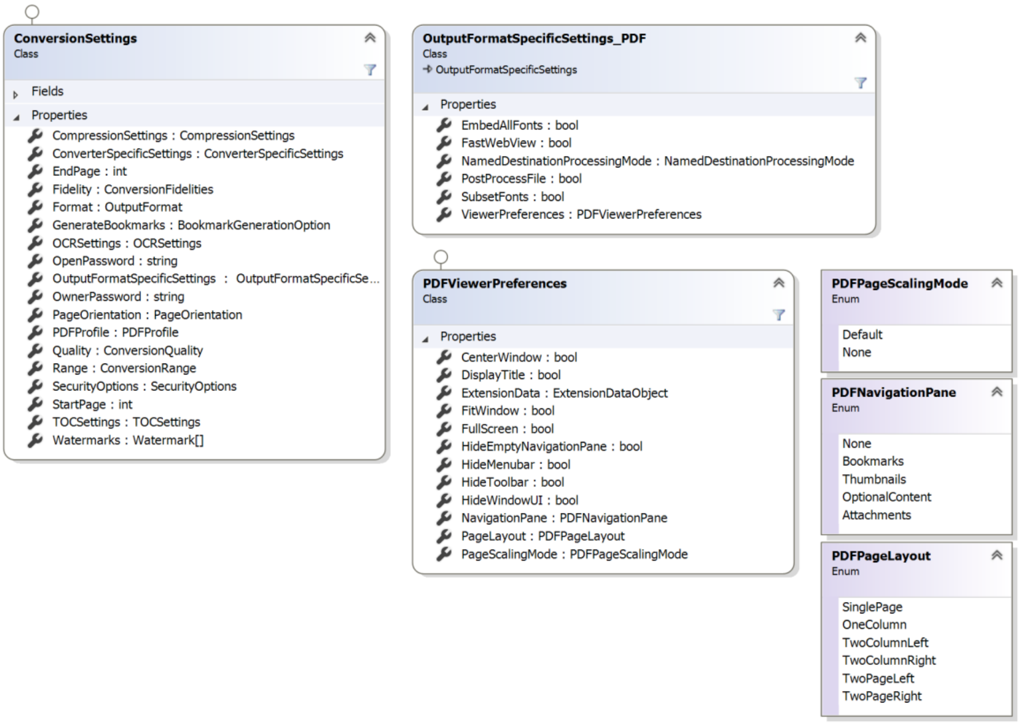
As all this functionality is exposed via a Web Services interface, it works equally well from .NET, PHP, Ruby and other web services enabled environments.

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:


ConversionSettings
Class
Fields
Properties
CompressionSettings : CompressionSettings
ConverterSpecificSettings : ConverterSpecificSettings
EndPage : int
Fidelity : ConversionFidelities
Format : OutputFormat
GenerateBookmarks : BookmarkGenerationOption
OCRSettings : OCRSettings
OpenPassword : string
OutputFormatSpecificSettings : OutputFormatSpecificSe...
OwnerPassword : string
PageOrientation : PageOrientation
PDFProfile : PDFProfile
Quality : ConversionQuality
Range : ConversionRange
SecurityOptions : SecurityOptions
StartPage : int
TOCSettings : TOCSettings
Watermarks : Watermark[]
OutputFormatSpecificSettings_PDF
Class
OutputFormatSpecificSettings
Properties
EmbedAllFonts : bool
FastWebView : bool
NamedDestinationProcessingMode : NamedDestinationProcessingMode
PostProcessFile : bool
SubsetFonts : bool
ViewerPreferences : PDFViewerPreferences
PDFViewerPreferences
Class
Properties
CenterWindow : bool
DisplayTitle : bool
ExtensionData : ExtensionDataObject
FitWindow : bool
FullScreen : bool
HideEmptyNavigationPane : bool
HideMenubar : bool
HideToolbar : bool
HideWindowUI : bool
NavigationPane : PDFNavigationPane
PageLayout : PDFPageLayout
PageScalingMode : PDFPageScalingMode
PDFPageScalingMode
Enum
Default
None
PDFNavigationPane
Enum
None
Bookmarks
Thumbnails
OptionalContent
Attachments
PDFPageLayout
Enum
SinglePage
OneColumn
TwoColumnLeft
TwoColumnRight
TwoPageLeft
TwoPageRight

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 hides 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.Output FormatSpecificSettings* (or *MergeSettings.OutputFormatSpecificSettings*) to an instance of *OutputFormatSpecificSettings_PDF* and populating the *Viewer Preferences* 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 *OutputFormat SpecificSettings_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.OutputFormatSpecific Settings*) 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. Ghostscript 9.07 or later will need to be installed to make use of *FastWebView*.

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_A3B:** Use the PDF/A3b 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.

7.3 Generating Named Destinations

A particular useful feature when opening PDF files from a web page, is the ability to use *named destinations* to automatically navigate to a specific section in a PDF file, e.g. *http://somedomain/AdminGuide.pdf#2Deployment*. These named destinations are similar to anchor tags inside an HTML document.

Historically it has always been a challenge to (often manually) maintain all named destinations in a document, especially when dealing with large and complex files. Fortunately, the Muhimbi PDF Converter can automatically generate this information based on a PDF's bookmarks, which in turn are automatically generated from – for example – an MS-Word file's headings.

The generation of these named destinations can be controlled globally, via the Conversion Service's config file in the *PDF.NamedDestinationProcessingMode* setting, as well as on a request-by-request basis in ConversionSettings.Output FormatSpecificSettings. The possible values are as follows:

- **None** (default): Make no change to the named destinations defined in the document.

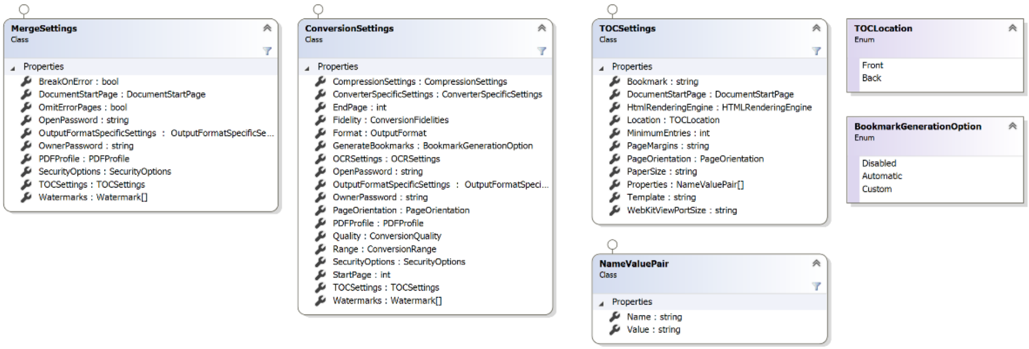
- **ClearAll**: Remove all named destinations. (All bookmarks pointing to existing named destinations will be fixed up automatically)
- **Merge**: Keep existing named destinations and add new ones based on the PDF's bookmarks.
- **Replace**: Remove all existing named destinations and add new ones based on the PDF's bookmarks.

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.

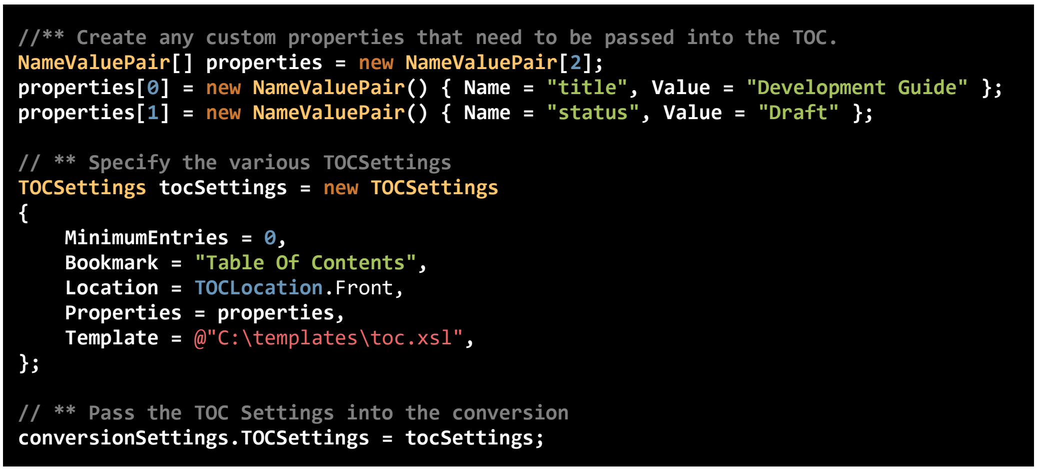
8.1 Object Model

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


MergeSettings
Class
Properties
BreakOnError : bool
DocumentStartPage : DocumentStartPage
OmitErrorPages : bool
OpenPassword : string
OutputFormatSpecificSettings : OutputFormatSpecificSe...
OwnerPassword : string
PDFProfile : PDFProfile
SecurityOptions : SecurityOptions
TOCSettings : TOCSettings
Watermarks : Watermark[]
ConversionSettings
Class
Properties
CompressionSettings : CompressionSettings
ConverterSpecificSettings : ConverterSpecificSettings
EndPage : int
Fidelity : ConversionFidelities
Format : OutputFormat
GenerateBookmarks : BookmarkGenerationOption
OCRSettings : OCRSettings
OpenPassword : string
OutputFormatSpecificSettings : OutputFormatSpeci...
OwnerPassword : string
PageOrientation : PageOrientation
PDFProfile : PDFProfile
Quality : ConversionQuality
Range : ConversionRange
SecurityOptions : SecurityOptions
StartPage : int
TOCSettings : TOCSettings
Watermarks : Watermark[]
TOCSettings
Class
Properties
Bookmark : string
DocumentStartPage : DocumentStartPage
HtmlRenderingEngine : HTMLRenderingEngine
Location : TOCLocation
MinimumEntries : int
PageMargins : string
PageOrientation : PageOrientation
PaperSize : string
Properties : NameValuePair[]
Template : string
WebKitViewPortSize : string
NameValuePair
Class
Properties
Name : string
Value : string
TOCLocation
Enum
Front
Back
BookmarkGenerationOption
Enum
Disabled
Automatic
Custom

- **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 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. For details see the class definitions in 3.6.1.
- **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 (<install location>\Muhimbi Document Converter\Sample Code or from GitHub), 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, the following sample (download) is a good starting point and can be amended to suit your particular needs (or used as is).

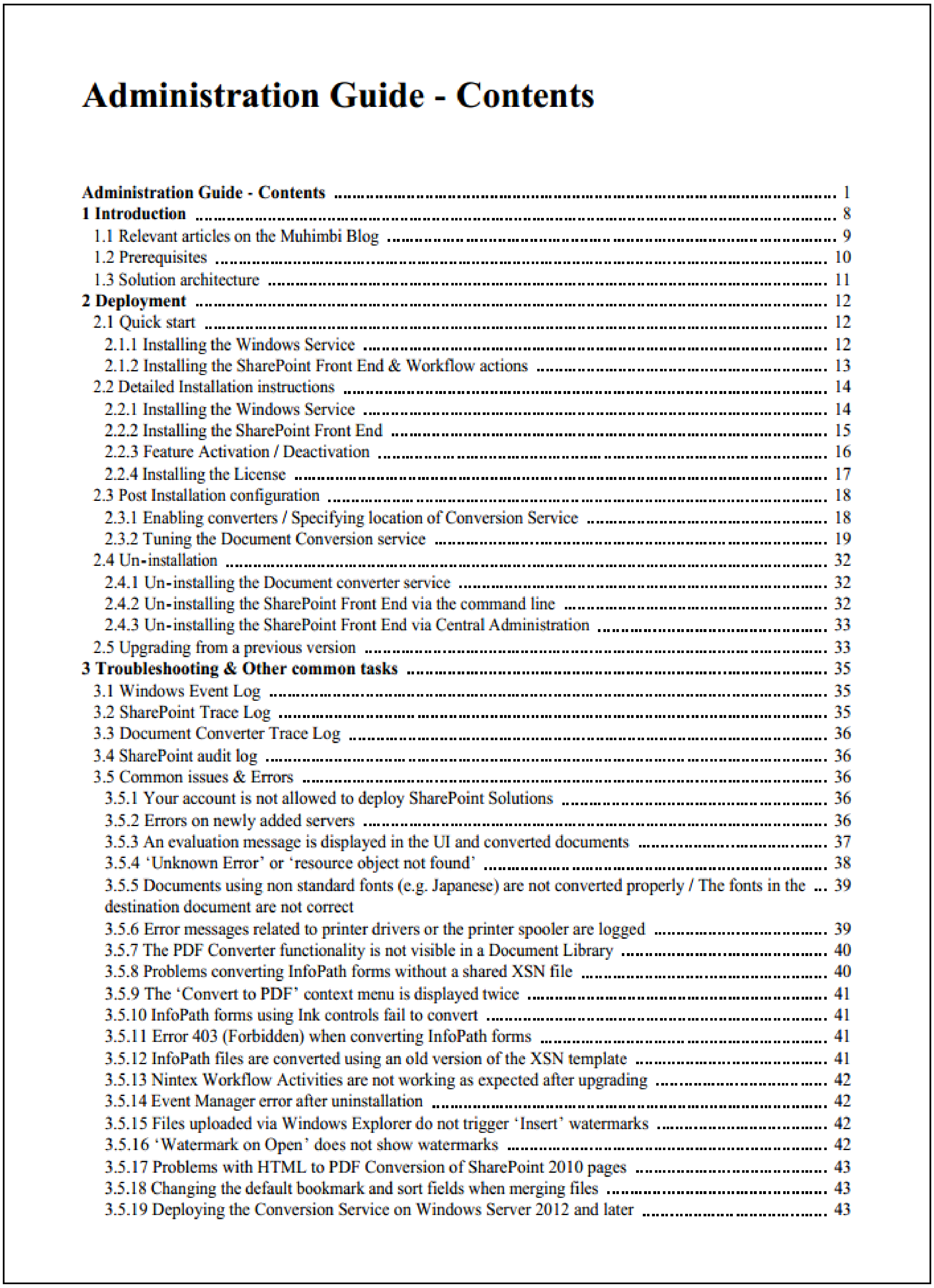
```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:msxsl="urn:schemas-microsoft-com:xslt"
                exclude-result-prefixes="msxsl">

  <xsl:output method="html" indent="yes"/>

  <xsl:template match="/toc">
    <html>
      <head>
        <style type="text/css">
          ul.toc
          {
              margin: 0;
              padding: 0;
              list-style: none;
          }
          ol.toc
          {
              margin: 0;
              padding: 0;
              margin-left: 10px;
              list-style: none;
          }
          ul.toc li
          {
              clear: both;
              overflow: hidden;
          }
          ol.toc li
          {
              overflow: hidden;
          }
          span.title
          {
              float: left;
              padding-right: 4px;
          }
          span.page
          {
              float: right;
              padding-left: 4px;
          }
          span.dots
          {
              font-size: 0px;
              width:100%;
              border-bottom: 2px dotted black;
          }
          a.toc
          {
            text-decoration: none;
            color: #000;
          }
        </style>
      </head>
      <body>
        <h1>
          <xsl:value-of select="properties/property[@name='title']"/>
        </h1>
        <br/>
        <br/>
        <xsl:apply-templates/>
      </body>
    </html>
  </xsl:template>

  <xsl:template match="topics">
    <ul class="toc">
      <xsl:apply-templates/>
    </ul>
  </xsl:template>

  <!-- Empty template so properties are not appearing -->
  <xsl:template match="properties"></xsl:template>

  <xsl:template match="topic[@level='0']">
    <li>
      <xsl:element name="a">
        <xsl:attribute name="href">
          <xsl:value-of select="@target"/>
        </xsl:attribute>
        <xsl:attribute name="class">toc</xsl:attribute>
        <span class="title" style="font-weight: 900;">
          <xsl:value-of select="@title"/>
        </span>
        <span class="page">
          <xsl:value-of select="@page"/>
        </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.

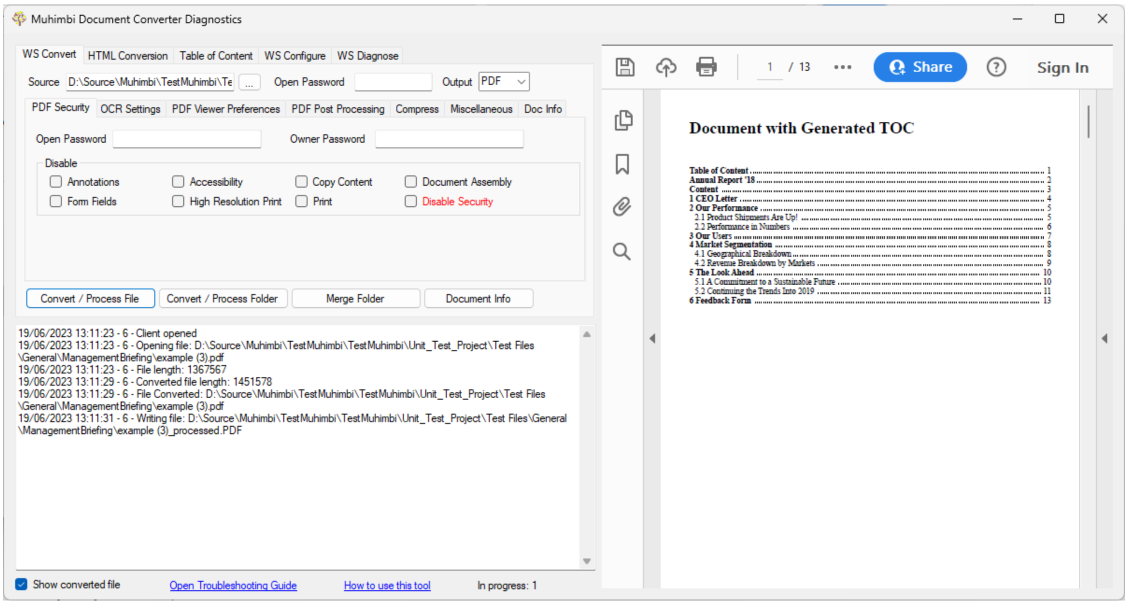
If your experience with XML and XSL is limited then we recommend using the XSL sample provided above. As can be seen below, the results look very good.



Administration Guide - Contents

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.



Source D:\Source\Muhimbi\TestMuhimbi\Te ... Open Password Output PDF

Open Password Owner Password

Disable

□ Annotations □ Accessibility □ Copy Content □ Document Assembly

□ Form Fields □ High Resolution Print □ Print □ Disable Security

