PDF Converter for SharePoint - User Guide

Muhimbi Ltd

Version 10.3

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	Stephen Carter	26/04/2023	Updated for version 10.3

Purpose and audience of document

This document contains instructions about how to use the PDF Converter for SharePoint on a day by day basis as well as how to create workflows that utilise the PDF Converter.

The intended audience is any SharePoint user with the need to convert documents to PDF format, apply watermarks or PDF Security as well as power users who create workflows using Nintex and SharePoint Designer or who develop custom software.

Disclaimer

Contents

1 Introduction

This document describes how to use the PDF Converter for SharePoint on a day-by-day basis. The intended audience is any SharePoint user with the need to convert documents to PDF format, merge files, extract forms data, OCR images, apply watermarks or PDF security as well as power users who create workflows using Nintex and SharePoint Designer or who develop custom software.

It is assumed that the audience has some familiarity with SharePoint and have been given the proper privileges to write documents to a SharePoint site.

From time to time, you will encounter screenshots of different SharePoint versions in this document. Unless mentioned otherwise usage of the PDF Converter is the same for all supported versions of SharePoint.

Please note that the functionality described in this document applies to the *on-premises* (SP2007-2019 & SE) versions of the PDF Converter for SharePoint. The functionality provided by the *Online* (App Store) based version of the software is largely identical but not always exactly the same. The Online / App Store version can be installed on-premises as well. For details see this blog post.

For more details about this product please see:

1. Product Information: https://www.muhimbi.com/Products/PDF-Converter-for-SharePoint/
2. Product Overview: https://www.muhimbi.com/guides/pdf-converter/
3. Knowledge Base / Frequently Asked Questions: https://www.muhimbi.com/
4. Release Notes: https://www.muhimbi.com/guides/pdf-converter/sharepoint/release-notes/
5. Installation & Administration Guide https://www.muhimbi.com/guides/pdf-converter/sharepoint/
6. Developer Guide: https://www.muhimbi.com/guides/pdf-converter-services/
7. PDF Converter related content on the Muhimbi Blog: https://www.muhimbi.com/blog/

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

1.1 Available SharePoint Features

SharePoint Features are plug-ins that allow new functionality such as PDF Conversion to be added to a SharePoint environment. Individual Features can be enabled or disabled by Site or Farm Administrators.

The following SharePoint Features come with the PDF Converter for SharePoint. For further details contact your SharePoint administrator or see section 2.2.3 in the Administration Guide.

Feature Name	Scope	Enabled
Muhimbi.PDFConverter.Farm Enable the Workflow Activities and the Central Administration Configuration screen.	Farm	V
Muhimbi.PDFConverter Add 'Convert to PDF' to context menus / ribbons for all site collections in the Web Application.	Web App	V
Muhimbi.PDFConverter.Nintex.WebApp[1] Add Nintex Workflow actions on systems that have this third party product installed.	Web App	-
Muhimbi.PDFConverter.Watermarker.UI.WebApp Enable the user interface for the automatic watermarking and security facilities on all site collections in the Web Application.	Web App	-
Muhimbi.PDFConverter.Watermarker.Processor.WebApp Enable the automatic watermarking and security processing logic on all site collections in the Web Application	Web App	-
Muhimbi.PDFConverter.Convert.Site Add 'Convert to PDF' to context menus / ribbons for a single site collection.	Site Coll.	-
Muhimbi.PDFConverter.ConvertAndDownload.Site Add the 'Download as PDF' option to the file's context menu.	Site Coll.	-
Muhimbi.PDFConverter.ConvertWebPage.Site Add the 'Convert Page to PDF' option to the user's Personal Actions menu.	Site Coll.	-
Muhimbi.PDFConverter.Watermarker.UI.Site Add the user interface for the automatic watermarking and security facilities to the site collection.	Site Coll.	-
Muhimbi.PDFConverter.API.WebApp Add K2 Prerequisites, Feature is only present on SharePoint 2007 and not needed by newer SharePoint versions.	Web App	-
Muhimbi.PDFConverter.SP2013_WebFeature Enable workflow actions for the optional Workflow Manager part of SharePoint 2013 and later versions.	Web	'-

[1] Due to an issue with Nintex Workflow, please carry out an IISRESET after deactivating this feature.

The *Muhimbi.PDFConverter* and *Muhimbi.PDFConverter.Convert.Site* features are identical with the exception of the scope. The *WebApplication* scoped feature is enabled by default and adds PDF Converter options to all site collections in the web application. If you prefer to enable PDF Conversion at the Site Collection level, then make sure the *WebApplication* scoped feature is disabled. Do not leave both features enabled at the same time to prevent duplicate menu options.

The same mechanism is used by the *Muhimbi.PDFConverter.Watermarker.UI.WebApp* and *Muhimbi.PDFConverter.Watermarker.UI.Site* Features. Use the 'WebApp' version of the Feature to enable it on all site collections. If you just wish to use it on a select number of site collections, then disable it at the Web Application level and enable it at the Site Collection level.

Please note that certain SharePoint Features have internal dependencies. For example, in order to enable the *Automatic PDF Processing User Interface* Feature at either the Site Collection or Web Application level, the *Automatic PDF Processor* Feature must be enabled first, which must be done at the Web Application level.

1.2 Supported document formats

The PDF Converter for SharePoint supports the most common file formats including MS-Word, Excel, PowerPoint, InfoPath, MSG (email), Visio and Microsoft Publisher. Legacy file formats starting with Office 95 are supported as well as the latest formats used by Office 2019 / 365. Non MS-Office related file types such as HTML, AutoCAD and common image formats are supported as well.

The PDF Converter also supports output in non-PDF file formats. For details see Chapter 9 *Cross-Converting between document types*.

	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, xml, csv, dif, ods, ots	xltx, xltm, xlt, txt (tab delimited), prn, slk, xlam, xla
PowerPoint	ppt, pptx, pptm, xml, odp, otp, pps, ppsx, ppsm	potx, potm, pot, thmx, ppam, ppa
InfoPath	xml, infopathxml	
Publisher	pub	
Email	eml, msg	
Visio & Vector formats	vsd,vdx,vdw,svg,svgz	
HTML & Web pages	html, htm, mht and any url that returns HTML such as .aspx or .jsp.	
Image formats	gif, png, jpg, bmp, tif, tiff	
AutoCAD formats[2]	dwg, dxf	
Postscript	ps, eps	
PDF	pdf, fdf, xfdf, xml	

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

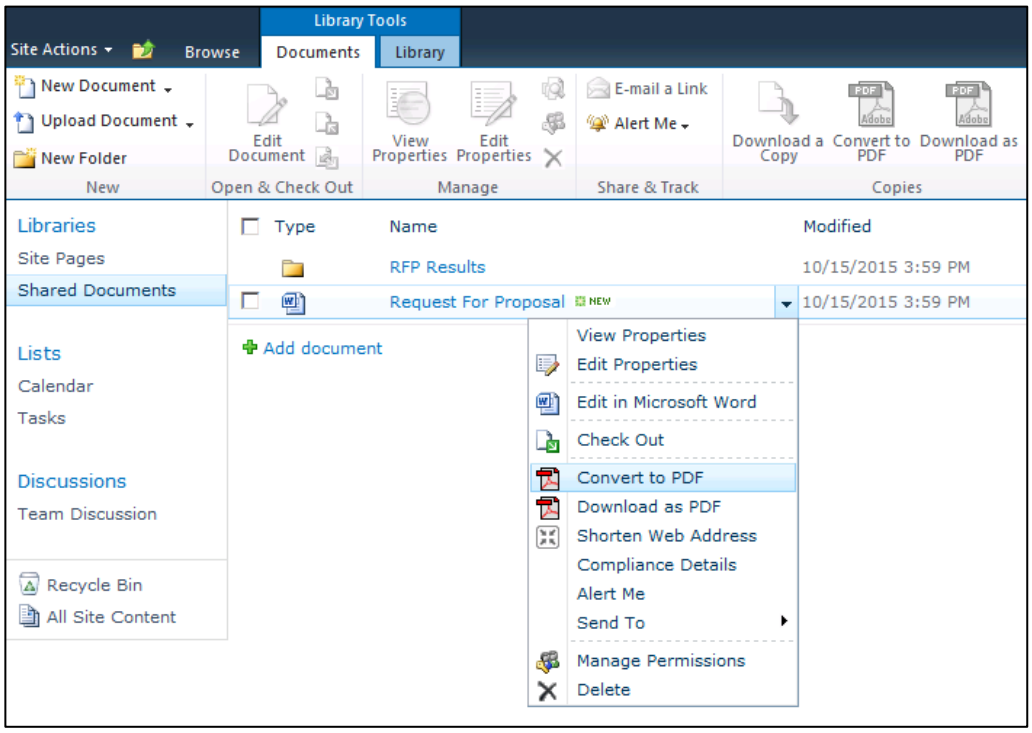
2 Converting documents using the SharePoint U.I.