```
19/06/2023 13:11:23 - 6 - Client opened
19/06/2023 13:11:23 - 6 - Opening file: D:\Source\Muhimbi\TestMuhimbi\TestMuhimbi\Unit_Test_Project\Test Files
\General\ManagementBriefing\example (3).pdf
19/06/2023 13:11:23 - 6 - File length: 1367567
19/06/2023 13:11:29 - 6 - Converted file length: 1451578
19/06/2023 13:11:29 - 6 - File Converted: D:\Source\Muhimbi\TestMuhimbi\TestMuhimbi\Unit_Test_Project\Test Files
\General\ManagementBriefing\example (3).pdf
19/06/2023 13:11:31 - 6 - Writing file: D:\Source\Muhimbi\TestMuhimbi\TestMuhimbi\Unit_Test_Project\Test Files\General
\ManagementBriefing\example (3)_processed.PDF
```

☑ Show converted file In progress: 1

1 / 13

Document with Generated TOC

Table of Content 1
Annual Report '18 2
Content 3
1 CEO Letter 4
2 Our Performance 5
2.1 Product Shipments Are Up! 5
2.2 Performance in Numbers 6
3 Our Users 7
4 Market Segmentation 8
4.1 Geographical Breakdown 8
4.2 Revenue Breakdown by Markets 9
5 The Look Ahead 10
5.1 A Commitment to a Sustainable Future 10
5.2 Continuing the Trends Into 2019 11
6 Feedback Form 13

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 Compressing output files

As of version 10.3, PDF Converter can make use of the Hyper Compression option. This can reduce the size of the output files by removing unwanted items (annotations, blank pages, JavaScript), downscaling images and using improved image compression techniques.

Hyper-compression, also known as Mixed Raster Content (MRC), is an image compression benefiting from image segmentation methods. It is particularly useful for images containing text and continuous-tone graphics.

9.1 Specifying the compression options

The following options are available:

- **RemoveAnnotations** Remove annotations
- **RemoveBlankPages** Remove blank pages
- **RemoveBookmarks** Remove bookmarks
- **RemoveEmbeddedFiles** Remove embedded files
- **RemoveFormFields** Remove form fields
- **RemoveHyperlinks** Remove hyperlinks
- **RemoveJavaScript** Remove JavaScript
- **RemoveMetadata** Remove XMP metadata (normal PDF – Title, Author etc and custom metadata is untouched)
- **RemovePageThumbnails** Remove page thumbnails
- **PackFonts** Pack the PDF's fonts to reduce their size.
- **PackDocument** Pack the PDF to reduce its size.
- **RecompressImages** Recompress the PDF's images
- **EnableMRC** MRC (Hyper Compression) engine will be used for compressing the PDF contents
- **DownscaleResolutionMRC** Resolution (DPI) for downscaling the background layer by the MRC engine. Default value is 100
- **PreserveSmoothing** MRC engine will preserve smoothing between different layers
- **ImageQuality** Image quality to be used for the compression of the images from the PDF
- **DownscaleImages** Images from the PDF will be downscaled
- **DownscaleResolution** Resolution used to downscale images. Default value is 150
- **EnableColorDetection** Color detection will be performed on the images from the PDF
- **EnableCharRepair** Character repairing will be performed during bitonal conversion
- **EnableJPEG2000** Use JPEG2000 compression scheme to compress color images
- **EnableJBIG2** Use JBIG2 compression scheme to compress bitonal images
- **JBIG2PMSThreshold** Threshold value for the JBIG2 encoder pattern matching and substitution. Range 0 to 100, any number lower than 100 may lead to lossy compression. Default value is 85

9.2 Compression using .NET

This program expects the file to process as the first argument and the folder to take the compressed file as the second argument.

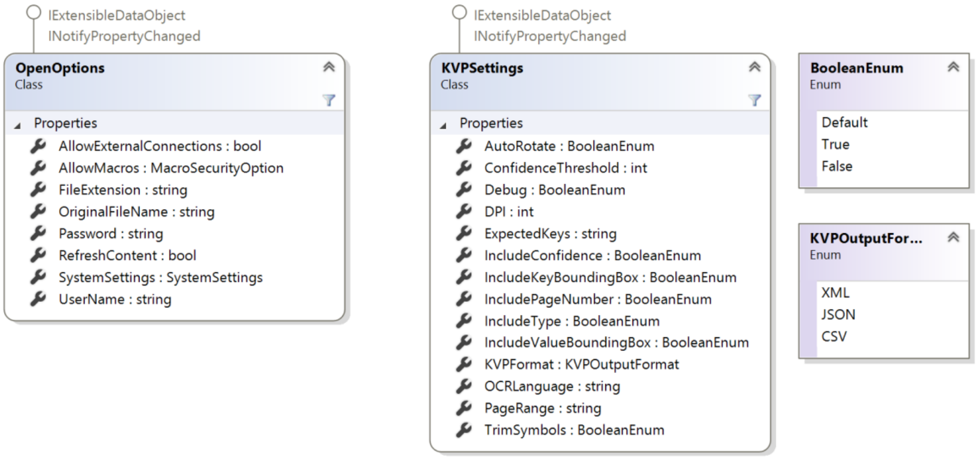
```
using Compress_PDF.ServiceReference1;
using System;
using System.IO;
using System.ServiceModel;

namespace Compress_PDF
{
    class Program
    {
        private static string ServiceURL =
"http://localhost:41734/Muhimbi.DocumentConverter.WebService/";

        static void Main(string[] args)
        {
            string sourceFilename = args[0];        // Source file is first
argument
            string targetFolder = args[1];          // Target folder is
second argument

            Console.WriteLine($"Source file: {sourceFilename}");
            Console.WriteLine($"Target folder: {targetFolder}");

            ServiceReference1.ConversionSettings conversionSettings = new
ServiceReference1.ConversionSettings();
            ServiceReference1.OpenOptions openOptions = new
ServiceReference1.OpenOptions();
            ServiceReference1.CompressionSettings compressionSettings;

            //** Set the absolute minimum open options
            openOptions.OriginalFileName = Path.GetFileName(sourceFilename);
            openOptions.FileExtension = Path.GetExtension(sourceFilename);
            openOptions.AllowMacros =
ServiceReference1.MacroSecurityOption.None;

            //** Conversion settings
            conversionSettings.Fidelity =
ServiceReference1.ConversionFidelities.Full;
            conversionSettings.StartPage = 0;
            conversionSettings.EndPage = 0;
            conversionSettings.Quality =
ServiceReference1.ConversionQuality.OptimizeForPrint;
            conversionSettings.GenerateBookmarks =
ServiceReference1.BookmarkGenerationOption.Automatic;
            conversionSettings.Range =
ServiceReference1.ConversionRange.AllDocuments;
            conversionSettings.Format = ServiceReference1.OutputFormat.PDF;
            conversionSettings.PDFProfile =
ServiceReference1.PDFProfile.Default;

            if (true)
            {
                //** Compression settings
                compressionSettings = new
ServiceReference1.CompressionSettings();

                // Remove options
                compressionSettings.RemoveAnnotations =
ServiceReference1.BooleanEnum.False;
                compressionSettings.RemoveBlankPages =
ServiceReference1.BooleanEnum.False;
                compressionSettings.RemoveBookmarks =
ServiceReference1.BooleanEnum.False;
                compressionSettings.RemoveFormFields =
ServiceReference1.BooleanEnum.False;
                compressionSettings.RemoveJavaScript =
ServiceReference1.BooleanEnum.False;
                compressionSettings.RemoveMetadata =
ServiceReference1.BooleanEnum.False;
```

```
                compressionSettings.PackFonts =
ServiceReference1.BooleanEnum.True;
                compressionSettings.PackDocument =
ServiceReference1.BooleanEnum.True;
                compressionSettings.EnableJPEG2000 =
ServiceReference1.BooleanEnum.True;
                compressionSettings.EnableJBIG2 =
ServiceReference1.BooleanEnum.True;
                if (compressionSettings.EnableJBIG2 ==
ServiceReference1.BooleanEnum.True)
                {
                    compressionSettings.JBIG2PMSThreshold = 85;
                }

                compressionSettings.RecompressImages =
ServiceReference1.BooleanEnum.True;
                compressionSettings.PreserveSmoothing =
ServiceReference1.BooleanEnum.True;
                compressionSettings.EnableColorDetection =
ServiceReference1.BooleanEnum.True;
                compressionSettings.EnableCharRepair =
ServiceReference1.BooleanEnum.True;

                compressionSettings.EnableMRC =
ServiceReference1.BooleanEnum.False;
                if (compressionSettings.EnableMRC ==
ServiceReference1.BooleanEnum.True)
                {
                    compressionSettings.DownscaleResolutionMRC = 200;
                }

                compressionSettings.DownscaleImages =
ServiceReference1.BooleanEnum.False;
                if (compressionSettings.DownscaleImages ==
ServiceReference1.BooleanEnum.True)
                {
                    compressionSettings.DownscaleResolution = 200;
                }

                compressionSettings.ImageQuality =
(ServiceReference1.ImageQuality.ImageQualityMedium);

                // Add compression settings to conversion settings
                conversionSettings.CompressionSettings =
compressionSettings;
            }
            // Read source file
            byte[] sourceFile = File.ReadAllBytes(sourceFilename);

            // Create converter object
            DocumentConverterServiceClient converter =
OpenService(ServiceURL);

            // Call conversion
            byte[] convFile = converter.Convert(sourceFile, openOptions,
conversionSettings);

            // Close conversion object
            CloseService(converter);
            // ** Write the processed file back to the file system with a
PDF extension.
            string destinationFileName =
Path.Combine(targetFolder,Path.GetFileNameWithoutExtension(sourceFilename)+
".pdf");
            Console.WriteLine($"Output file: {destinationFileName}");
            using (FileStream fs = File.Create(destinationFileName))
            {
                fs.Write(convFile, 0, convFile.Length);
                fs.Close();
            }
            Console.WriteLine($"Finished, press any key");
            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.MaxStringContentLength = 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();
        }
    }
}
```

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 to use this functionality.

10 Extracting Key-Value Pairs

10.1 Object Model



- **OpenOptions:** provides details about the file (original file name, extension), security (username and password, macros).
- KVPSettings**:** provides the settings for the extraction. For more details see *3.7*.

10.2 Expected keys

You can add an Expected Keys string to tell PDF Converter which keys to extract from the input file.

Within the string, you can specify synonyms for your keys, so that values paired with any synonym will also be extracted with the key.

This is very useful when processing files with varying formats and different ways of framing the same data.
There is an example 'Expected Key' string in the Appendices that shows how it can be used to cover multiple naming for the same key.

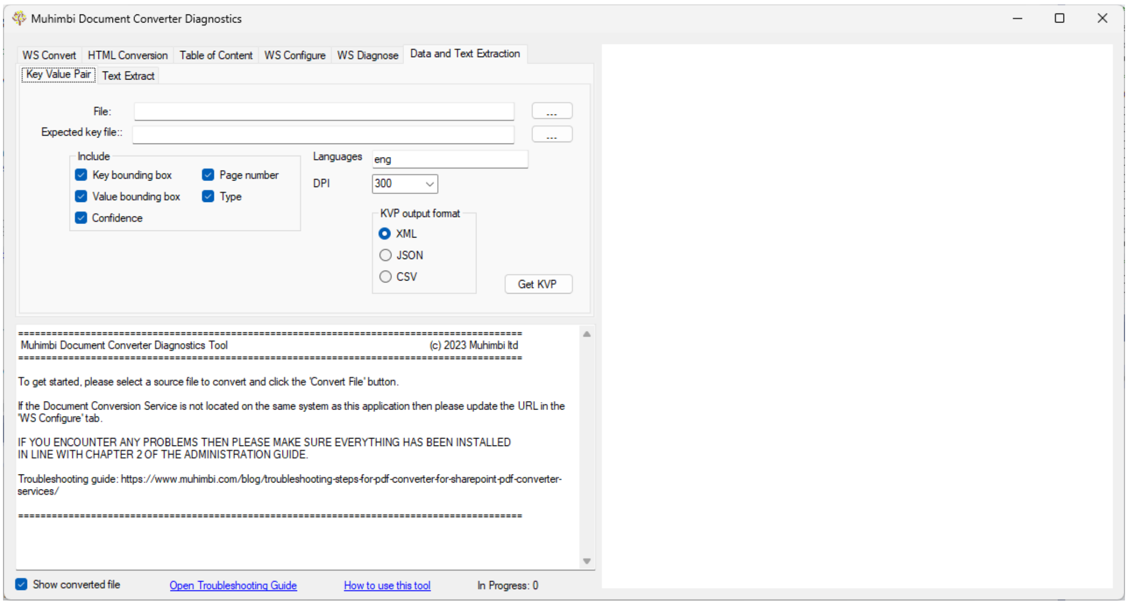
10.3 Extracting Key-Value Pairs in .NET

```
/// <summary>
/// Get Key Value pairs with expected keys
/// </summary>
/// <param name="sourceFileName">File to process</param>
/// <param name="targetFolder">Folder for the result of the processing</param>
static void KVPExtract(string sourceFileName, string targetFolder, string
expectedKeys = null)
{
    DocumentConverterServiceClient client = null;
    try
    {
        if (!Directory.Exists(targetFolder))
        {
            Directory.CreateDirectory(targetFolder);
        }
        // ** Determine the source file and read it into a byte array.
        byte[] sourceFile = File.ReadAllBytes(sourceFileName);

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

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

        KVPSettings kvpSettings = new KVPSettings();
        kvpSettings.DPI = 300;
        kvpSettings.KVPFormat = KVPOutputFormat.XML;
        kvpSettings.OCRLanguage = "eng";
        kvpSettings.IncludePageNumber = BooleanEnum.False;
        kvpSettings.IncludeType = BooleanEnum.False;
        kvpSettings.IncludeKeyBoundingBox = BooleanEnum.False;
        kvpSettings.IncludeValueBoundingBox = BooleanEnum.False;
        if (!string.IsNullOrEmpty(expectedKeys))
        {
            kvpSettings.ExpectedKeys = expectedKeys;
        }
        // ** Carry out the conversion.
        byte[] result = client.ExtractKeyValuePairs(sourceFile, openOptions,
kvpSettings);

        if (result != null)
        {
            if (!Directory.Exists(targetFolder))
            {
                Directory.CreateDirectory(targetFolder);
            }

            string filename = Path.GetFileNameWithoutExtension(sourceFileName);
            string extension = Path.GetExtension($".{kvpSettings.KVPFormat}");
            string destinationFileName;
            destinationFileName = Path.Combine(targetFolder, filename + extension);
            using (FileStream fs = File.Create(destinationFileName))
            {
                fs.Write(result, 0, result.Length);
                fs.Close();
            }
            Console.WriteLine("File converted to " + destinationFileName);
        }
        else
        {
            Console.WriteLine("Nothing returned");
        }
    }
    catch (FaultException<WebServiceFaultException> ex)
    {
        Console.WriteLine($"FaultException occurred: ExceptionType:
{ex.Detail.ExceptionType.ToString()}");
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.ToString());
    }
    finally
    {
        if (client != null)
        {
            CloseService(client);
        }
    }
```

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 to use this functionality.

10.4 Testing and Troubleshooting

Although it is only a basic application, the Diagnostics Tool (included in the installation, as is the tool's full source code) can be used to test the Key-Value Pair operation of PDF Converter, including the Expected Keys.



File:

Expected key file::

Include
☑ Key bounding box ☑ Page number
☑ Value bounding box ☑ Type
☑ Confidence

Languages: eng

DPI: 300

KVP output format
● XML
○ JSON
○ CSV

```
============================================================================================
 Muhimbi Document Converter Diagnostics Tool                                (c) 2023 Muhimbi ltd
============================================================================================

To get started, please select a source file to convert and click the 'Convert File' button.

If the Document Conversion Service is not located on the same system as this application then please update the URL in the
'WS Configure' tab.

IF YOU ENCOUNTER ANY PROBLEMS THEN PLEASE MAKE SURE EVERYTHING HAS BEEN INSTALLED
IN LINE WITH CHAPTER 2 OF THE ADMINISTRATION GUIDE.

Troubleshooting guide: https://www.muhimbi.com/blog/troubleshooting-steps-for-pdf-converter-for-sharepoint-pdf-converter-
services/

============================================================================================
```

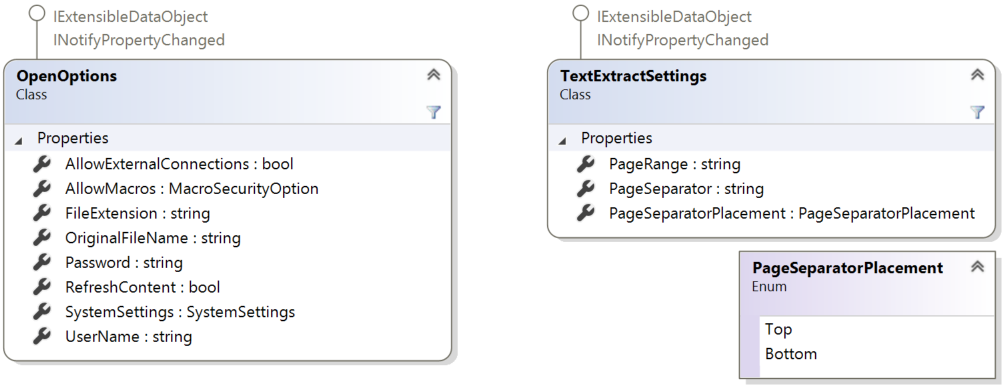
☑ Show converted file

In Progress: 0

The Expected Keys are supplied via file – you will need to edit the file externally to the Diagnostic Tool.

11 Text Extraction

11.1 Object Model



- **OpenOptions:** provides details about the file (original file name, extension), security (username and password, macros).
- **TextExtractSettings:** provides the settings for the text extraction. For more details see the TextExtractSettings class.

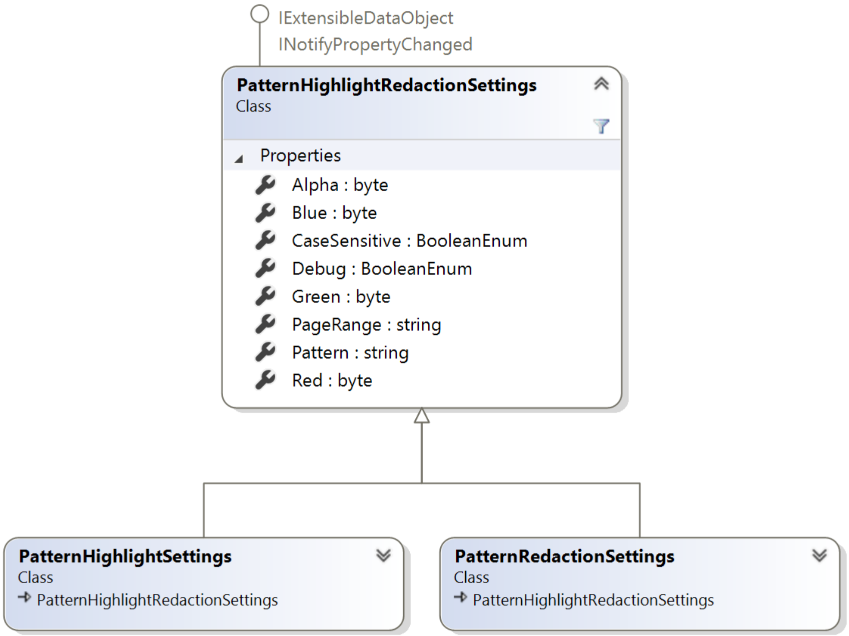
11.2 Extracting text in .NET

```
/// <summary>
/// Text Extraction
/// </summary>
/// <param name="sourceFileName">File to process</param>
/// <param name="targetFolder">Folder for the result of the OCR</param>
static void TextExtract(string sourceFileName, string targetFolder)
{
    DocumentConverterServiceClient client = null;
    try
    {
        if (!Directory.Exists(targetFolder))
        {
            Directory.CreateDirectory(targetFolder);
        }
        // ** Determine the source file and read it into a byte array.
        byte[] sourceFile = File.ReadAllBytes(sourceFileName);

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

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

        TextSettings textSettings = new TextSettings();
        textSettings.PageRange = "*"; // All pages

        // ** Carry out the Extraction.
        byte[] convFile = client.ExtractText(sourceFile, openOptions, textSettings);

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

        Console.WriteLine("Text extracted to " +
Path.GetFullPath(destinationFileName));
    }
    catch (FaultException<WebServiceFaultException> ex)
    {
        Console.WriteLine($"FaultException occurred: ExceptionType:
{ex.Detail.ExceptionType.ToString()}");
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.ToString());
    }
    finally
    {
        if (client != null)
        {
            CloseService(client);
        }
    }
}
```

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 to use this functionality.

12 Pattern Redaction and Highlighting

12.1 Object Model



Pattern Redaction and Pattern Highlighting use different Settings classes, but they inherit from a common class.
The only difference is that there is an Alpha property on the PatternHighlightSettings class (the Alpha property is fixed at 255 on the PatternRedactionSettings class).
For more details see 3.9.

12.2 Pattern Redaction in .NET

```
/// <summary>
/// Perform Pattern Redaction on the supplied file, writing the
/// result into the target folder
/// </summary>
/// <param name="ServiceURL">URL endpoint for the PDF Converter service</param>
/// <param name="sourceFileName">Source filename</param>
/// <param name="targetFolder">Target folder to receive the output file</param>
static void PatternRedaction(string ServiceURL, string sourceFileName, string
targetFolder)
{
    DocumentConverterServiceClient client = null;
    try
    {
        // Create minimum OpenOptions object
        OpenOptions openOptions = new OpenOptions();
        openOptions.OriginalFileName = Path.GetFileName(sourceFileName);
        // Create minimum PatternHighlightSettings
        PatternHighlightSettings patternHighlightSettings = new
PatternHighlightSettings();
        // Set the highlight color
        patternHighlightSettings.Red = 0;
        patternHighlightSettings.Green = 0;
        patternHighlightSettings.Blue = 255;
        patternHighlightSettings.Alpha = 255;
        patternHighlightSettings.Pattern = "\"374245455400126\"";
        // Create target folder if required
        if (!Directory.Exists(targetFolder))
        {
            Directory.CreateDirectory(targetFolder);
        }
        // ** Read the source file into a byte array.
        byte[] sourceFile = File.ReadAllBytes(sourceFileName);
        // ** Open the service and configure the bindings
        client = OpenService(ServiceURL);
        // ** Carry out the highlighting.
        byte[] result = client.PatternHighlight(sourceFile, openOptions,
patternHighlightSettings);
        // ** Save the results
        if (result != null)
        {
            if (!Directory.Exists(targetFolder))
            {
                Directory.CreateDirectory(targetFolder);
            }
            string filename = Path.GetFileNameWithoutExtension(sourceFileName);
            string destinationFileName = Path.GetFullPath(Path.Combine(targetFolder,
filename + "-highlighted.pdf"));
            using (FileStream fs = File.Create(destinationFileName))
            {
                fs.Write(result, 0, result.Length);
                fs.Close();
            }
            Console.WriteLine("File converted to " + destinationFileName);
            // Open the destination file
            ProcessStartInfo psi = new ProcessStartInfo();
            psi.FileName = destinationFileName;
            psi.UseShellExecute = true;
            Process.Start(psi);
        }
        else
        {
            Console.WriteLine("Nothing returned");
        }
    }
    catch (FaultException<WebServiceFaultException> ex)
    {
        Console.WriteLine($"FaultException occurred: ExceptionType:
{ex.Detail.ExceptionType.ToString()}");
        Console.WriteLine();
        Console.WriteLine($"Error Detail: {string.Join(Environment.NewLine,
ex.Detail.ExceptionDetails)}");
```

```
        Console.WriteLine($"Error message: {ex.Message}");
        Console.WriteLine();
        Console.WriteLine($"Error reason: {ex.Reason}");
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
        Console.WriteLine(ex.StackTrace);
        Console.WriteLine(ex.Data.ToString());
    }
    finally
    {
        if (client != null)
        {
            CloseService(client);
        }
    }
}
```

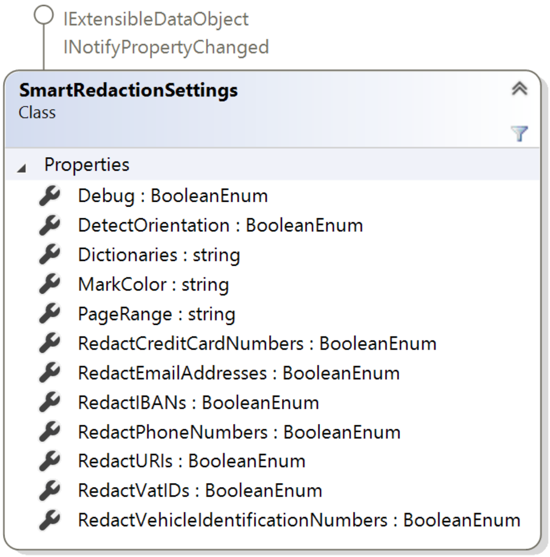
Please note that you need a *PDF Converter Professional* add-on license in addition to a valid *PDF Converter Services* or *PDF Converter for SharePoint* License to use this functionality.

12.3 Pattern Highlighting in .NET

```
/// <summary>
/// Perform Pattern Redaction on the supplied file, writing
/// the result into the target folder
/// </summary>
/// <param name="ServiceURL">URL endpoint for the PDF Converter service</param>
/// <param name="sourceFileName">Source filename</param>
/// <param name="targetFolder">Target folder to receive the output file</param>
static void PatternRedaction(string ServiceURL, string sourceFileName, string
targetFolder)
{
    DocumentConverterServiceClient client = null;
    try
    {
        // Create minimum OpenOptions object
        OpenOptions openOptions = new OpenOptions();
        openOptions.OriginalFileName = Path.GetFileName(sourceFileName);

        // Create minimum PatternRedactionSettings
        PatternRedactionSettings patternRedactionSettings = new
PatternRedactionSettings();
        // Set what needs to be redacted
        patternRedactionSettings.Red = 0;
        patternRedactionSettings.Green = 0;
        patternRedactionSettings.Blue = 255;
        patternRedactionSettings.Pattern = "\"374245455400126\"";

        // Create target folder if required
        if (!Directory.Exists(targetFolder))
        {
            Directory.CreateDirectory(targetFolder);
        }
        // ** Read the source file into a byte array.
        byte[] sourceFile = File.ReadAllBytes(sourceFileName);
        // ** Open the service and configure the bindings
        client = OpenService(ServiceURL);
        // ** Carry out the conversion.
        byte[] result = client.PatternRedaction(sourceFile, openOptions,
patternRedactionSettings);
        // ** Save the results
        if (result != null)
        {
            if (!Directory.Exists(targetFolder))
            {
                Directory.CreateDirectory(targetFolder);
            }
            string filename = Path.GetFileNameWithoutExtension(sourceFileName);
            string destinationFileName = Path.GetFullPath(Path.Combine(targetFolder,
filename + "-redacted.pdf"));
            using (FileStream fs = File.Create(destinationFileName))
            {
                fs.Write(result, 0, result.Length);
                fs.Close();
            }
            Console.WriteLine("File converted to " + destinationFileName);
            // Open the destination file
            ProcessStartInfo psi = new ProcessStartInfo();
            psi.FileName = destinationFileName;
            psi.UseShellExecute = true;
            Process.Start(psi);
        }
        else
        {
            Console.WriteLine("Nothing returned");
        }
    }
    catch (FaultException<WebServiceFaultException> ex)
    {
        Console.WriteLine($"FaultException occurred: ExceptionType:
{ex.Detail.ExceptionType.ToString()}");
        Console.WriteLine();
        Console.WriteLine($"Error Detail: {string.Join(Environment.NewLine,
ex.Detail.ExceptionDetails)}");
        Console.WriteLine($"Error message: {ex.Message}");
        Console.WriteLine();
        Console.WriteLine($"Error reason: {ex.Reason}");
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
        Console.WriteLine(ex.StackTrace);
        Console.WriteLine(ex.Data.ToString());
    }
    finally
    {
        if (client != null)
        {
            CloseService(client);
        }
    }
}
```

13 Smart Redaction

13.1 Object Model



IExtensibleDataObject
INotifyPropertyChanged

SmartRedactionSettings
Class

Properties

- Debug : BooleanEnum
- DetectOrientation : BooleanEnum
- Dictionaries : string
- MarkColor : string
- PageRange : string
- RedactCreditCardNumbers : BooleanEnum
- RedactEmailAddresses : BooleanEnum
- RedactIBANs : BooleanEnum
- RedactPhoneNumbers : BooleanEnum
- RedactURIs : BooleanEnum
- RedactVatIDs : BooleanEnum
- RedactVehicleIdentificationNumbers : BooleanEnum

For more details see 3.10.

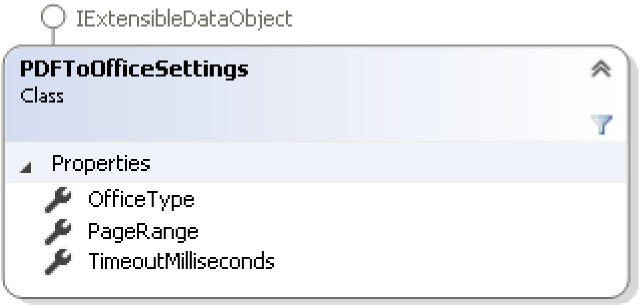
13.2 Smart Redaction in .NET

```
/// <summary>
/// Perform Smart Redaction on the supplied file, writing
/// the result into the target folder
/// </summary>
/// <param name="ServiceURL">URL endpoint for the PDF Converter service</param>
/// <param name="sourceFileName">Source filename</param>
/// <param name="targetFolder">Target folder to receive the output file</param>
static void SmartRedaction(string ServiceURL, string sourceFileName, string
targetFolder)
{
    DocumentConverterServiceClient client = null;
    try
    {
        // Create minimum OpenOptions object
        OpenOptions openOptions = new OpenOptions();
        openOptions.OriginalFileName = Path.GetFileName(sourceFileName);

        // Create minimum SmartRedactionSettings
        SmartRedactionSettings smartRedactionSettings = new SmartRedactionSettings();
        smartRedactionSettings = new SmartRedactionSettings();
        // Set what needs to be redacted
        smartRedactionSettings.RedactCreditCardNumbers = BooleanEnum.True;
        smartRedactionSettings.RedactEmailAddresses = BooleanEnum.True;
        smartRedactionSettings.RedactPhoneNumbers = BooleanEnum.True;