The PDF Converter for SharePoint is a friendly and intuitive application. This chapter describes the available functionality in more detail.

2.1 Converting a single file

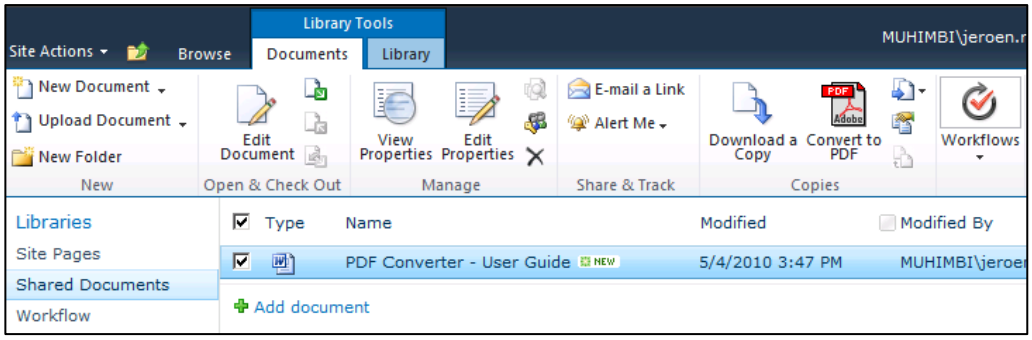
To convert a single file to PDF format, follow the steps outlined below:

1. Navigate to the Document Library and folder the file is located in.
2. Open the context menu (See screenshot below) and select 'Convert to PDF'.



3. Review the settings and click the '*Convert*' button to start the conversion process.
4. Verify the file has been converted without errors.
5. Click the '*Destination Library*' to go to the directory that holds the converted PDF file.

When using the PDF Converter in combination with SharePoint 2010 (or later) then a selection of files and folders can be converted / merged by selecting them and choosing *Convert to PDF* in the Ribbon.



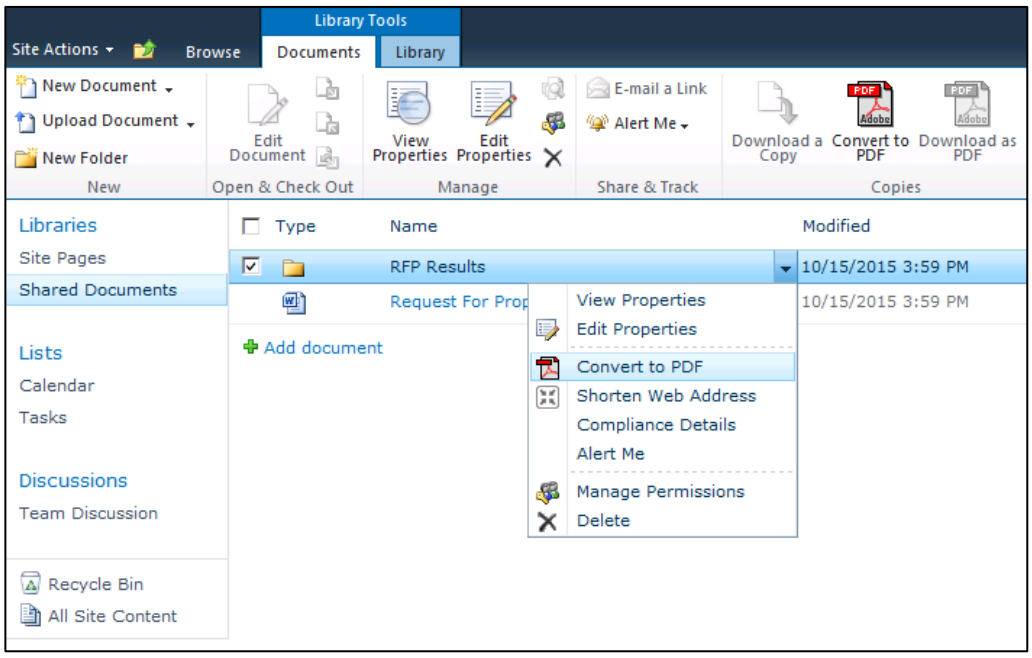
If the *Muhimbi.PDFConverter.ConvertAndDownload.Site* Feature is enabled on a site collection then a *Download as PDF* option is added to both the SharePoint ribbon as well as the file Context Menu.

Download as PDF works similar to *Convert to PDF* except that it opens / downloads the PDF file on the local machine rather than saving it to SharePoint.

2.2 Converting an entire folder

Converting an entire folder to PDF format works similar to converting a single file:

1. Navigate to the Document Library that contains the folder to convert.
2. Open the context menu (See screenshot below) and select 'Convert to PDF'.



3. Specify if you want to convert subfolders as well. Note that this may take a long time depending on how many files are in the folder hierarchy.
4. Review the other settings and click the '*Convert*' button to start the conversion process.
5. Verify the files have been converted without errors.
6. Click the '*Destination Library*' to go to the directory that holds the converted PDF files.

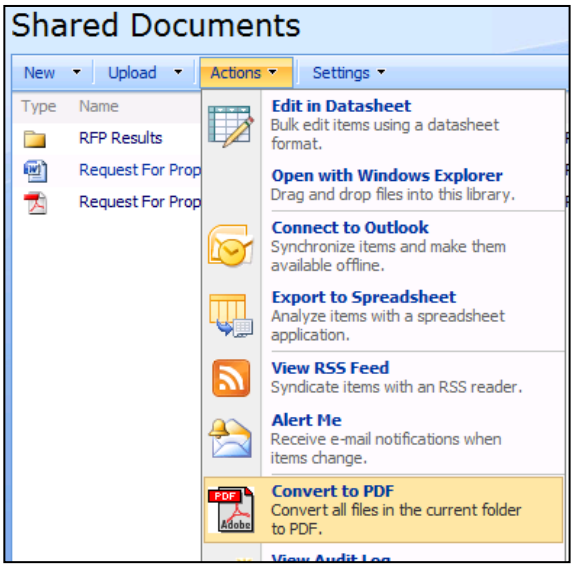
Although the *Convert to PDF* function works fine with folders, the *Download as PDF* feature cannot be used on folders.

2.3 Converting an entire document library

Although it is possible to convert an entire document library in one go, it may take a long time to complete, or even time out, if it contains many files. You will automatically receive a warning if there are many files in the library.

Convert the entire document library as follows:

1. Navigate to the root folder of the document library to convert.
2. Select '*Convert to PDF*' from the Actions menu (SP2007 only, see screenshot below).