        // Create target folder if required
        if (!Directory.Exists(targetFolder))
        {
            Directory.CreateDirectory(targetFolder);
        }
        // ** Read the source file into a byte array.
        byte[] sourceFile = File.ReadAllBytes(sourceFileName);
        // ** Open the service and configure the bindings
        client = OpenService(ServiceURL);
        // ** Carry out the conversion.
        byte[] result = client.SmartRedaction(sourceFile, openOptions, 
smartRedactionSettings);
        // ** Save the results
        if (result != null)
        {
            if (!Directory.Exists(targetFolder))
            {
                Directory.CreateDirectory(targetFolder);
            }
            string filename = Path.GetFileNameWithoutExtension(sourceFileName);
            string destinationFileName = Path.GetFullPath(Path.Combine(targetFolder, 
filename + "-redacted.pdf"));
            using (FileStream fs = File.Create(destinationFileName))
            {
                fs.Write(result, 0, result.Length);
                fs.Close();
            }
            Console.WriteLine("File converted to " + destinationFileName);
        }
        else
        {
            Console.WriteLine("Nothing returned");
        }
    }
    catch (FaultException<WebServiceFaultException> ex)
    {
        Console.WriteLine($"FaultException occurred: ExceptionType: 
{ex.Detail.ExceptionType.ToString()}");
        Console.WriteLine();
        Console.WriteLine($"Error Detail: {string.Join(Environment.NewLine, 
ex.Detail.ExceptionDetails)}");
        Console.WriteLine($"Error message: {ex.Message}");
        Console.WriteLine();
        Console.WriteLine($"Error reason: {ex.Reason}");
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
        Console.WriteLine(ex.StackTrace);
        Console.WriteLine(ex.Data.ToString());
    }
    finally
    {
        if (client != null)
        {
            CloseService(client);
        }
    }
}
```

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 to use this functionality.

14 PDF to Office (Preview)

14.1 Object Model



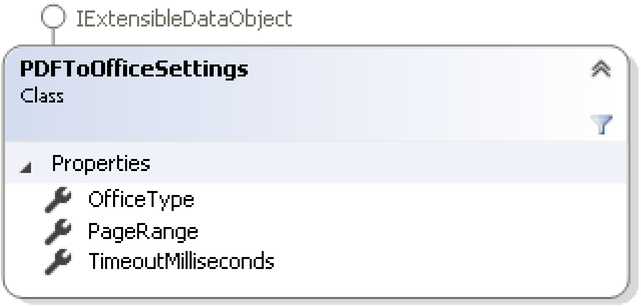
14.2 PDF to Office using .NET

This sample code will convert a PDF file to a Microsoft Office file. Though it will convert a PDF to SVG (an XML based 2D grahics format), SVG files are only one page, so a multipage PDF would only convert the first selected page.

```
        /// <summary>
        /// Perform PDF to DOCX on the supplied file, writing the result into 
the target folder
        /// </summary>
        /// <param name="ServiceURL">URL endpoint for the PDF Converter 
service</param>
        /// <param name="sourceFileName">Source filename</param>
        /// <param name="targetFolder">Target folder to receive the output 
file</param>
        /// <param name="officeType">Office type for output</param>
        static void PDFToOffice(string ServiceURL, string sourceFileName, string 
targetFolder, OfficeTypes officeType)
        {
            DocumentConverterServiceClient client = null;
            try
            {
                // Create minimum OpenOptions object
                OpenOptions openOptions = new OpenOptions();
                openOptions.OriginalFileName = Path.GetFileName(sourceFileName);
                // Create minimum PatternHighlightSettings
                PDFToOfficeSettings pDFToOfficeSettings = new 
PDFToOfficeSettings();
                pDFToOfficeSettings.OfficeType = officeType;
                pDFToOfficeSettings.PageRange = "*";
                // Create target folder if required
                if (!Directory.Exists(targetFolder))
                {
                    Directory.CreateDirectory(targetFolder);
                }
                // ** Read the source file into a byte array.
                byte[] sourceFile = File.ReadAllBytes(sourceFileName);
                // ** Open the service and configure the bindings
                client = OpenService(ServiceURL);
                // ** Carry out the conversion.
                byte[] result = client.PDFToOffice(sourceFile, openOptions,
pDFToOfficeSettings);
                // ** Save the results
                if (result != null)
                {
                    if (!Directory.Exists(targetFolder))
                    {
                        Directory.CreateDirectory(targetFolder);
                    }
                    string filename =
Path.GetFileNameWithoutExtension(sourceFileName);
                    string destinationFileName =
Path.GetFullPath(Path.Combine(targetFolder, filename +
$".{pDFToOfficeSettings.OfficeType}"));
                    using (FileStream fs = File.Create(destinationFileName))
                    {
                        fs.Write(result, 0, result.Length);
                        fs.Close();
                    }
                    Console.WriteLine("File converted to " +
destinationFileName);
                    // Open the destination file
                    ProcessStartInfo psi = new ProcessStartInfo();
                    psi.FileName = destinationFileName;
                    psi.UseShellExecute = true;
                    Process.Start(psi);
                }
                else
                {
                    Console.WriteLine("Nothing returned");
                }
            }
            catch (FaultException<WebServiceFaultException> ex)
            {
                Console.WriteLine($"FaultException occurred: ExceptionType:
{ex.Detail.ExceptionType.ToString()}");
                Console.WriteLine();
                Console.WriteLine($"Error Detail:
{string.Join(Environment.NewLine, ex.Detail.ExceptionDetails)}");
                Console.WriteLine($"Error message: {ex.Message}");
                Console.WriteLine();
                Console.WriteLine($"Error reason: {ex.Reason}");
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
                Console.WriteLine(ex.StackTrace);
                Console.WriteLine(ex.Data.ToString());
            }
            finally
            {
                if (client != null)
                {
                    CloseService(client);
                }
            }
        }
```

15 PDF to SVG

15.1 Object Model



15.2 PDF to SVG files using .NET

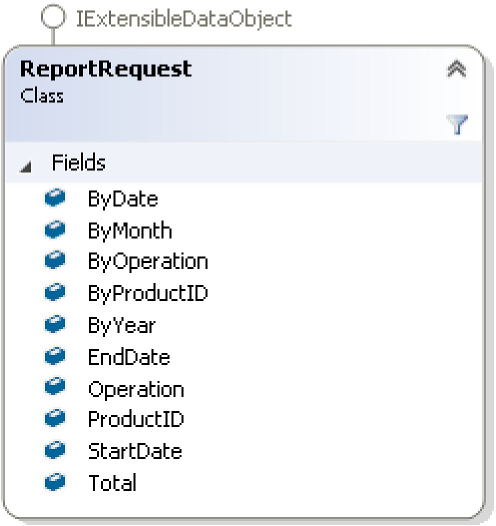
This sample code will convert a multi-page PDF file to a collection of **Scalable Vector Graphics** files.

```
        /// <summary>
        /// Convert a PDF to a collection of SVG files
        /// </summary>
        /// <param name="ServiceURL"></param>
        /// <param name="sourceFileName"></param>
        /// <param name="targetFolder"></param>
        static void PDFToSVG(string ServiceURL, string sourceFileName, string 
targetFolder)
        {
            DocumentConverterServiceClient client = null;
            try
            {
                // Create minimum OpenOptions object
                OpenOptions openOptions = new OpenOptions();
                openOptions.OriginalFileName = Path.GetFileName(sourceFileName);
                // Create minimum PatternHighlightSettings
                PDFToOfficeSettings pDFToOfficeSettings = new 
PDFToOfficeSettings();
                pDFToOfficeSettings.OfficeType = OfficeTypes.SVG;
                pDFToOfficeSettings.PageRange = "*";
                // Create target folder if required
                if (!Directory.Exists(targetFolder))
                {
                    Directory.CreateDirectory(targetFolder);
                }
                // ** Read the source file into a byte array.
                byte[] sourceFile = File.ReadAllBytes(sourceFileName);

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

                // Convert the pages to SVG files
                BatchResults batchResults = client.PDFToSVG(sourceFile, 
openOptions, pDFToOfficeSettings);

                // If results are returned
                if (batchResults != null && batchResults.Results != null &&
batchResults.Results.Length > 0)
                {
                    // For each result
                    foreach (BatchResult result in batchResults.Results)
                    {
                        // Get the filename
                        string filename = result.FileName;
                        string destinationFileName =
Path.GetFullPath(Path.Combine(targetFolder, filename));
                        Console.WriteLine(destinationFileName);
                        // Write the file content to a file
                        using (FileStream fs = File.Create(destinationFileName))
                        {
                            fs.Write(result.File, 0, result.File.Length);
                            fs.Close();
                        }
                        Console.WriteLine("File converted to " +
destinationFileName);

                    }
                }
                else
                {
                    Console.WriteLine("Nothing returned");
                }

            }
            catch (FaultException<WebServiceFaultException> ex)
            {
                Console.WriteLine($"FaultException occurred: ExceptionType:
{ex.Detail.ExceptionType.ToString()}");
                Console.WriteLine();
                Console.WriteLine($"Error Detail:
{string.Join(Environment.NewLine, ex.Detail.ExceptionDetails)}");
                Console.WriteLine($"Error message: {ex.Message}");
                Console.WriteLine();
                Console.WriteLine($"Error reason: {ex.Reason}");
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
                Console.WriteLine(ex.StackTrace);
                Console.WriteLine(ex.Data.ToString());
            }
            finally
            {
                if (client != null)
                {
                    CloseService(client);

                }
            }
        }
```

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 to use this functionality.

16 Instrumentation

16.1 Object Model



IExtensibleDataObject

ReportRequest
Class

Fields

- ByDate
- ByMonth
- ByOperation
- ByProductID
- ByYear
- EndDate
- Operation
- ProductID
- StartDate
- Total

16.2 Instrumentation Reports - CSV

The sample code for instrumentation features the OpenService and CloseService methods from Document Converter Service Client sample code.

```
        /// <summary>
        /// Get a usage report from Document Converter Services
        /// </summary>
        /// <param name="url">Service URL</param>
        /// <param name="targetFile">File to save the report data (optional)</param>
        /// <returns></returns>
        static string GetReport(string url, string targetFile = null)
        {
            DocumentConverterServiceClient client = null;
            string reportText = null;
            try
            {
                client = OpenService(url);
                ReportRequest reportRequest = new ReportRequest();
                reportRequest.ByOperation = BooleanEnum.True;
                reportRequest.ByDate = BooleanEnum.True;
                reportRequest.ProductID = 4;

                reportText = client.GetReport(reportRequest);
                if (!string.IsNullOrEmpty(targetFile))
                {
                    File.WriteAllText(targetFile, reportText);
                }

            }
            catch (FaultException<WebServiceFaultException> ex)
            {
                Console.WriteLine($"FaultException occurred: ExceptionType:
{ex.Detail.ExceptionType.ToString()}");
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.ToString());
            }
            finally
            {
                if (client != null)
                {
                    CloseService(client);

                }
            }
            return reportText;
        }
```

16.3 Instrumentation Reports – PDF

The sample code for instrumentation features the OpenService and CloseService methods from Document Converter Service Client sample code.

```
        static void Main(string[] args)
        {
            // Output result file name
            string reportFile = Path.GetFullPath("report.pdf");
            // Create and populate Report Request
            ReportRequest reportRequest = new ReportRequest();
            reportRequest.ByYear = BooleanEnum.True;
            reportRequest.StartDate = "2025-01-01";
            reportRequest.EndDate = "2026-01-01";
            // Get the report
            bool success = GetPDFReport(reportFile, reportRequest);
        }
        /// <summary>
        /// Get a PDF report
        /// </summary>
        /// <param name="reportOutputFile">Report file name</param>
        /// <param name="reportRequest">Report Request settings</param>
        /// <returns></returns>
        static bool GetPDFReport(string reportOutputFile, ReportRequest reportRequest)
        {
            bool success = false;
            DocumentConverterServiceClient client = null;
            try
            {
                // Open the client
                client = new DocumentConverterServiceClient();
                // If the directory exists,
                if (Directory.Exists(Path.GetDirectoryName(reportOutputFile)))
                {
                    // If the report file exists, delete it
                    if (File.Exists(reportOutputFile))
                    {
                        File.Delete(reportOutputFile);
                    }
                }
                // Otherwise create it
                else
                {
                    Directory.CreateDirectory(Path.GetDirectoryName(reportOutputFile));
                }
                // Get the report
                byte[] report = client.GetPDFReport(reportRequest);
                if (report != null)
                {
                    // ** Write the result back to the file reportOutputFile.
                    File.WriteAllBytes(reportOutputFile, report);
                    Console.WriteLine($"Result saved to {reportOutputFile}");
                    success = true;
                }
                else
                {
                    Console.WriteLine("Returned null");
                    success = false;
                }
            }
            finally
            {
                // If the client is not null,
                if (client != null)
                {
                    // Close the client
                    client.Close();
                }
            }
            // Return success status
            return success;
        }
```

17 Troubleshooting

Although Muhimbi Document Converter Service 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.

17.1 Problems parsing the WSDL

By default, the Conversion Service uses the host name of the local system 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* to the system's IP-address.

To change this, modify the *baseAddress* attribute in the configuration file and restart the service. For details see this Knowledge Base article.

17.2 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.

17.3 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 / 2013.

17.4 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.

17.5 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 My InfoPath form fails to convert, how can I troubleshoot this? (muhimbi.com)

17.6 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
- Extract PDF Forms Data (FDF, XFDF, XML) using SharePoint or C#, Java, PHP
- 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)
- 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 xslx, 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

Appendix – Licensing

All Muhimbi products are licensed in a way that allows maximum flexibility. Please familiarise yourself with the licensing agreement, particularly section 2 – *License Grants*, before purchasing our software.

For details see:

1. Master Subscription Agreement.
2. Details about pricing & licensing.

In summary we support the following license types.

1. **Free evaluation version:** If the software is installed without a license then you are using the evaluation version. The software is fully functional without any time limits, but a small evaluation watermark will be displayed in each processed document. It is not permitted to use the evaluation version in production environments. Support is provided using any of the means in the Support area on our website.
2. **Basic License:** Starter edition for environments consisting of a single conversion server.
3. **Small Farm License:** Covers up to 3 individual servers, either stand-alone or load balanced.
4. **Enterprise License:** Covers an unlimited number of Servers, Developers and Users in a single legal entity.
5. **OEM License:** If you wish to bundle our software with your own solution and redistribute it to 3rd parties then you require an OEM License. Please read the details in the Software License Agreement for more information.

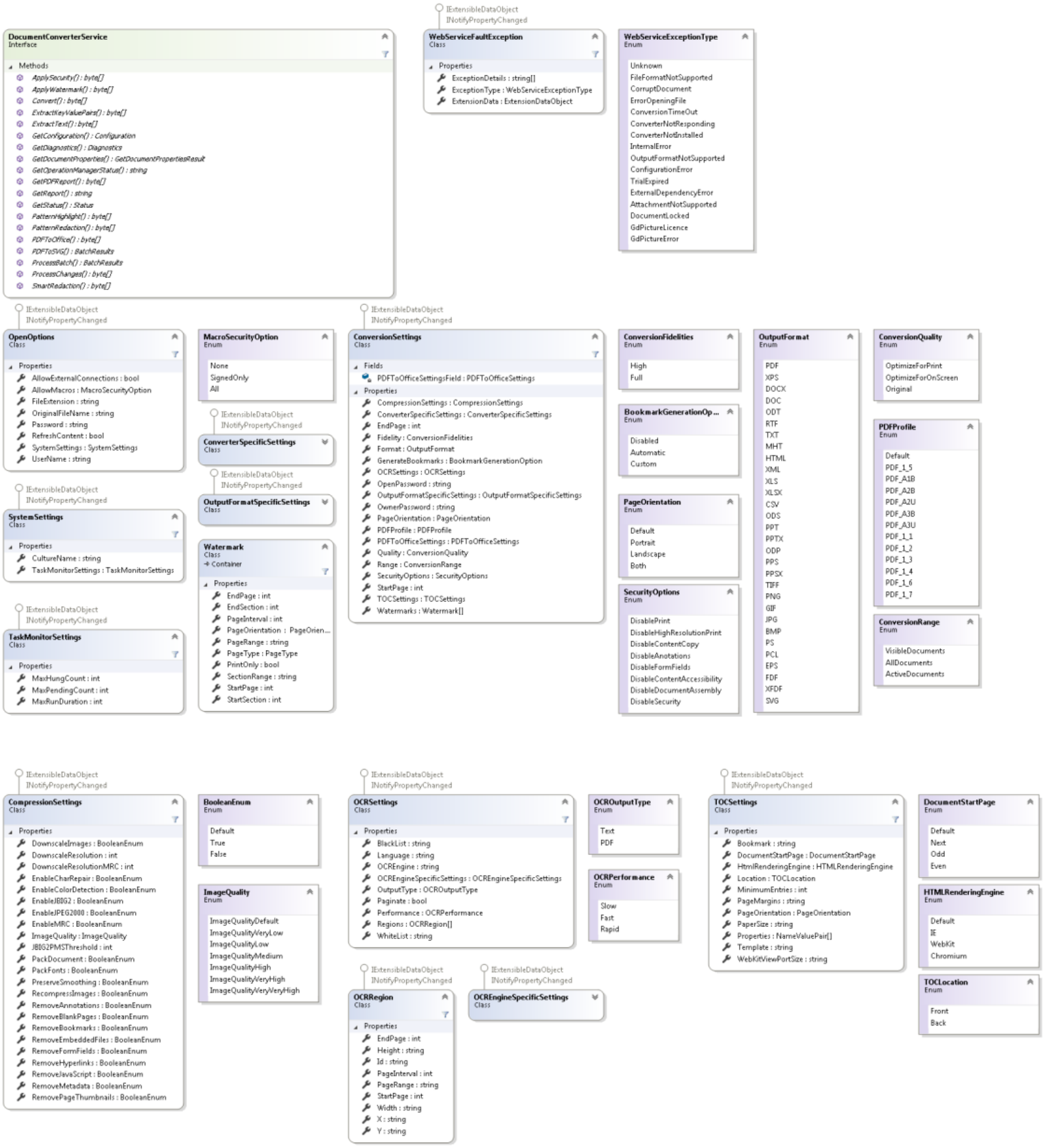
   Note that you are not allowed to use our Products to develop derived works that offer similar functionality as the Product or expose the features of the Product for use by an unlicensed third party unless agreed with Muhimbi. From a licensing perspective embedding our software in a SAAS based solution is considered redistributing our software and requires an OEM license.

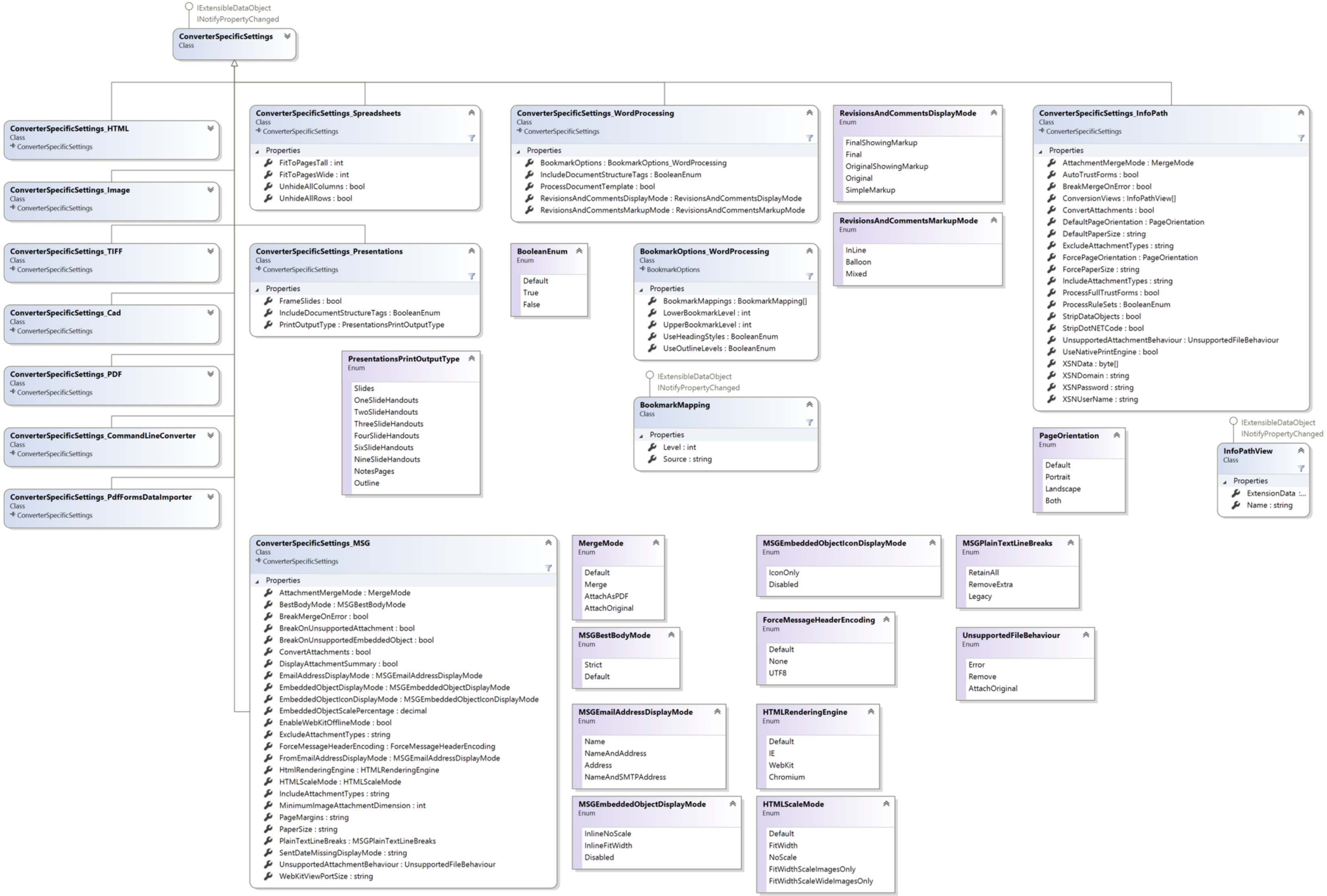
Please note that some older license types have been discontinued. However, these are still valid for those customers that have purchased them in the past. Please see the License Agreement for details about these old-style licenses.

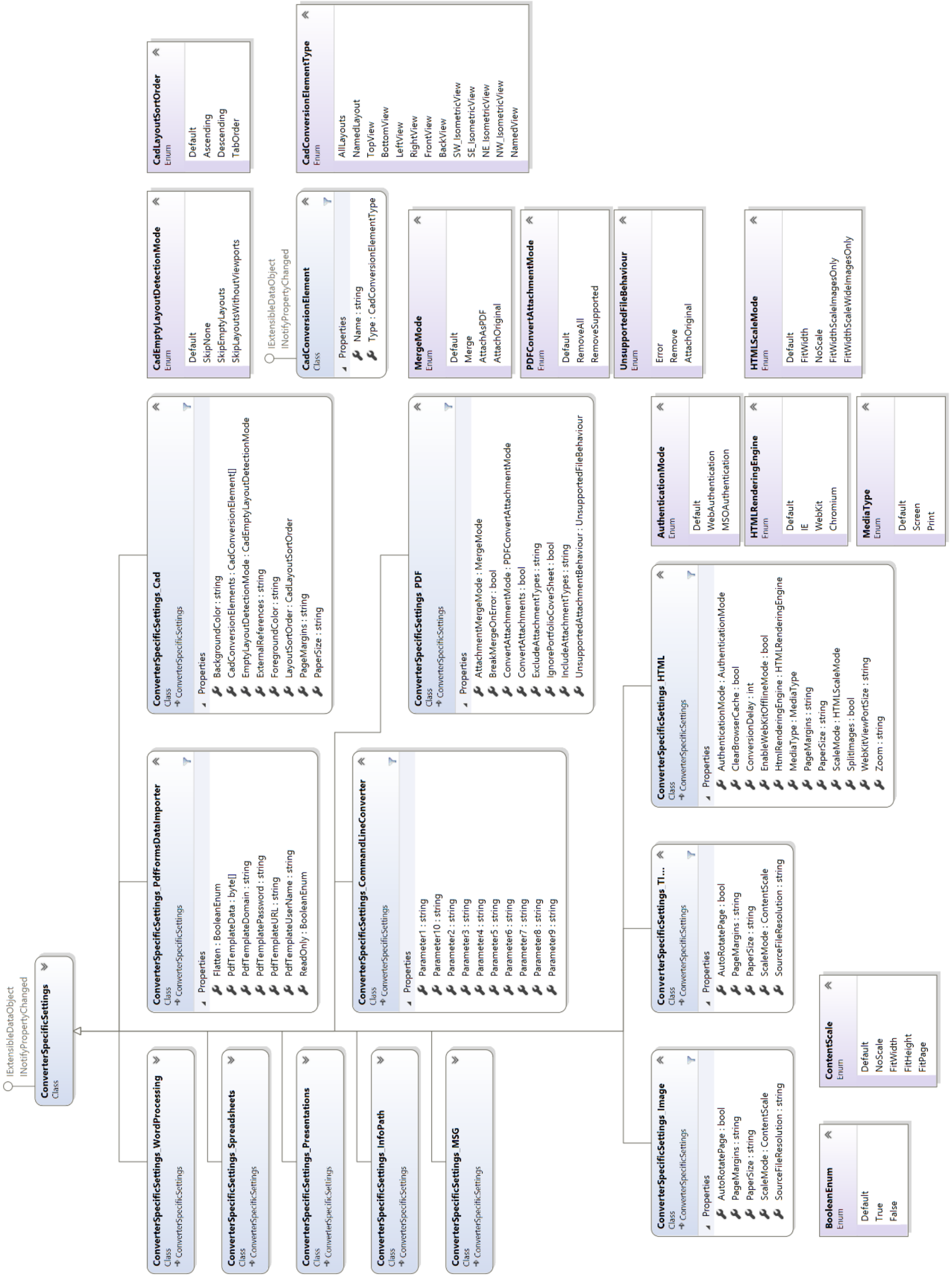
The PDF Converter for SharePoint license is limited to use from SharePoint environments only. If you wish to invoke the PDF Conversion Service from a non-SharePoint based environment, e.g. Java, .NET or any other Web Services capable system then you will need to purchase a license for the PDF Converter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

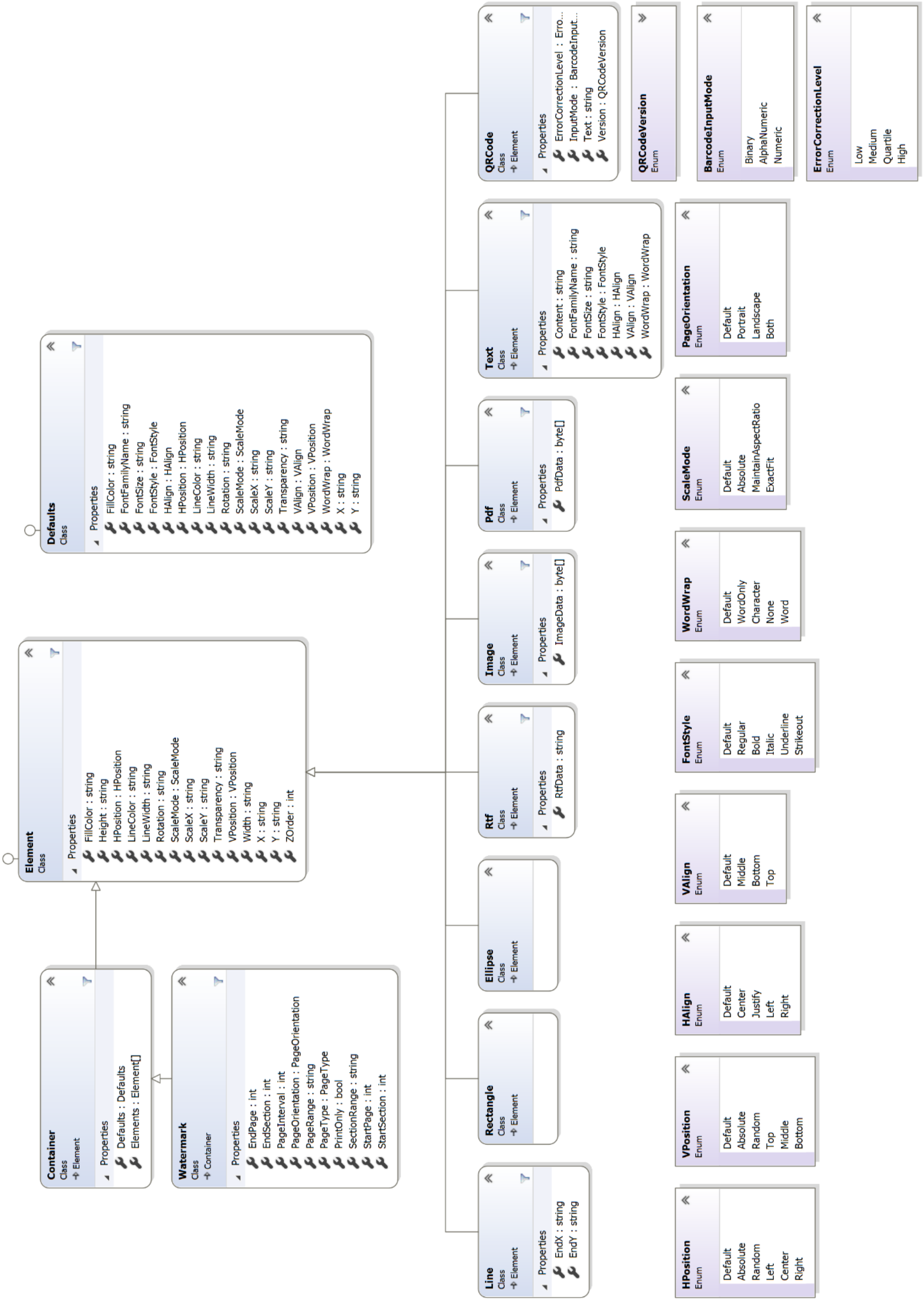
The PDF Converter Professional license is an add-on that adds additional functionality to either the PDF Converter for SharePoint or the PDF Converter Services. This functionality, e.g. PDF/A post-processing and OCR, is usually associated with more complex environments and has therefore been separated 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.

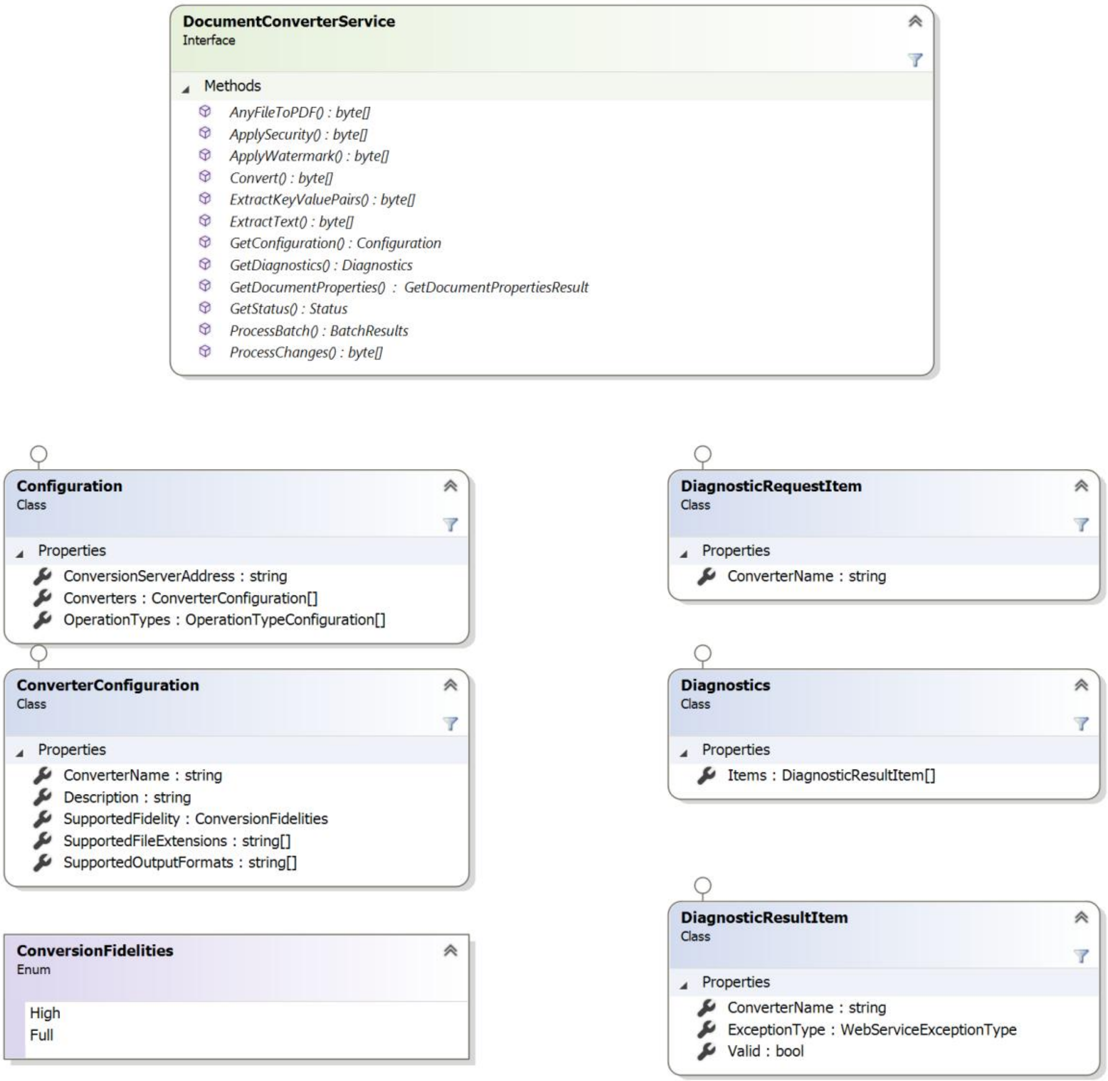
Appendix – Class Diagrams




IExtensibleDataObject
INotifyPropertyChanged
ConverterSpecificSettings
Class
ConverterSpecificSettings_HTML
Class
ConverterSpecificSettings
ConverterSpecificSettings_Image
Class
ConverterSpecificSettings
ConverterSpecificSettings_TIFF
Class
ConverterSpecificSettings
ConverterSpecificSettings_Cad
Class
ConverterSpecificSettings
ConverterSpecificSettings_PDF
Class
ConverterSpecificSettings
ConverterSpecificSettings_CommandLineConverter
Class
ConverterSpecificSettings
ConverterSpecificSettings_PdfFormsDataImporter
Class
ConverterSpecificSettings
ConverterSpecificSettings_Spreadsheets
Class
ConverterSpecificSettings
Properties
FitToPagesTall : int
FitToPagesWide : int
UnhideAllColumns : bool
UnhideAllRows : bool
ConverterSpecificSettings_Presentations
Class
ConverterSpecificSettings
Properties
FrameSlides : bool
IncludeDocumentStructureTags : BooleanEnum
PrintOutputType : PresentationsPrintOutputType
PresentationsPrintOutputType
Enum
Slides
OneSlideHandouts
TwoSlideHandouts
ThreeSlideHandouts
FourSlideHandouts
SixSlideHandouts
NineSlideHandouts
NotesPages
Outline
ConverterSpecificSettings_WordProcessing
Class
ConverterSpecificSettings
Properties
BookmarkOptions : BookmarkOptions_WordProcessing
IncludeDocumentStructureTags : BooleanEnum
ProcessDocumentTemplate : bool
RevisionsAndCommentsDisplayMode : RevisionsAndCommentsDisplayMode
RevisionsAndCommentsMarkupMode : RevisionsAndCommentsMarkupMode
BooleanEnum
Enum
Default
True
False
BookmarkOptions_WordProcessing
Class
BookmarkOptions
Properties
BookmarkMappings : BookmarkMapping[]
LowerBookmarkLevel : int
UpperBookmarkLevel : int
UseHeadingStyles : BooleanEnum
UseOutlineLevels : BooleanEnum
IExtensibleDataObject
INotifyPropertyChanged
BookmarkMapping
Class
Properties
Level : int
Source : string
RevisionsAndCommentsDisplayMode
Enum
FinalShowingMarkup
Final
OriginalShowingMarkup
Original
SimpleMarkup
RevisionsAndCommentsMarkupMode
Enum
InLine
Balloon
Mixed
ConverterSpecificSettings_InfoPath
Class
ConverterSpecificSettings
Properties
AttachmentMergeMode : MergeMode
AutoTrustForms : bool
BreakMergeOnError : bool
ConversionViews : InfoPathView[]
ConvertAttachments : bool
DefaultPageOrientation : PageOrientation
DefaultPaperSize : string
ExcludeAttachmentTypes : string
ForcePageOrientation : PageOrientation
ForcePaperSize : string
IncludeAttachmentTypes : string
ProcessFullTrustForms : bool
ProcessRuleSets : BooleanEnum
StripDataObjects : bool
StripDotNETCode : bool
UnsupportedAttachmentBehaviour : UnsupportedFileBehaviour
UseNativePrintEngine : bool
XSNData : byte[]
XSNDomain : string
XSNPassword : string
XSNUserName : string
PageOrientation
Enum
Default
Portrait
Landscape
Both
IExtensibleDataObject
INotifyPropertyChanged
InfoPathView
Class
Properties
ExtensionData : ...
Name : string
ConverterSpecificSettings_MSG
Class
ConverterSpecificSettings
Properties
AttachmentMergeMode : MergeMode
BestBodyMode : MSGBestBodyMode
BreakMergeOnError : bool
BreakOnUnsupportedAttachment : bool
BreakOnUnsupportedEmbeddedObject : bool
ConvertAttachments : bool
DisplayAttachmentSummary : bool
EmailAddressDisplayMode : MSGEmailAddressDisplayMode
EmbeddedObjectDisplayMode : MSGEmbeddedObjectDisplayMode
EmbeddedObjectIconDisplayMode : MSGEmbeddedObjectIconDisplayMode
EmbeddedObjectScalePercentage : decimal
EnableWebKitOfflineMode : bool
ExcludeAttachmentTypes : string
ForceMessageHeaderEncoding : ForceMessageHeaderEncoding
FromEmailAddressDisplayMode : MSGEmailAddressDisplayMode
HtmlRenderingEngine : HTMLRenderingEngine
HTMLScaleMode : HTMLScaleMode
IncludeAttachmentTypes : string
MinimumImageAttachmentDimension : int
PageMargins : string
PaperSize : string
PlainTextLineBreaks : MSGPlainTextLineBreaks
SentDateMissingDisplayMode : string
UnsupportedAttachmentBehaviour : UnsupportedFileBehaviour
WebKitViewPortSize : string
MergeMode
Enum
Default
Merge
AttachAsPDF
AttachOriginal
MSGBestBodyMode
Enum
Strict
Default
MSGEmailAddressDisplayMode
Enum
Name
NameAndAddress
Address
NameAndSMTPAddress
MSGEmbeddedObjectDisplayMode
Enum
InlineNoScale
InlineFitWidth
Disabled
MSGEmbeddedObjectIconDisplayMode
Enum
IconOnly
Disabled
ForceMessageHeaderEncoding
Enum
Default
None
UTF8
HTMLRenderingEngine
Enum
Default
IE
WebKit
Chromium
HTMLScaleMode
Enum
Default
FitWidth
NoScale
FitWidthScaleImagesOnly
FitWidthScaleWideImagesOnly
MSGPlainTextLineBreaks
Enum
RetainAll
RemoveExtra
Legacy
UnsupportedFileBehaviour
Enum
Error
Remove
AttachOriginal


CadLayoutSortOrder
Enum
Default
Ascending
Descending
TabOrder
CadConversionElementType
Enum
AllLayouts
NamedLayout
TopView
BottomView
LeftView
RightView
FrontView
BackView
SW_IsometricView
SE_IsometricView
NE_IsometricView
NW_IsometricView
NamedView
CadEmptyLayoutDetectionMode
Enum
Default
SkipNone
SkipEmptyLayouts
SkipLayoutsWithoutViewports
IExtensibleDataObject
INotifyPropertyChanged
CadConversionElement
Class
Properties
Name : string
Type : CadConversionElementType
MergeMode
Enum
Default
Merge
AttachAsPDF
AttachOriginal
PDFConvertAttachmentMode
Enum
Default
RemoveAll
RemoveSupported
UnsupportedFileBehaviour
Enum
Error
Remove
AttachOriginal
HTMLScaleMode
Enum
Default
FitWidth
NoScale
FitWidthScaleImagesOnly
FitWidthScaleWideImagesOnly
ConverterSpecificSettings_Cad
Class
ConverterSpecificSettings
Properties
BackgroundColor : string
CadConversionElements : CadConversionElement[]
EmptyLayoutDetectionMode : CadEmptyLayoutDetectionMode
ExternalReferences : string
ForegroundColor : string
LayoutSortOrder : CadLayoutSortOrder
PageMargins : string
PaperSize : string
ConverterSpecificSettings_PDF
Class
ConverterSpecificSettings
Properties
AttachmentMergeMode : MergeMode
BreakMergeOnError : bool
ConvertAttachmentMode : PDFConvertAttachmentMode
ConvertAttachments : bool
ExcludeAttachmentTypes : string
IgnorePortfolioCoverSheet : bool
IncludeAttachmentTypes : string
UnsupportedAttachmentBehaviour : UnsupportedFileBehaviour
AuthenticationMode
Enum
Default
WebAuthentication
MSOAuthentication
HTMLRenderingEngine
Enum
Default
IE
WebKit
Chromium
MediaType
Enum
Default
Screen
Print
ConverterSpecificSettings_HTML
Class
ConverterSpecificSettings
Properties
AuthenticationMode : AuthenticationMode
ClearBrowserCache : bool
ConversionDelay : int
EnableWebKitOfflineMode : bool
HtmlRenderingEngine : HTMLRenderingEngine
MediaType : MediaType
PageMargins : string
PaperSize : string
ScaleMode : HTMLScaleMode
SplitImages : bool
WebKitViewPortSize : string
Zoom : string
ConverterSpecificSettings_PdfFormsDataImporter
Class
ConverterSpecificSettings
Properties
Flatten : BooleanEnum
PdfTemplateData : byte[]
PdfTemplateDomain : string
PdfTemplatePassword : string
PdfTemplateURL : string
PdfTemplateUserName : string
ReadOnly : BooleanEnum
ConverterSpecificSettings_CommandLineConverter
Class
ConverterSpecificSettings
Properties
Parameter1 : string
Parameter10 : string
Parameter2 : string
Parameter3 : string
Parameter4 : string
Parameter5 : string
Parameter6 : string
Parameter7 : string
Parameter8 : string
Parameter9 : string
ConverterSpecificSettings_TI...
Class
ConverterSpecificSettings
Properties
AutoRotatePage : bool
PageMargins : string
PaperSize : string
ScaleMode : ContentScale
SourceFileResolution : string
IExtensibleDataObject
INotifyPropertyChanged
ConverterSpecificSettings
Class
ContentScale
Enum
Default
NoScale
FitWidth
FitHeight
FitPage
ConverterSpecificSettings_WordProcessing
Class
ConverterSpecificSettings