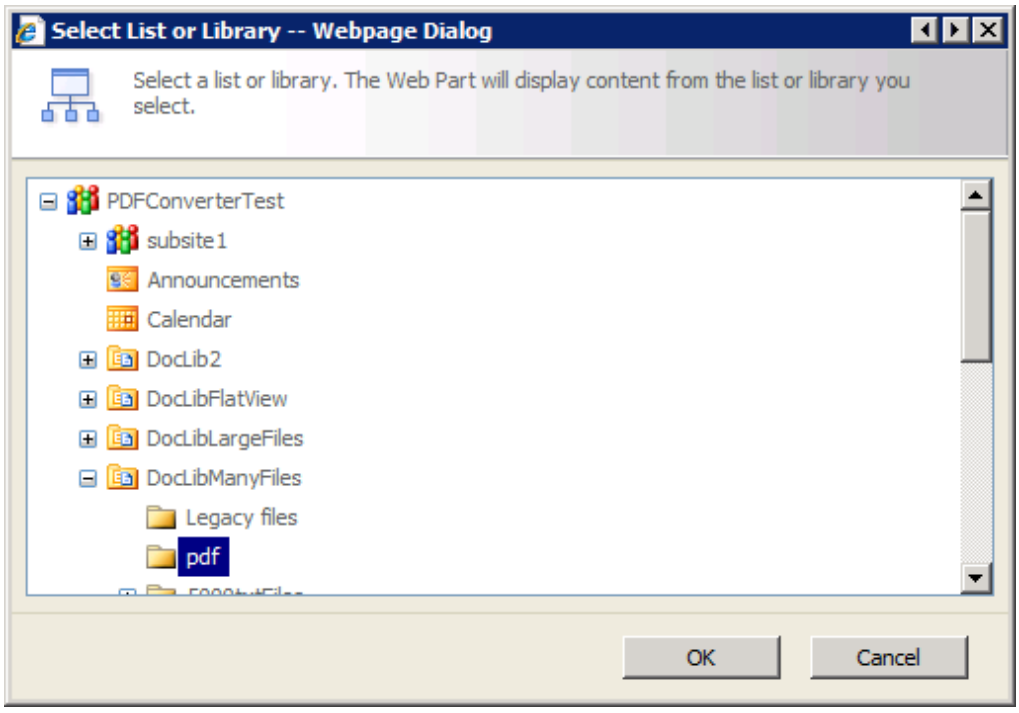
3. Specify if you want to convert subfolders as well.
4. Review the other settings and click the '*Convert*' button to start the conversion process.
5. Verify the files have been converted without errors.
6. Click the '*Destination Library*' to go to the directory that holds the converted PDF files.

Please note that this menu option is not available in SharePoint 2010 or later. Select all files and use the 'Convert to PDF' option in the Ribbon instead.

2.4 Writing the converted file to a different location

By default, PDF files are written to the same folder the source files are located in. However, it is possible to specify a different folder, different document library, different site or even a different site collection to write the converted PDF files to.

To specify a different folder, open the *PDF Conversion* screen and specify the destination folder. When the site is hosted on SharePoint server, rather than the free Foundation server, then a user-friendly *Browse* button is available which allows folders or sites to be selected within the current site collection.



Browse for a destination folder (Commercial versions of SharePoint only)

If the *Browse* button is not available, or the destination folder is located in a different site collection, then the destination path will need to be entered manually. The format is as follows:

1. A folder in the current site collection: *Document Library Name/Folder path*. E.g.,
   Shared Documents/PDF Files
2. A folder in the different site collection: *Absolute path to Site Collection/ Document Library Name/Folder path*[3]. E.g.

   /sites/Press Office/Shared Documents/PDF Files

For more details about specifying paths see *Appendix - Specifying path and file names*.

[3] Please do not include 'http://' and the domain name in the absolute path. Absolute paths always start with '/'. For details see *Appendix* - .

If the files are always written to the same directory, then it may be a good idea to click *Remember this path* to automatically default to whatever custom path was last entered. Note that these settings are remembered per user and stored at the Site collection level.

Clearing the *Remember this path* checkbox will automatically forget the settings during future conversions.

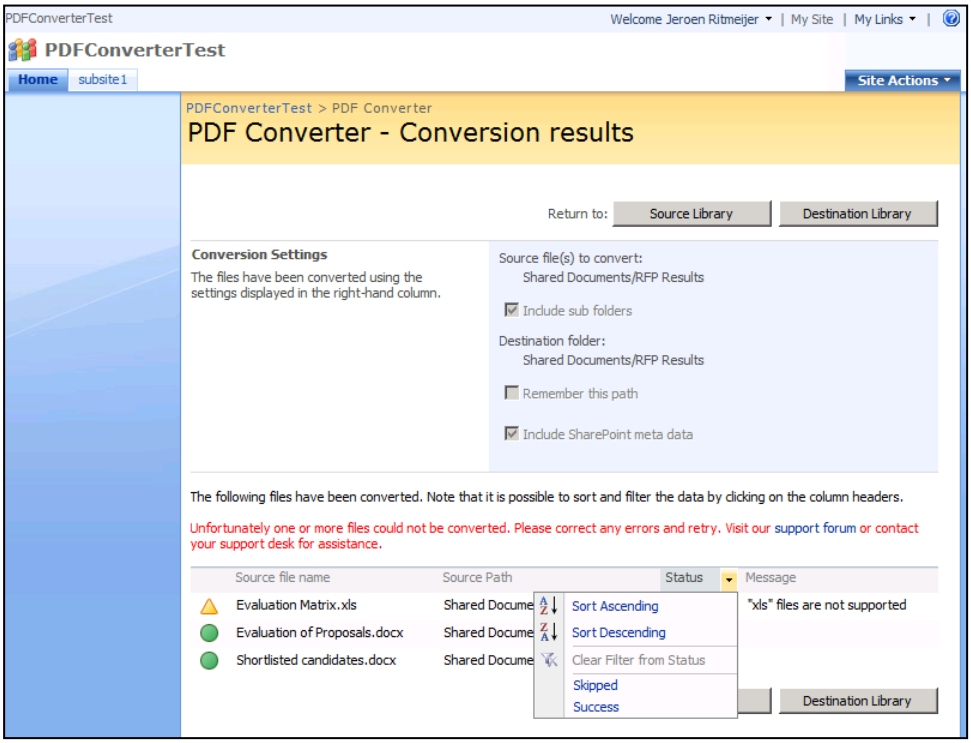
Please note that *Document Sets* can be used as a source as well as a destination for PDF Conversion. They work exactly the same as folders.

2.5 Working with the 'Conversion Results' screen

The *Conversion Results* screen contains functionality to simplify dealing with a large number of converted documents.

The following features are available:

1. **Filter by column:** Click on the header of the column to filter on and select the value to filter for. For example, select 'Error' in the 'Status' column to generate an overview of all files that caused an error during the conversion process.
2. **Sort by column:** Click on the header of a column to sort its contents either in Ascending or Descending order. For example, click on the 'Source Path' column to group all documents by folder.



3. **Jump to destination folder:** Click the '*Destination Library*' button to navigate to the folder the PDF file was written to. This is particularly useful when a different destination folder has been specified during the conversion process.
4. **Page through results:** A pager is automatically displayed when more than 100 files have been converted.

3 Converting documents via a SharePoint workflow

The PDF Converter for SharePoint comes with a number of *Workflow Activities* including one that can be used to convert documents as part of a SharePoint Designer (SPD) workflow. What follows is an example of how to create a simple workflow in SharePoint Designer, which converts a file to PDF format as soon as it has been approved.

The legacy SharePoint 2007 / 2010 workflow engine is fully supported, as is the optional *Workflow Manager* that comes with SharePoint 2013 and later versions. For details see this post.

Before you start, make sure the PDF Converter for SharePoint has been installed and you have access to a site collection with the appropriate rights to create workflows. Some basic knowledge about creating and configuring document libraries is assumed.

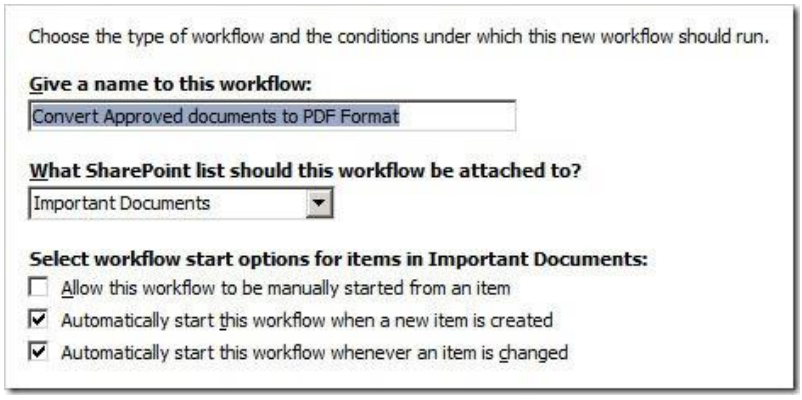
Another, even more basic, example can be found in our Knowledge Base.

3.1 Create and configure the Document Library

1. Create a document library named *Important Documents*.
2. Once created, enable *Requires content approval* on the library's *Settings* > *Document Library Settings* > *Versioning Settings* screen.
3. In the document library, create two folders: *Confidential Proposals* and *Approved PDF Files*.

3.2 Create and configure the workflow

1. Start SharePoint Designer and open the site collection that contains the *Important Documents* library.
2. From the File menu select *New* > *Workflow*.
3. On the first screen of the Workflow wizard, specify the following settings:
   a. Name the workflow *Convert Approved documents to PDF Format*.
   b. Select the *Important Documents* list.



   c. When creating a SharePoint Designer Workflow in SharePoint 2013 or later then please select the appropriate *Platform Type.*

d. Select the 2nd and 3rd checkboxes to make sure the workflow is triggered whenever a document is created or (its status) is updated.

e. Click the *Next* button to proceed.

4. We are now ready to create the workflow. From the 'Conditions' menu, select 'Compare any data source'. This inserts the 'If value equals value' condition.
5. Click on the first *value* followed by the *display data binding (fx)* button.
6. Select *Current Item* as the Source and select *Approval Status* in the field. Click the *OK* button to continue.
7. Click on the second *value* and select *Approved* from the list.

To avoid that the same workflow is executed on the converted PDF file after the conversion has taken place, we need to specifically exclude pdf files as follows:

1. Add another *Compare any data source* condition.
2. Click on the first *value* followed by the *display data binding (fx)* button.
3. Select *Current Item* as the Source and select *File Type* in the Field. Click the *OK* button to continue.
4. Click *equals* and change it to *not equals*.
5. Click on the second *value* and enter *pdf*. (Do not prefix the extension with a period '.').

With the Conditions in place, we can now add the Actions, which is where the magic happens.

1. From the *Actions* menu, select *Convert to PDF*. It may be hidden behind the *More Actions* option.
2. The following action is inserted:

   Convert this document to this url using the same file name and include / exclude meta data. Store the converted item details in List ID: Variable: List ID, Item ID: Variable: List Item ID.

   Let's examine what the various options mean:

   a. this document: Specify which document to convert. Select the option and make sure *Current Item* is selected.

   b. this url: Specify the location the converted file will be written to. The following options are available:

      i. Leave it empty: When no value is specified then the converted document is written to the same folder as the source file.

      ii. Site Relative URL[4]: By specifying a URL relative to the current site, e.g., *subsite/shared documents/PDF Files*, any folder location in the

[4] For more details see 2.4 *Writing the converted file to a different location*.

current site collection can be targeted. ***This is the option we want to use so enter 'Important Documents/Approved PDF Files'.***

iii. Web Application relative URL[5]: Using a URL that is relative to the entire web application, e.g., */sites/Press Office/Public Documents/To Distribute*, any folder location in any site collection can be targeted.

For more details see *Appendix - Specifying path and file names*.

c. the same file name: The name of the converted file can be specified here. In our case we'll leave it empty to make sure we use the same name as the original document.

d. include / exclude meta data: In case of sensitive documents, we may want to strip any custom SharePoint columns from the file. Assuming that our document library contains a column *Yearly sales forecast*, we want to select *Exclude*.

e. Variable: List ID: A new workflow variable named *List ID* is automatically created. After the file has been converted, this variable will contain the ID of the list the converted file was saved to. This can later be fed into another action in order to manipulate this file further.

f. Variable: List Item ID: A new workflow variable named 'List Item ID' is automatically created. After the file has been converted, this variable will contain the ID of the item the converted file was saved to. This can later be fed into another action in order to manipulate this file further.

3. Insert a new action named *Log to History List* and enter *File converted*.

4. Insert a new action named *Update List Item* and click this list. We are now going to automatically mark the converted PDF file as approved.

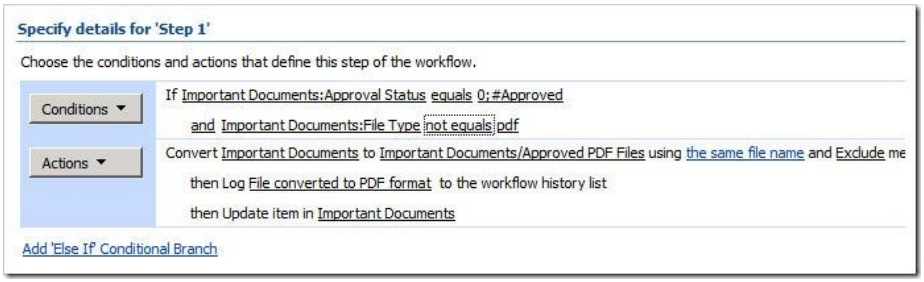
   a. From the *List* dropdown select *Important Documents*.

   b. Click the *Add* button and set the field to *Approval Status* and the value to *Approved*. Click *OK*. *Please note that behaviour has changed starting with SharePoint 2010. It is no longer possible to set the value of the 'Approved' field.*

   c. In the *Find the List Item* area set the field to *Important Documents:ID*.

   d. Click the *fx* button next to *Value*, specify *Workflow Data* as the Source and set the field to *Variable: List Item ID*. Click *OK*.

   e. Click *OK* again to return to the Workflow Designer. It should look like the image below.



[5] Please do not include 'http://' and the domain name in the absolute path. Absolute paths always start with '/'. For details see *Appendix - .*

Click the *Finish* button to activate the workflow.

3.3 Testing the workflow

The workflow can be tested as follows: create or upload an MS-Word file to the *Confidential Proposals* folder. From the context menu select *Approve / Reject* and approve the file.

This will automatically start the workflow and after a few seconds the Workflow status should change to *Completed* as shown in the screenshot below.

Type	Name	Modified	Modified By	Approval Status	Convert Approved documents to F
	Corporate Restructuring ! NEW	16/04/2009 11:48	Jeroen Ritmeijer	Approved	Completed

Once the workflow has completed, you will find the PDF version of the document in the *Approved PDF Files* folder.

Type	Name	Modified	Modified By	Approval Status	Convert Approved documents to F
	Corporate Restructuring ! NEW	16/04/2009 12:02	Jeroen Ritmeijer	Approved	Completed

If an error occurs during the execution of the workflow, then have a look at the following:

1. Check the messages on the workflow status screen.
2. Look in the Windows Event log.
3. Look in the SharePoint trace log.