ConverterSpecificSettings_Spreadsheets
Class
ConverterSpecificSettings
ConverterSpecificSettings_Presentations
Class
ConverterSpecificSettings
ConverterSpecificSettings_InfoPath
Class
ConverterSpecificSettings
ConverterSpecificSettings_MSG
Class
ConverterSpecificSettings
ConverterSpecificSettings_Image
Class
ConverterSpecificSettings
Properties
AutoRotatePage : bool
PageMargins : string
PaperSize : string
ScaleMode : ContentScale
SourceFileResolution : string
BooleanEnum
Enum
Default
True
False


Defaults
Class
Properties
FillColor : string
FontFamilyName : string
FontSize : string
FontStyle : FontStyle
HAlign : HAlign
HPosition : HPosition
LineColor : string
LineWidth : string
Rotation : string
ScaleMode : ScaleMode
ScaleX : string
ScaleY : string
Transparency : string
VAlign : VAlign
VPosition : VPosition
WordWrap : WordWrap
X : string
Y : string
Element
Class
Properties
FillColor : string
Height : string
HPosition : HPosition
LineColor : string
LineWidth : string
Rotation : string
ScaleMode : ScaleMode
ScaleX : string
ScaleY : string
Transparency : string
VPosition : VPosition
Width : string
X : string
Y : string
ZOrder : int
Container
Class
Element
Properties
Defaults : Defaults
Elements : Element[]
Watermark
Class
Container
Properties
EndPage : int
EndSection : int
PageInterval : int
PageOrientation : PageOrientation
PageRange : string
PageType : PageType
PrintOnly : bool
SectionRange : string
StartPage : int
StartSection : int
QRCode
Class
Element
Properties
ErrorCorrectionLevel : Erro...
InputMode : BarcodeInput...
Text : string
Version : QRCodeVersion
Text
Class
Element
Properties
Content : string
FontFamilyName : string
FontSize : string
FontStyle : FontStyle
HAlign : HAlign
VAlign : VAlign
WordWrap : WordWrap
Pdf
Class
Element
Properties
PdfData : byte[]
Image
Class
Element
Properties
ImageData : byte[]
Rtf
Class
Element
Properties
RtfData : string
Ellipse
Class
Element
Rectangle
Class
Element
Line
Class
Element
Properties
EndX : string
EndY : string
QRCodeVersion
Enum
BarcodeInputMode
Enum
Binary
AlphaNumeric
Numeric
ErrorCorrectionLevel
Enum
Low
Medium
Quartile
High
PageOrientation
Enum
Default
Portrait
Landscape
Both
ScaleMode
Enum
Default
Absolute
MaintainAspectRatio
ExactFit
WordWrap
Enum
Default
WordOnly
Character
None
Word
FontStyle
Enum
Default
Regular
Bold
Italic
Underline
Strikeout
VAlign
Enum
Default
Middle
Bottom
Top
HAlign
Enum
Default
Center
Justify
Left
Right
VPosition
Enum
Default
Absolute
Random
Top
Middle
Bottom
HPosition
Enum
Default
Absolute
Random
Left
Center
Right



DocumentConverterService
Interface

Methods

- *AnyFileToPDF() : byte[]*
- *ApplySecurity() : byte[]*
- *ApplyWatermark() : byte[]*
- *Convert() : byte[]*
- *ExtractKeyValuePairs() : byte[]*
- *ExtractText() : byte[]*
- *GetConfiguration() : Configuration*
- *GetDiagnostics() : Diagnostics*
- *GetDocumentProperties() : GetDocumentPropertiesResult*
- *GetStatus() : Status*
- *ProcessBatch() : BatchResults*
- *ProcessChanges() : byte[]*

Configuration
Class

Properties

- ConversionServerAddress : string
- Converters : ConverterConfiguration[]
- OperationTypes : OperationTypeConfiguration[]

ConverterConfiguration
Class

Properties

- ConverterName : string
- Description : string
- SupportedFidelity : ConversionFidelities
- SupportedFileExtensions : string[]
- SupportedOutputFormats : string[]

ConversionFidelities
Enum

High
Full

DiagnosticRequestItem
Class

Properties

- ConverterName : string

Diagnostics
Class

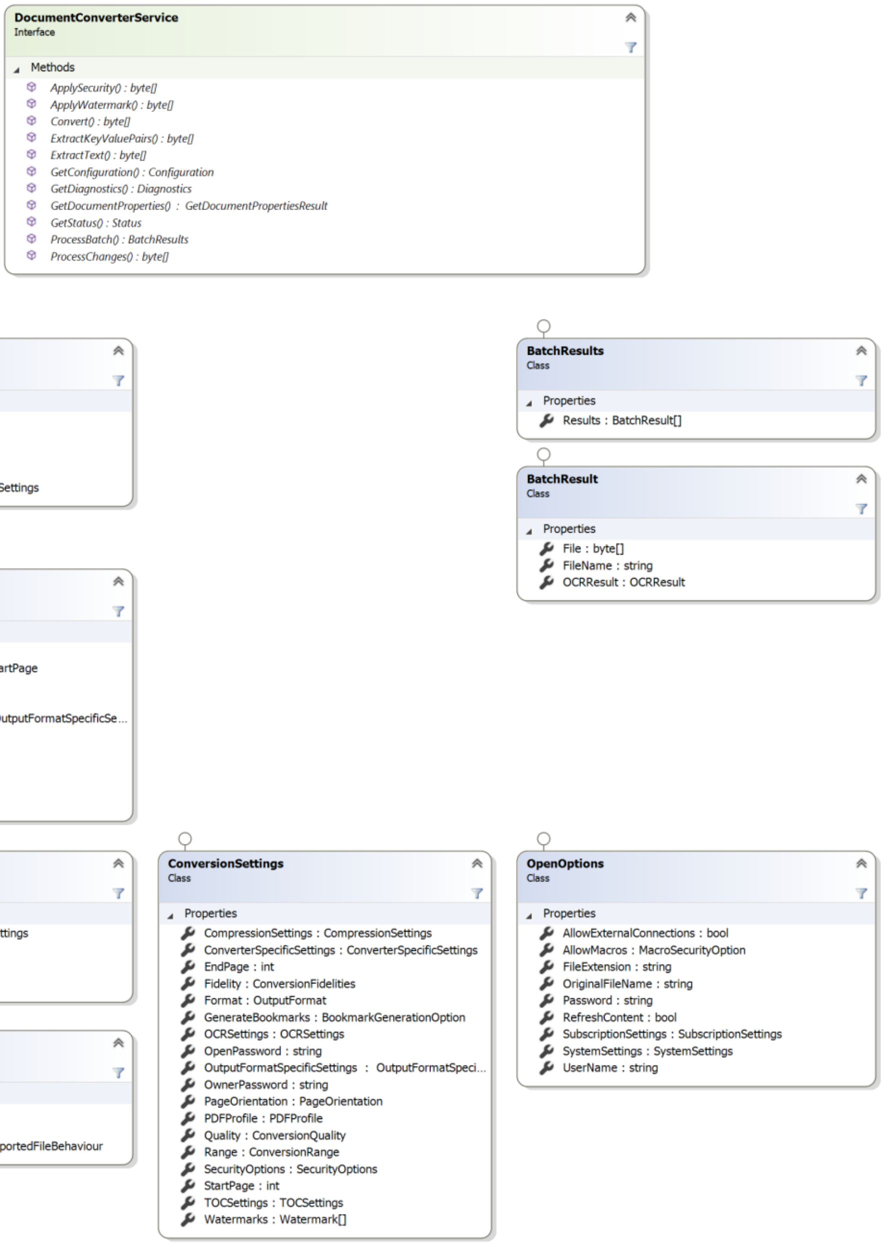
Properties

- Items : DiagnosticResultItem[]

DiagnosticResultItem
Class

Properties

- ConverterName : string
- ExceptionType : WebServiceExceptionType
- Valid : bool



DocumentConverterService
Interface

Methods

- *ApplySecurity() : byte[]*
- *ApplyWatermark() : byte[]*
- *Convert() : byte[]*
- *ExtractKeyValuePairs() : byte[]*
- *ExtractText() : byte[]*
- *GetConfiguration() : Configuration*
- *GetDiagnostics() : Diagnostics*
- *GetDocumentProperties() : GetDocumentPropertiesResult*
- *GetStatus() : Status*
- *ProcessBatch() : BatchResults*
- *ProcessChanges() : byte[]*

ProcessingOptions
Class

Properties

- MergeSettings : MergeSettings
- OCRSettings : OCRSettings
- SourceFiles : SourceFile[]
- SplitOptions : FileSplitOptions
- SubscriptionSettings : SubscriptionSettings

BatchResults
Class

Properties

- Results : BatchResult[]

BatchResult
Class

Properties

- File : byte[]
- FileName : string
- OCRResult : OCRResult

MergeSettings
Class

Properties

- BreakOnError : bool
- DocumentStartPage : DocumentStartPage
- OmitErrorPages : bool
- OpenPassword : string
- OutputFormatSpecificSettings : OutputFormatSpecificSe...
- OwnerPassword : string
- PDFProfile : PDFProfile
- SecurityOptions : SecurityOptions
- TOCSettings : TOCSettings
- Watermarks : Watermark[]

SourceFile
Class

Properties

- ConversionSettings : ConversionSettings
- File : byte[]
- MergeSettings : FileMergeSettings
- OpenOptions : OpenOptions

FileMergeSettings
Class

Properties

- MergeMode : MergeMode
- TopLevelBookmark : string
- UnsupportedFileBehaviour : UnsupportedFileBehaviour

ConversionSettings
Class

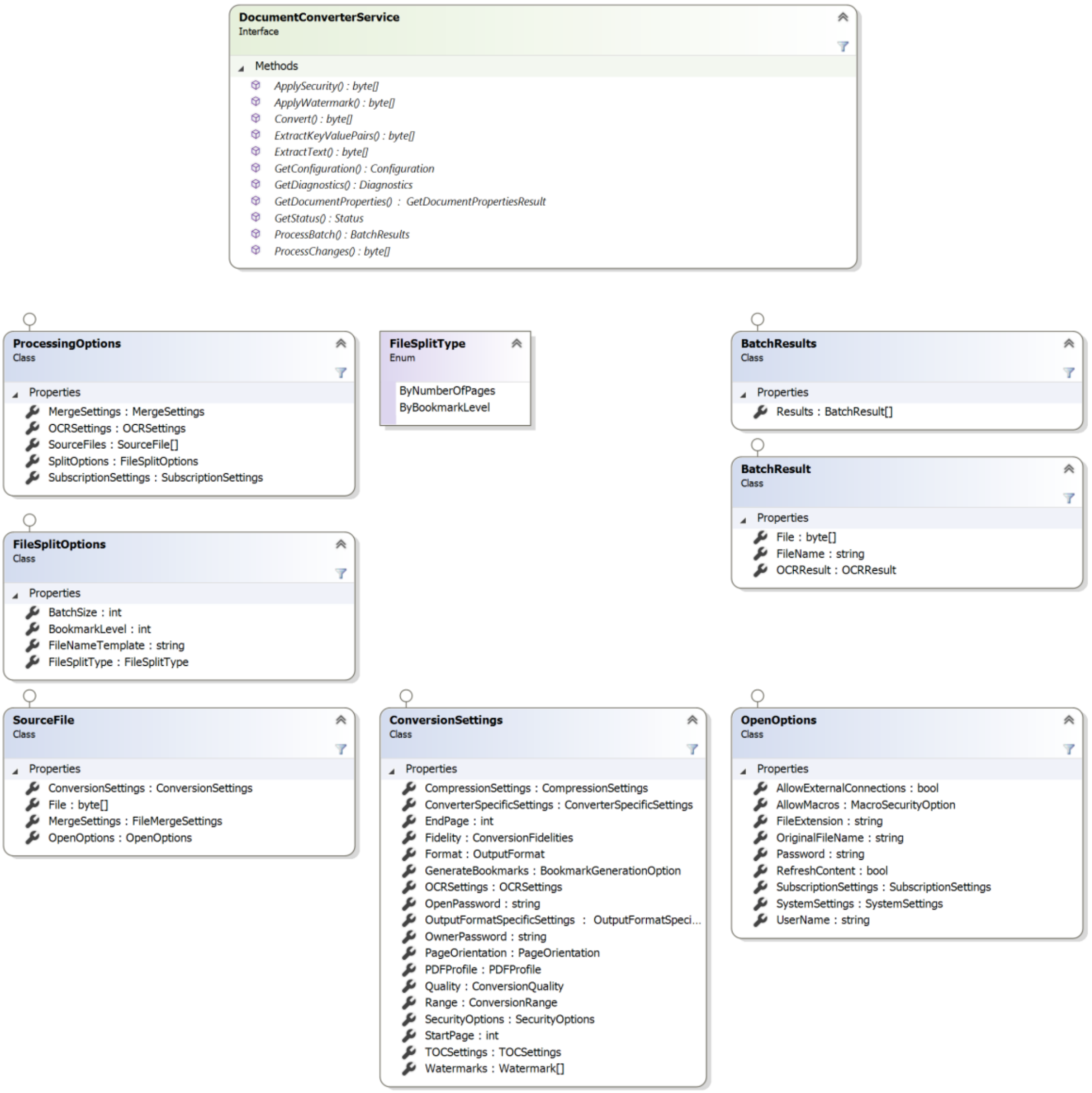
Properties

- CompressionSettings : CompressionSettings
- ConverterSpecificSettings : ConverterSpecificSettings
- EndPage : int
- Fidelity : ConversionFidelities
- Format : OutputFormat
- GenerateBookmarks : BookmarkGenerationOption
- OCRSettings : OCRSettings
- OpenPassword : string
- OutputFormatSpecificSettings : OutputFormatSpeci...
- OwnerPassword : string
- PageOrientation : PageOrientation
- PDFProfile : PDFProfile
- Quality : ConversionQuality
- Range : ConversionRange
- SecurityOptions : SecurityOptions
- StartPage : int
- TOCSettings : TOCSettings
- Watermarks : Watermark[]

OpenOptions
Class

Properties

- AllowExternalConnections : bool
- AllowMacros : MacroSecurityOption
- FileExtension : string
- OriginalFileName : string
- Password : string
- RefreshContent : bool
- SubscriptionSettings : SubscriptionSettings
- SystemSettings : SystemSettings
- UserName : string



DocumentConverterService
Interface

Methods
- ApplySecurity() : byte[]
- ApplyWatermark() : byte[]
- Convert() : byte[]
- ExtractKeyValuePairs() : byte[]
- ExtractText() : byte[]
- GetConfiguration() : Configuration
- GetDiagnostics() : Diagnostics
- GetDocumentProperties() : GetDocumentPropertiesResult
- GetStatus() : Status
- ProcessBatch() : BatchResults
- ProcessChanges() : byte[]

ProcessingOptions
Class

Properties
- MergeSettings : MergeSettings
- OCRSettings : OCRSettings
- SourceFiles : SourceFile[]
- SplitOptions : FileSplitOptions
- SubscriptionSettings : SubscriptionSettings

FileSplitType
Enum

- ByNumberOfPages
- ByBookmarkLevel

BatchResults
Class

Properties
- Results : BatchResult[]

BatchResult
Class

Properties
- File : byte[]
- FileName : string
- OCRResult : OCRResult

FileSplitOptions
Class

Properties
- BatchSize : int
- BookmarkLevel : int
- FileNameTemplate : string
- FileSplitType : FileSplitType

SourceFile
Class

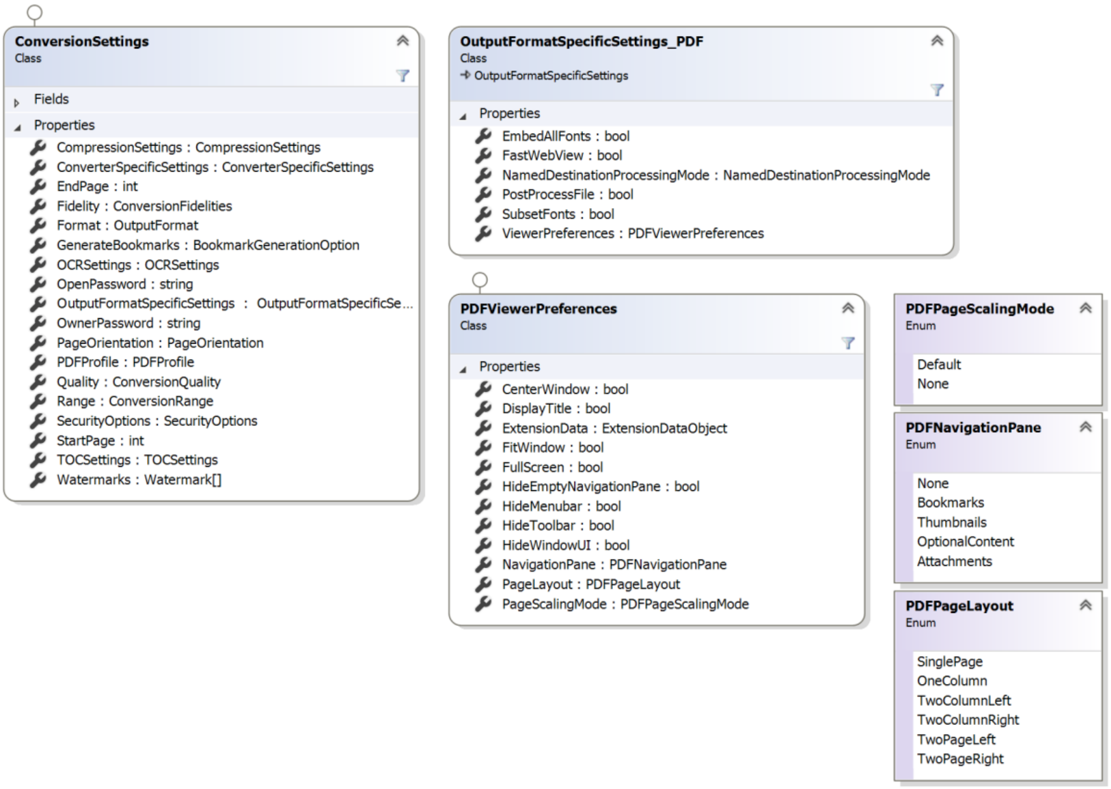
Properties
- ConversionSettings : ConversionSettings
- File : byte[]
- MergeSettings : FileMergeSettings
- OpenOptions : OpenOptions

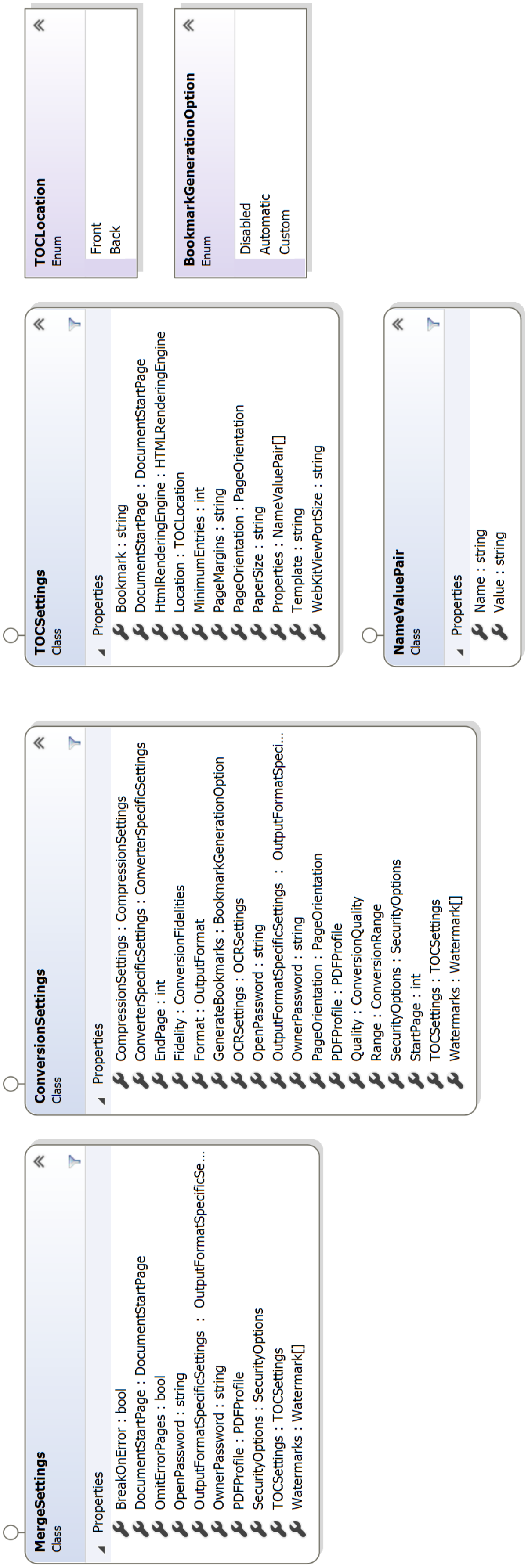
ConversionSettings
Class

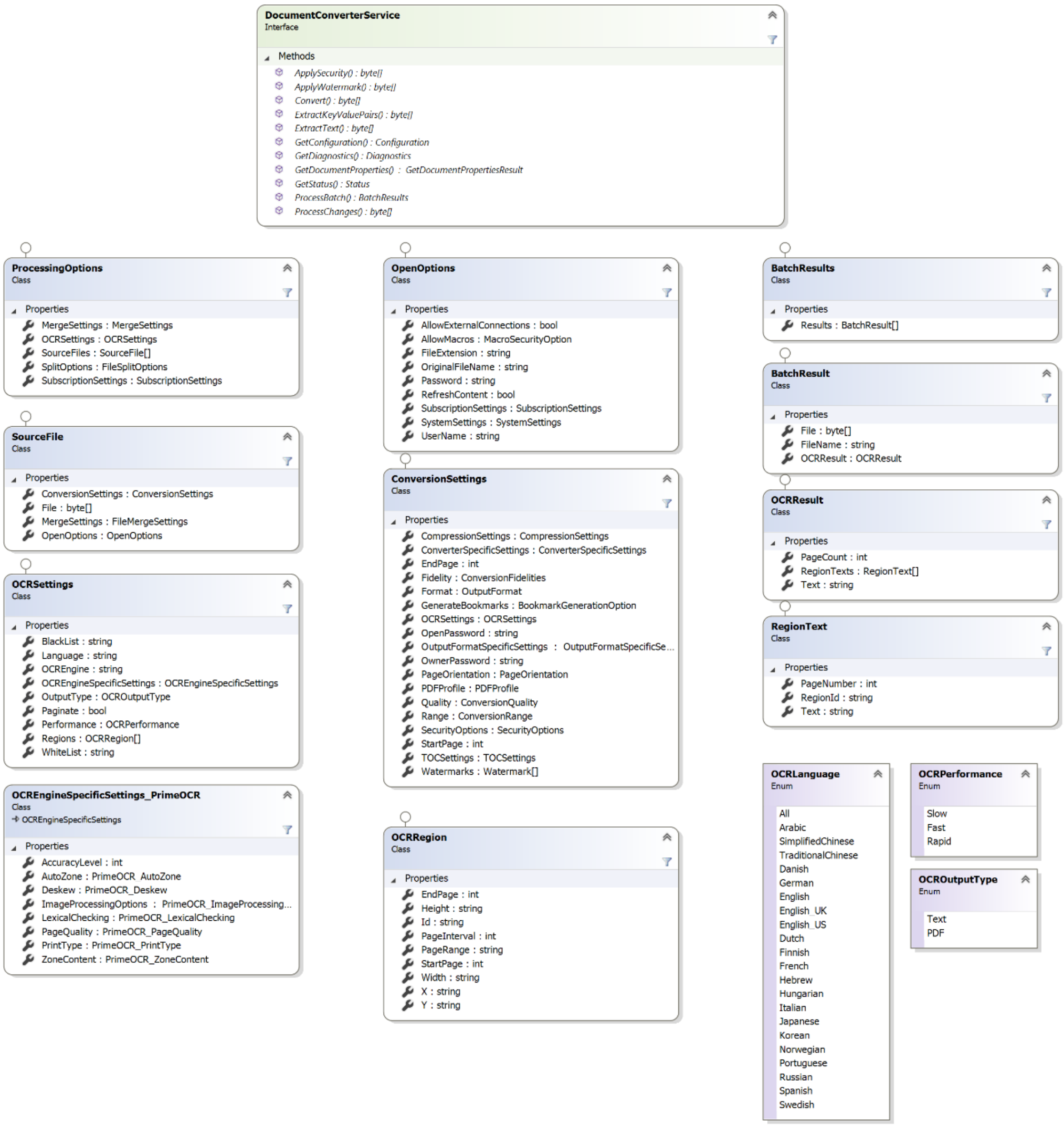
Properties
- CompressionSettings : CompressionSettings
- ConverterSpecificSettings : ConverterSpecificSettings
- EndPage : int
- Fidelity : ConversionFidelities
- Format : OutputFormat
- GenerateBookmarks : BookmarkGenerationOption
- OCRSettings : OCRSettings
- OpenPassword : string
- OutputFormatSpecificSettings : OutputFormatSpeci...
- OwnerPassword : string
- PageOrientation : PageOrientation
- PDFProfile : PDFProfile
- Quality : ConversionQuality
- Range : ConversionRange
- SecurityOptions : SecurityOptions
- StartPage : int
- TOCSettings : TOCSettings
- Watermarks : Watermark[]

OpenOptions
Class

Properties
- AllowExternalConnections : bool
- AllowMacros : MacroSecurityOption
- FileExtension : string
- OriginalFileName : string
- Password : string
- RefreshContent : bool
- SubscriptionSettings : SubscriptionSettings
- SystemSettings : SystemSettings
- UserName : string


ConversionSettings
Class
Fields
Properties
CompressionSettings : CompressionSettings
ConverterSpecificSettings : ConverterSpecificSettings
EndPage : int
Fidelity : ConversionFidelities
Format : OutputFormat
GenerateBookmarks : BookmarkGenerationOption
OCRSettings : OCRSettings
OpenPassword : string
OutputFormatSpecificSettings : OutputFormatSpecificSe...
OwnerPassword : string
PageOrientation : PageOrientation
PDFProfile : PDFProfile
Quality : ConversionQuality
Range : ConversionRange
SecurityOptions : SecurityOptions
StartPage : int
TOCSettings : TOCSettings
Watermarks : Watermark[]
OutputFormatSpecificSettings_PDF
Class
OutputFormatSpecificSettings
Properties
EmbedAllFonts : bool
FastWebView : bool
NamedDestinationProcessingMode : NamedDestinationProcessingMode
PostProcessFile : bool
SubsetFonts : bool
ViewerPreferences : PDFViewerPreferences
PDFViewerPreferences
Class
Properties
CenterWindow : bool
DisplayTitle : bool
ExtensionData : ExtensionDataObject
FitWindow : bool
FullScreen : bool
HideEmptyNavigationPane : bool
HideMenubar : bool
HideToolbar : bool
HideWindowUI : bool
NavigationPane : PDFNavigationPane
PageLayout : PDFPageLayout
PageScalingMode : PDFPageScalingMode
PDFPageScalingMode
Enum
Default
None
PDFNavigationPane
Enum
None
Bookmarks
Thumbnails
OptionalContent
Attachments
PDFPageLayout
Enum
SinglePage
OneColumn
TwoColumnLeft
TwoColumnRight
TwoPageLeft
TwoPageRight


TOCLocation
Enum
Front
Back
BookmarkGenerationOption
Enum
Disabled
Automatic
Custom
TOCSettings
Class
Properties
Bookmark : string
DocumentStartPage : DocumentStartPage
HtmlRenderingEngine : HTMLRenderingEngine
Location : TOCLocation
MinimumEntries : int
PageMargins : string
PageOrientation : PageOrientation
PaperSize : string
Properties : NameValuePair[]
Template : string
WebKitViewPortSize : string
NameValuePair
Class
Properties
Name : string
Value : string
ConversionSettings
Class
Properties
CompressionSettings : CompressionSettings
ConverterSpecificSettings : ConverterSpecificSettings
EndPage : int
Fidelity : ConversionFidelities
Format : OutputFormat
GenerateBookmarks : BookmarkGenerationOption
OCRSettings : OCRSettings
OpenPassword : string
OutputFormatSpecificSettings : OutputFormatSpeci...
OwnerPassword : string
PageOrientation : PageOrientation
PDFProfile : PDFProfile
Quality : ConversionQuality
Range : ConversionRange
SecurityOptions : SecurityOptions
StartPage : int
TOCSettings : TOCSettings
Watermarks : Watermark[]
MergeSettings
Class
Properties
BreakOnError : bool
DocumentStartPage : DocumentStartPage
OmitErrorPages : bool
OpenPassword : string
OutputFormatSpecificSettings : OutputFormatSpecificSe...
OwnerPassword : string
PDFProfile : PDFProfile
SecurityOptions : SecurityOptions
TOCSettings : TOCSettings
Watermarks : Watermark[]


DocumentConverterService
Interface
Methods
ApplySecurity() : byte[]
ApplyWatermark() : byte[]
Convert() : byte[]
ExtractKeyValuePairs() : byte[]
ExtractText() : byte[]
GetConfiguration() : Configuration
GetDiagnostics() : Diagnostics
GetDocumentProperties() : GetDocumentPropertiesResult
GetStatus() : Status
ProcessBatch() : BatchResults
ProcessChanges() : byte[]
ProcessingOptions
Class
Properties
MergeSettings : MergeSettings
OCRSettings : OCRSettings
SourceFiles : SourceFile[]
SplitOptions : FileSplitOptions
SubscriptionSettings : SubscriptionSettings
SourceFile
Class
Properties
ConversionSettings : ConversionSettings
File : byte[]
MergeSettings : FileMergeSettings
OpenOptions : OpenOptions
OCRSettings
Class
Properties
BlackList : string
Language : string
OCREngine : string
OCREngineSpecificSettings : OCREngineSpecificSettings
OutputType : OCROutputType
Paginate : bool
Performance : OCRPerformance
Regions : OCRRegion[]
WhiteList : string
OCREngineSpecificSettings_PrimeOCR
Class
OCREngineSpecificSettings
Properties
AccuracyLevel : int
AutoZone : PrimeOCR_AutoZone
Deskew : PrimeOCR_Deskew
ImageProcessingOptions : PrimeOCR_ImageProcessing...
LexicalChecking : PrimeOCR_LexicalChecking
PageQuality : PrimeOCR_PageQuality
PrintType : PrimeOCR_PrintType
ZoneContent : PrimeOCR_ZoneContent
OpenOptions
Class
Properties
AllowExternalConnections : bool
AllowMacros : MacroSecurityOption
FileExtension : string
OriginalFileName : string
Password : string
RefreshContent : bool
SubscriptionSettings : SubscriptionSettings
SystemSettings : SystemSettings
UserName : string
ConversionSettings
Class
Properties
CompressionSettings : CompressionSettings
ConverterSpecificSettings : ConverterSpecificSettings
EndPage : int
Fidelity : ConversionFidelities
Format : OutputFormat
GenerateBookmarks : BookmarkGenerationOption
OCRSettings : OCRSettings
OpenPassword : string
OutputFormatSpecificSettings : OutputFormatSpecificSe...
OwnerPassword : string
PageOrientation : PageOrientation
PDFProfile : PDFProfile
Quality : ConversionQuality
Range : ConversionRange
SecurityOptions : SecurityOptions
StartPage : int
TOCSettings : TOCSettings
Watermarks : Watermark[]
OCRRegion
Class
Properties
EndPage : int
Height : string
Id : string
PageInterval : int
PageRange : string
StartPage : int
Width : string
X : string
Y : string
BatchResults
Class
Properties
Results : BatchResult[]
BatchResult
Class
Properties
File : byte[]
FileName : string
OCRResult : OCRResult
OCRResult
Class
Properties
PageCount : int
RegionTexts : RegionText[]
Text : string
RegionText
Class
Properties
PageNumber : int
RegionId : string
Text : string
OCRLanguage
Enum
All
Arabic
SimplifiedChinese
TraditionalChinese
Danish
German
English
English_UK
English_US
Dutch
Finnish
French
Hebrew
Hungarian
Italian
Japanese
Korean
Norwegian
Portuguese
Russian
Spanish
Swedish
OCRPerformance
Enum
Slow
Fast
Rapid
OCROutputType
Enum
Text
PDF

Appendix – Expected Keys JSON

This is an example expected keys JSON