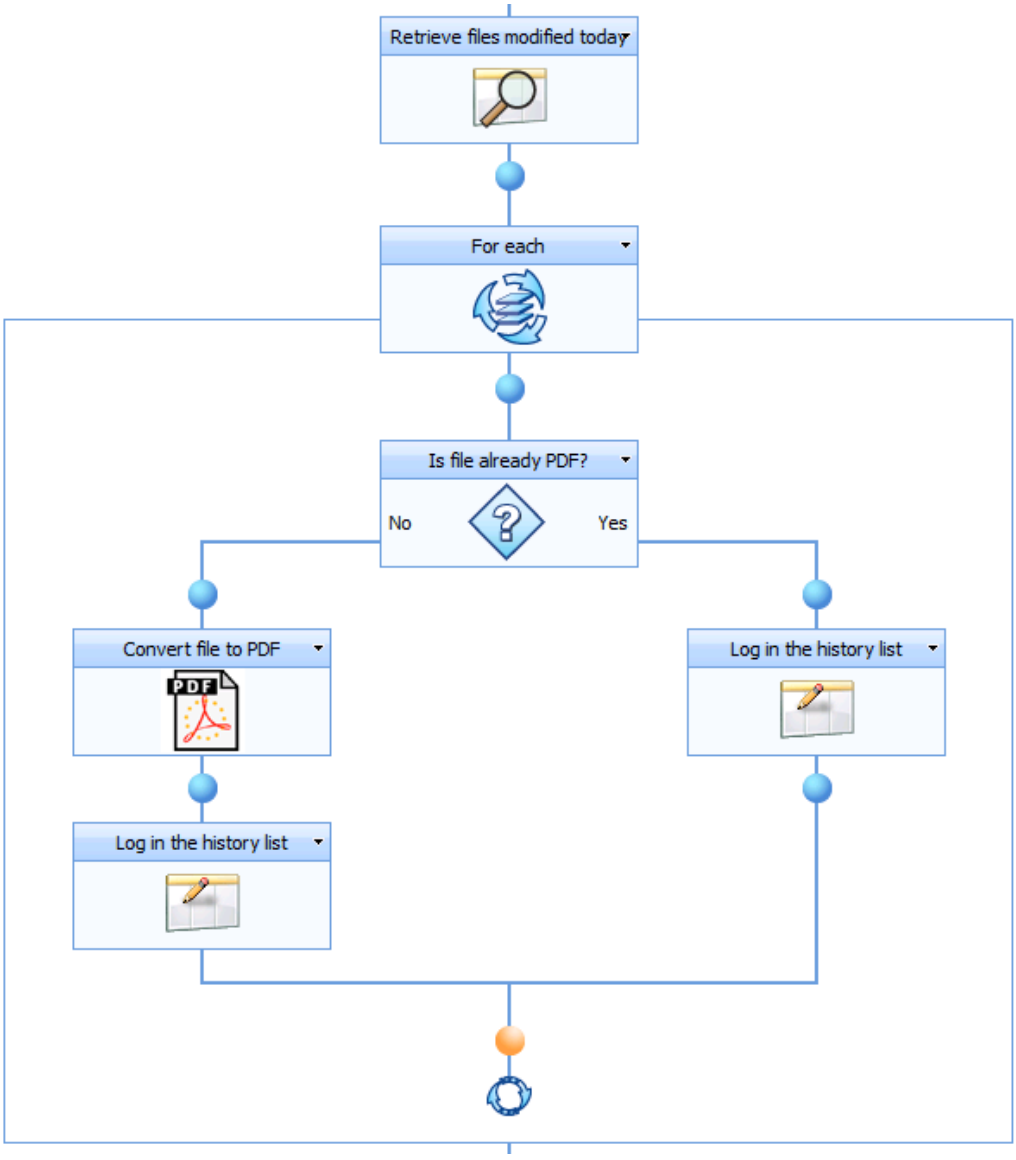
4 Converting Documents using Nintex Workflow

The Muhimbi PDF Converter for SharePoint comes with comprehensive support for all Nintex Workflow versions, currently NW2007 – NW2019.

Let's illustrate the power of these combined tools using an example that is very difficult to achieve in SharePoint designer: *Act on a set of list items*. In this example a workflow retrieves all files modified during the current day and converts them to PDF. Ideally you would schedule this workflow to run off-peak to batch process all new and modified files.

Please follow the instructions below or download the complete workflow in NWF format from the Muhimbi Blog at the following address:

https://www.muhimbi.com/blog/how-to-use-nintex-workflows-to-convert-to-pdf/



The finished workflow

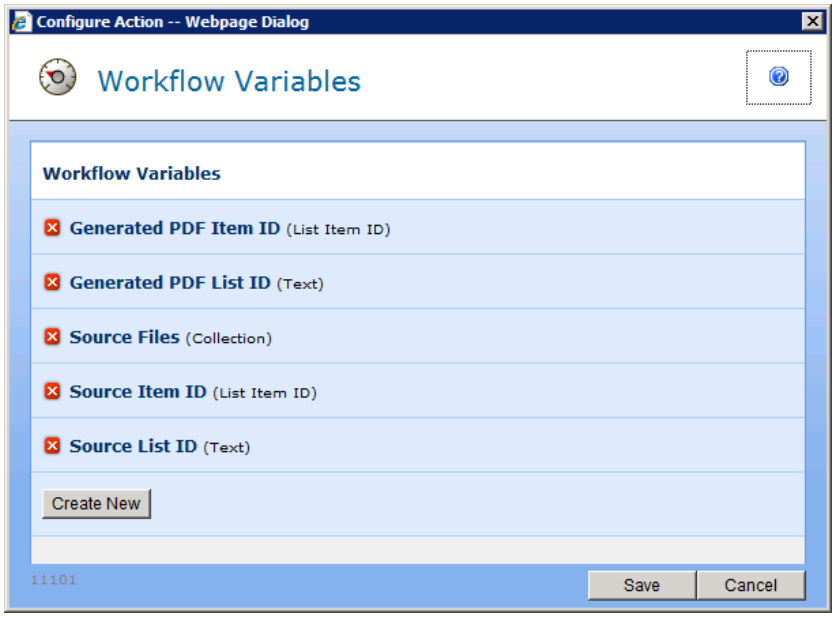
Prerequisites

Before we start building the workflow, please make sure all prerequisites are in place. It is also assumed that the reader has some knowledge of building Workflows using Nintex Workflow.

1. Make sure the PDF Converter for SharePoint version 4.1 (or newer) is installed in line with chapter 2 of the Administration Guide.
2. Naturally Nintex Workflow will need to be installed as well.
3. Make sure the *Muhimbi.PDFConverter.Nintex.WebApp* SharePoint Feature is activated on the relevant Web Application using SharePoint Central Administration.
4. The user (you) will need to have the appropriate privileges to create workflows.

Creating a new workflow

To get started, create a new workflow, and choose the *blank* template. Make sure the workflow doesn't start automatically and add the workflow variables listed in the following screenshot.



Please make sure that the appropriate data types are assigned (They are listed between round brackets behind each variable name). The names are largely self-describing, but some additional information is provided below:

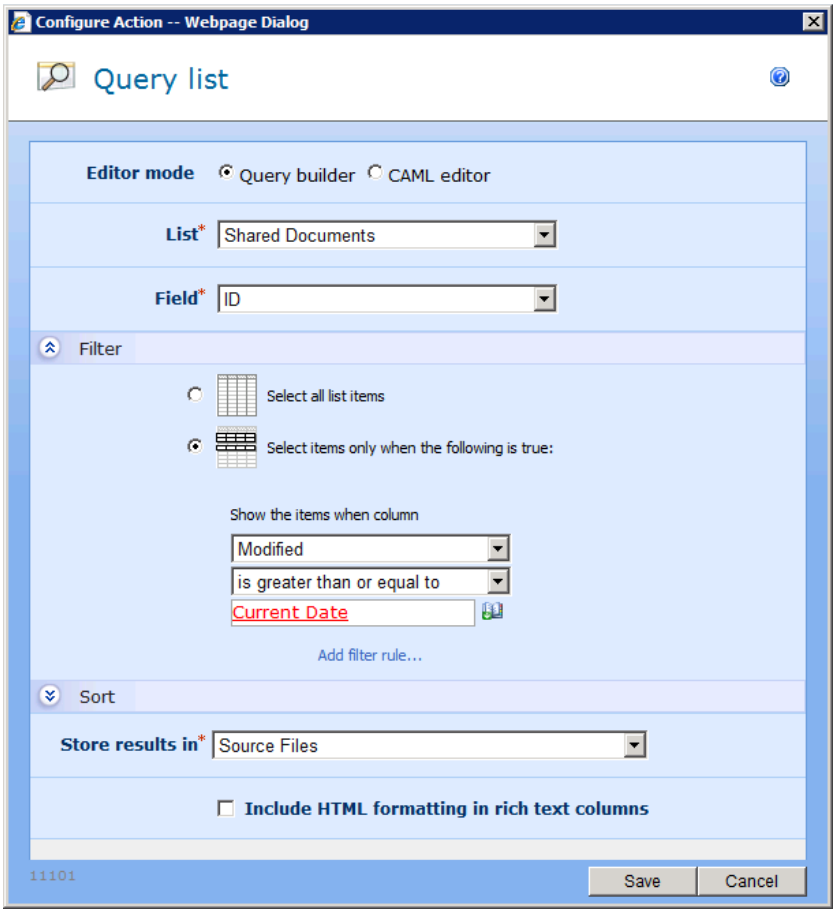
- **Source Item ID:** By default, the item that triggered the workflow is converted to PDF format. However, as we are iterating over multiple items, we need to specify the ID of the item to convert in this variable. In SharePoint 2010 and later select *Integer* as the Type, not *List Item ID.*
- **Source List ID:** The PDF Converter assumes the item that is being converted is located in the same list the workflow is attached to. However, if this is not the case then the ID (a GUID) of the list will need to be

specified as well. In this example everything is located in the same list, so this variable is not actually used.

- **Source Files:** As we are potentially converting multiple files, we need to define a variable of type *Collection* to hold the list of files we'll be iterating over.
- **Generated PDF Item ID:** Once a file has been converted to PDF, you may want to carry out additional actions on this new file, for example checking it in. Once converted, the ID of the PDF is automatically stored in this variable. In SharePoint 2010 and later select *Integer* as the Type, not *List Item ID*.
- **Generated PDF List ID:** As the PDF Converter allows files to be written to different document libraries, and even completely different Site Collections, you may want to know the ID of the destination list.

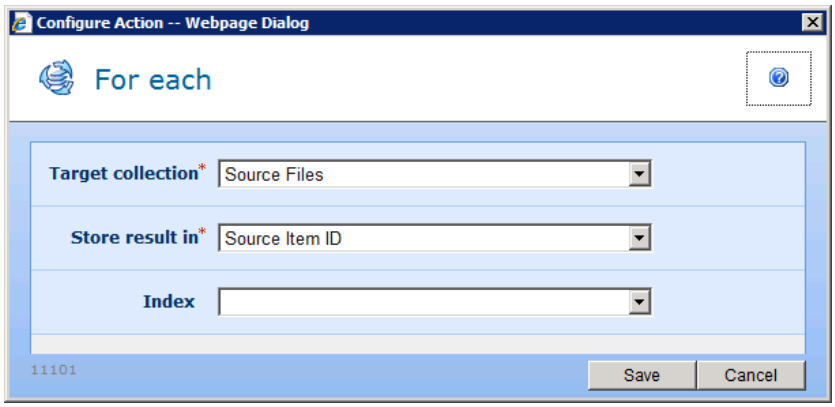
Adding the workflow actions

We are now ready to add the actions to the workflow. Begin by adding a *Query List* action, which will allow us to retrieve all files modified today and store the results in the *Source Files* collection.



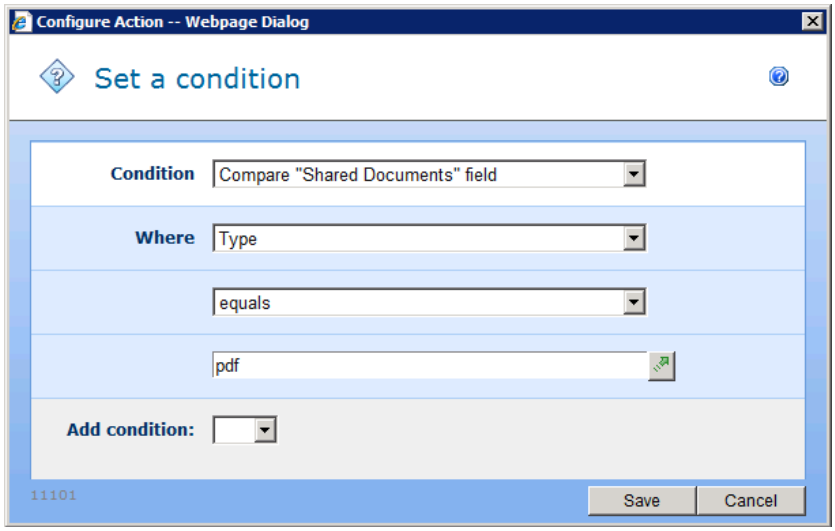
Please fill out the settings for this action as per the screenshot listed above. You may want to add an additional filter rule to check that *Content Type* is not equal to *Folder* or *Document Set*.

Continue by adding the *For Each* action to the workflow. Specify the name of the collection to iterate over and the name of the variable to store the Item's ID in.

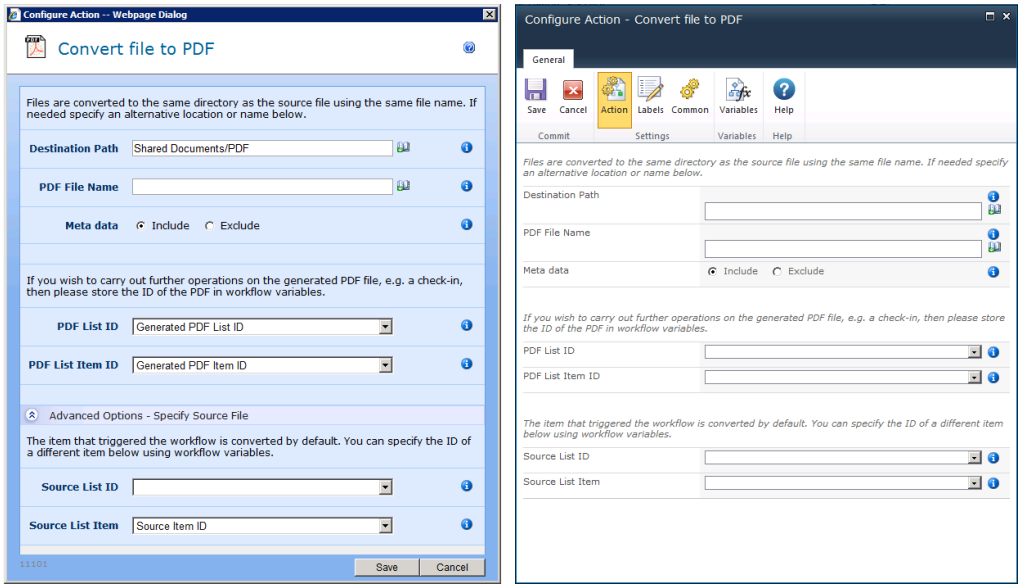


The next set of actions will all need to be added **inside** the *For Each* action, to make sure they are executed separately for each file in the list.