```
[
  {
   "expectedKey":"grand total",
   "synonyms":["total"]
  },
 {
   "expectedKey":"invoice number",
   "synonyms":["invoice no", "inv no"]
 }
]
```

Appendix - Instrumentation

In previous versions of PDF Converter it was not possible to determine the number of calls made.

With PDF Converter versions above 12.2, an Instrumentation system has been implemented.

This stores information about the number of calls made in a local database.

Database

The database (nutrient-usage.db) is a SQLite database. This is a single user access database mediated by the PDF Converter service itself.

The database is stored locally on the server, and is thus under the customer’s control.

It is **definitely** not recommended to examine the database using any tool while the PDF Converter service is running.

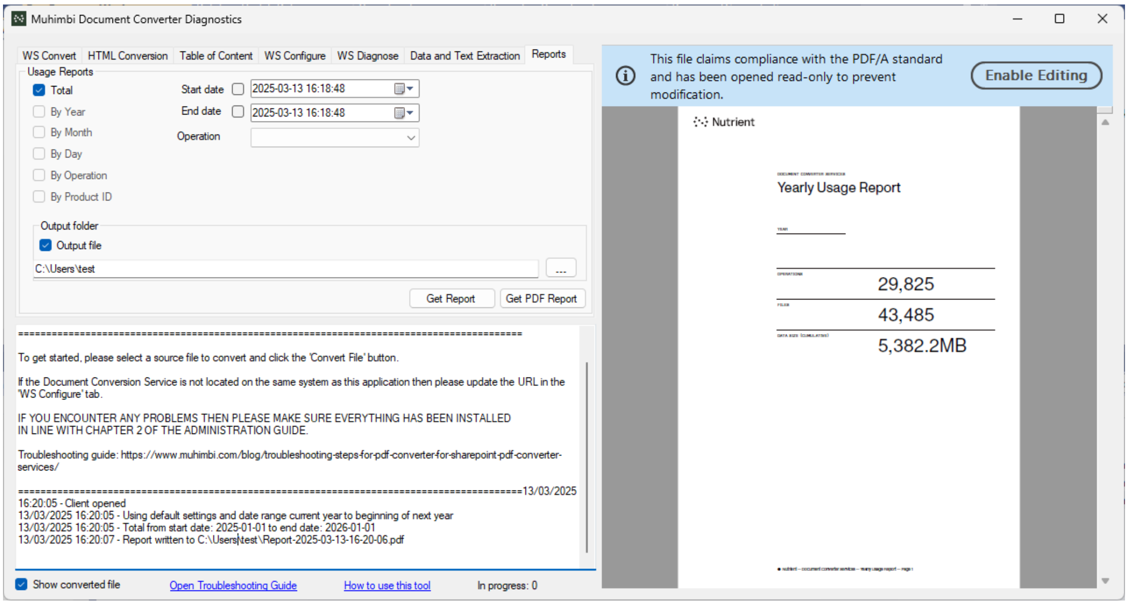
All data is accessible via reporting calls to the service.

The database does not contain the names of files processed. Each call generates a Globally Unique Identifier which is used to link all operations (OCR, Compress, Watermark etc.) in a call together.

Column	Description
ContextID	Unique ID for each request to the web server backend
ProductID	Product ID of product writing to the database
Operation	Operation number (based on OperationType for DCS, Job ID or Step ID for DAS
StartTime	Start time of the call
EndTime	End time of the call
Files	File count (generally input file count but output file count for URL based HTML conversion)
FilesSize	Cumulative file size for the files

Diagnostic Tool

The diagnostic tool included in versions above 12.2 includes a Reports tab.



Usage Reports
☑ Total
☐ By Year
☐ By Month
☐ By Day
☐ By Operation
☐ By Product ID

Start date ☐ 2025-03-13 16:18:48
End date ☐ 2025-03-13 16:18:48
Operation

Output folder
☑ Output file
C:\Users\test