We want to make sure that we only invoke the PDF Converter for files that are not already in PDF format, so add a condition and check that the file type equals 'pdf' as per the following screenshot.



Add the *Convert file to PDF* action listed under the *Muhimbi PDF* section to the *No* branch of the condition (type is not pdf). Fill it out as per the left most image in the following screenshot.



The PDF Converter integrates with all Nintex Workflow versions.

You may want to leave the *Destination Path* empty, which will write the PDF File to the same location as the source file. For more information about the *Destination Path* or any of the other fields, hover the mouse over the small information icons.

The workflow is now done. You may want to add some tracking information using the *Log In the History List* action. In our example we use the following 2:

- Last action in the *No* branch:
  `List ID: {WorkflowVariable:Generated PDF List ID} - List Item ID: {WorkflowVariable:Generated PDF Item ID}`

- Last action in the *Yes* branch:
  `Already in PDF Format: {WorkflowVariable:Source Item ID}`

Running the workflow

Finalise the workflow by saving and publishing it, after which it is ready to be executed.

You can either run the workflow manually or schedule it to run at a time of your choice.

An example showing how to use the output of one PDF Workflow Action as the input of the next Action can be found on our blog.

5 Converting Documents using a K2 workflow

As of version 7.3, the *Muhimbi PDF Converter for SharePoint* provides native support for the K2 workflow engine using K2's SmartObjects technology. This chapter describes how to create a basic workflow to convert documents to PDF using *K2 Studio* (See 5.2) as well as *K2 Designer* (See 5.3).

A note on licensing the Muhimbi PDF Converter for SharePoint when used in combination with K2 blackpearl. Muhimbi's licensing model is very simple, if a server runs Muhimbi Software in any way shape or form, then it requires a license. Even though the PDF Conversion engine may be installed on a non-K2 server, all K2 Servers run our SmartObjects and therefore require a license. For details, in plain English, about how Muhimbi's software is licensed, see this Knowledge Base Article.

5.1 Prerequisites

Before creating the workflow, please make sure Muhimbi's K2 Integration facilities have been deployed as described in the Administration Guide, *Appendix – Deploying K2 Integration facilities*. Basic knowledge of creating workflows in K2 Studio / Designer, and having the privileges, is assumed.

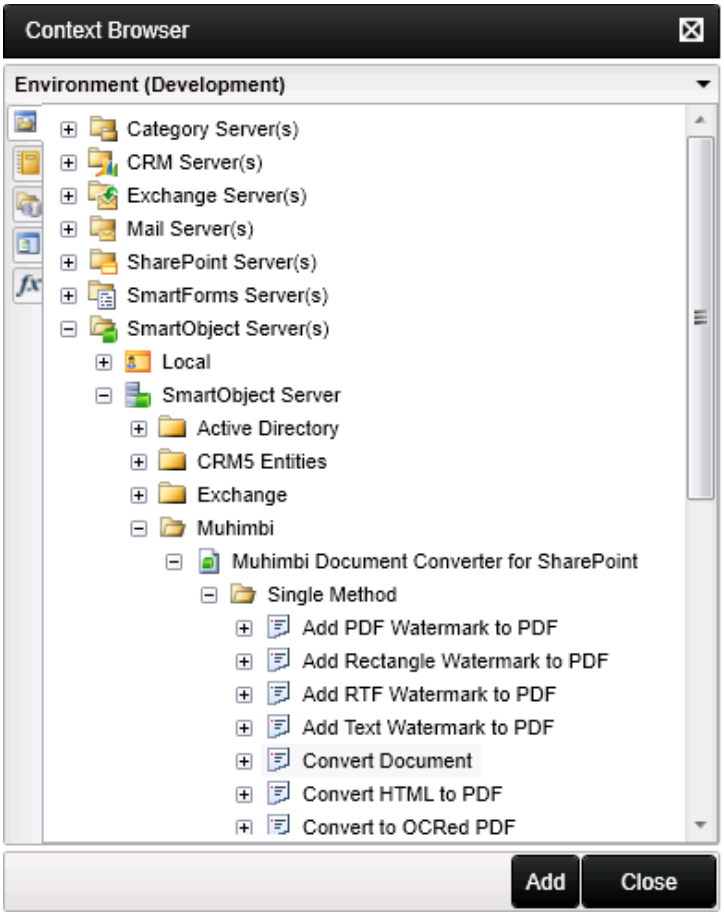
This tutorial was written for SharePoint 2010. Muhimbi's PDF Converter for SharePoint integrates equally well with other SharePoint versions, but the actual steps for creating K2 workflows differ in each SharePoint version, particularly in SharePoint 2013 and later. Please refer to K2's tutorials and documentation for your particular environment.

5.2 Creating the workflow using K2 Studio

In this tutorial we will use K2 Studio to create a basic workflow to convert files to PDF and associate the workflow with a Document Library named *Tutorial*.

1. Create a new Document Library named *Tutorial.*
2. Launch K2 Studio and create a new project of type *K2 Process* using template *Blank Process*. Save it in a location of your choice.
3. In the workflow editor select *Process Wizards* in the left-hand pane and drag the *SharePoint Events Process* onto the design surface.
4. Click *Next* in the wizard and then, in the *Action* screen, select the *Events for List and Library Items* option.
5. On the *Connection Settings* screen specify the URL of your site collection, browse to the *Tutorial* Document Library and click *Next*.
6. In the *Events Selection* screen select *Item Updated*. Although we could select the *Item Added* option as well to automatically convert new documents, that would require additional logic to see if the created type is of type PDF, which complicates this workflow and is beyond the scope of this tutorial. Click *Next*.
7. This tutorial does not require any of the item's Metadata, so click *Finish* to complete the wizard.
8. Using the *Event Wizards* pane drag the *SmartObject Event* onto the design surface and click *Next* on the first screen of the Wizard.

9. Enter *Convert Document* as the *Event Name* and click the ellipses (…) next to *SmartObject Method* to open the Context Browser.
10. In the Context Browser Navigate to the *Environment* (the first) tab and select *SmartObject Server(s) / SmartObject Server / Muhimbi / Muhimbi Document Converter for SharePoint / Single Method / Convert Document* and click the *Add* button followed by the *Next* button.



11. The Input Mapping screen shows the various properties supported by the *Convert Document* SmartObject.
    a. **Source URL:** The URL of the document to convert (See *Appendix - Specifying path and file names*), please make sure that the web application name (http://yourwebapp) **IS NOT** included in the URL. In this tutorial we are converting the document that started the workflow.
        i. Select *Source URL* and click the *Assign* button followed by the *Ellipses* button behind the *Value* field.
        ii. In the Context Browser select the *Process/Activity Data* tab (the 3rd one) and navigate to *XML Fields / [Your Project Name] / EventDetails / EventDetails / ListItemRelativeURL*. Click the *Add* button followed by *OK*.
    b. **Destination URL:** The optional path and file name of where the converted file will be written to. When left empty the converted file will be saved in the same folder as the source file using the same file name, just with the extension of the specified file type. This field uses the same format and rules as the Source URL field. In this tutorial we'll

leave this field empty. For details on what locations documents can be converted to and how to specify paths, please see *Appendix - Specifying path and file names*.