```
==========================================================================================

To get started, please select a source file to convert and click the 'Convert File' button.

If the Document Conversion Service is not located on the same system as this application then please update the URL in the
'WS Configure' tab.

IF YOU ENCOUNTER ANY PROBLEMS THEN PLEASE MAKE SURE EVERYTHING HAS BEEN INSTALLED
IN LINE WITH CHAPTER 2 OF THE ADMINISTRATION GUIDE.

Troubleshooting guide: https://www.muhimbi.com/blog/troubleshooting-steps-for-pdf-converter-for-sharepoint-pdf-converter-
services/

==========================================================================================13/03/2025
16:20:05 - Client opened
13/03/2025 16:20:05 - Using default settings and date range current year to beginning of next year
13/03/2025 16:20:05 - Total from start date: 2025-01-01 to end date: 2026-01-01
13/03/2025 16:20:07 - Report written to C:\Users\test\Report-2025-03-13-16-20-06.pdf
```

☑ Show converted file
In progress: 0

This file claims compliance with the PDF/A standard and has been opened read-only to prevent modification.

Yearly Usage Report

OPERATIONS	29,825
FILES	43,485
DATA SIZE (CUMULATIVE)	5,382.2MB

CSV Reports

As of 12.2, the Diagnostic Tool can generate CSV files.
The CSV data can be grouped by:

- Year
- Month
- Day
- Operation
- Product (only DCS in this case)

The date range can be specified and for a specific operation.

PDF Reports

As of 12.2, the Diagnostic Tool will choose the PDF report format based on the options selected.

After setting the Output Folder, the default report is total usage for the current year (starting 1st of January for the current year).

By setting the Start Date check box and the associated date, it will generate usage for the selected start year (starting 1st of January for that year)

If you set the start and end date check boxes, the report shows the calls, operations, files and cumulative files size between the start and end date.