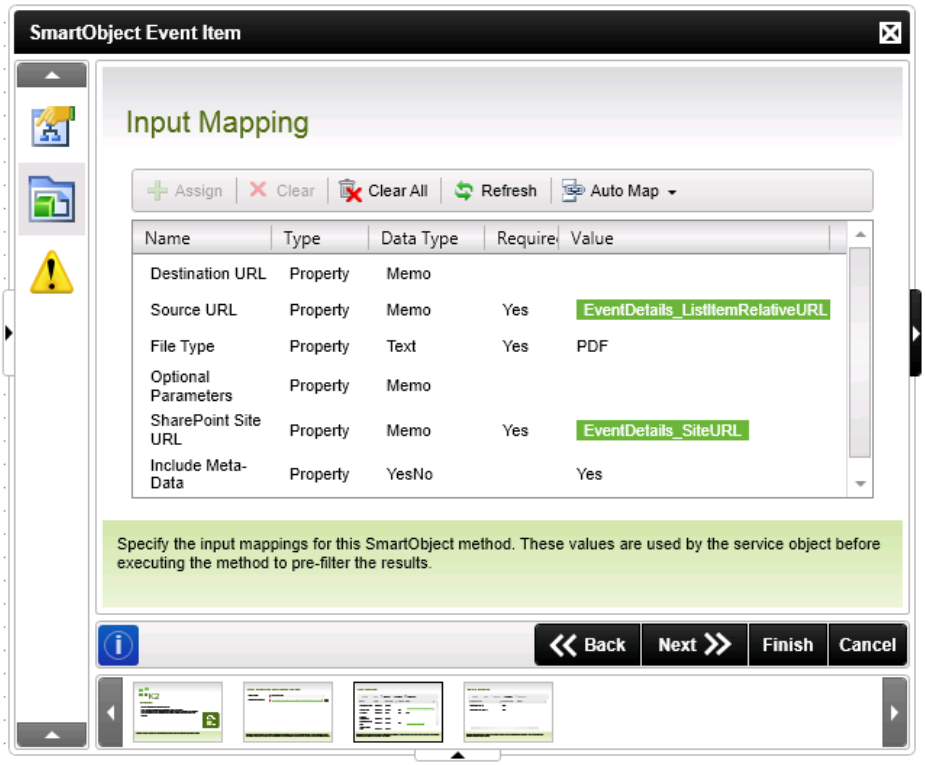
c. File Type: The extension of the file type we are converting to. In this case assign the **PDF** value.

d. **Optional Parameters:** The Muhimbi PDF Converter is a very powerful product that allows many different settings to be specified. It is not feasible to make all of these settings available via individual field mappings, which is why we have developed a special XML syntax to populate these parameters. For this tutorial leave this field empty, you can find more details in *Appendix - Override default conversion settings*. Please keep in mind that K2 Studio does not provide support for entering line breaks in SmartObject mappings, so we recommend creating this XML in a regular code editor (or Notepad) and copy it from there into the *Optional Parameters* field.

e. SharePoint Site URL: Similar to K2's other SharePoint SmartObjects and Wizards, you will need to specify the URL of the site collection the workflow is acting on. The steps are identical to specifying the *Source Url*, just select *SiteURL* from *EventDetails*.

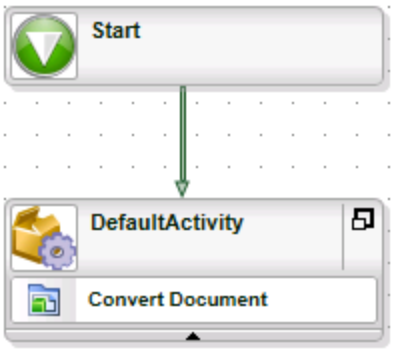
f. Include Meta-Data: In this example we wish to copy all meta-data available on the source document to the converted document. Please assign *Yes* to this field.

Please keep in mind that at the time of writing K2 Does not validate the values specified in *YesNo* fields, so pay extra attention when filling out this field.



In this tutorial we are not using the Return Mappings, so click Finish to close the wizard.

12. We need to connect the Workflow Activity to the workflow. On the Design Surface right-click on the *Start* element and drag a line to the newly created activity.
13. Click the *Deploy* button in the *Home* ribbon. Once the *Deploy Project Wizard* is displayed click the *Finish* button.



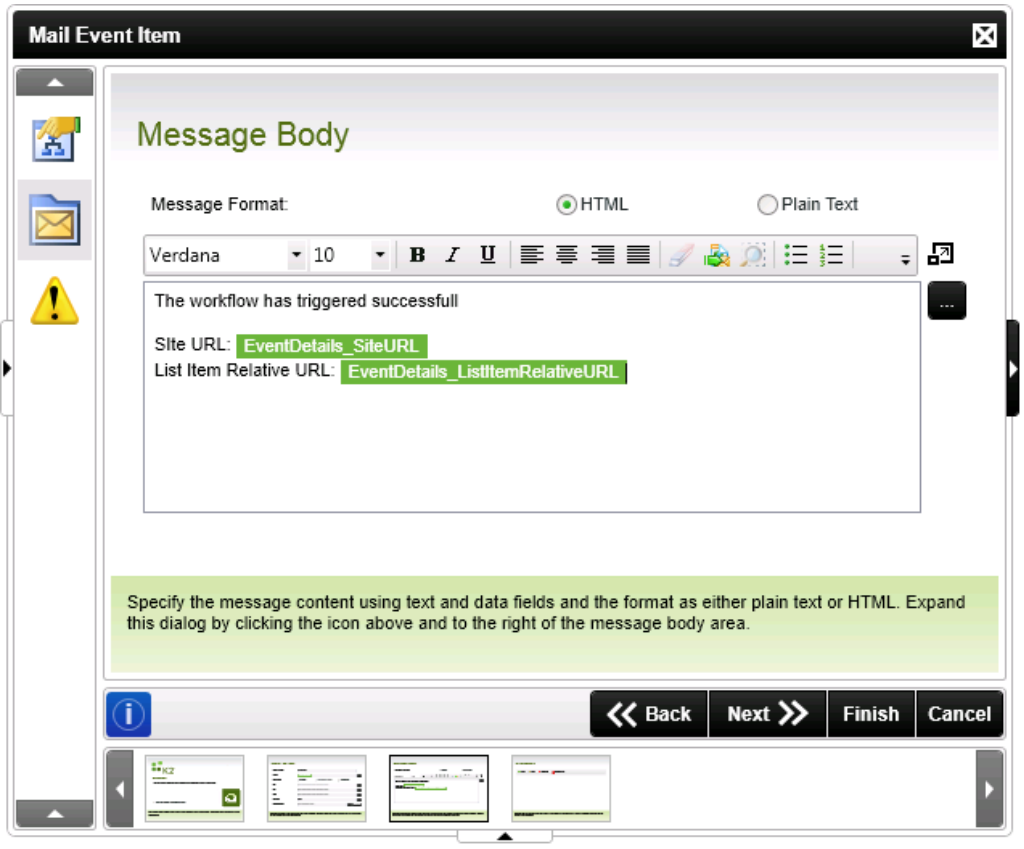
5.2.1 Testing the Workflow

Verify the workflow is working correctly by uploading an MS-Word (or Excel, MSG, TIFF, PowerPoint, or any of the many other formats we support) into the *Tutorial* Document Library. As we have configured the workflow to only trigger when a document is updated, either open the document and resave it, or update the document properties using the ribbon.

If all has been configured well and the workflow has been created correctly then, within a few seconds, a PDF copy of the source file should appear in the *Tutorial* library.

5.2.2 Troubleshooting

If the workflow does not work correctly then either use the *Process Overview* report in *K2 Workspace* to drill down into the workflow, or - and this is what we like to do - insert an *E-Mail event* step before the *Convert Document* step and populate the email with the content of the various lookup fields. Sending an email to the originator makes it easy to verify that the workflow is actually running and get an overview of what is going on without having to dive into the *K2 Workspace* Reports.



5.3 Creating the workflow using K2 Designer

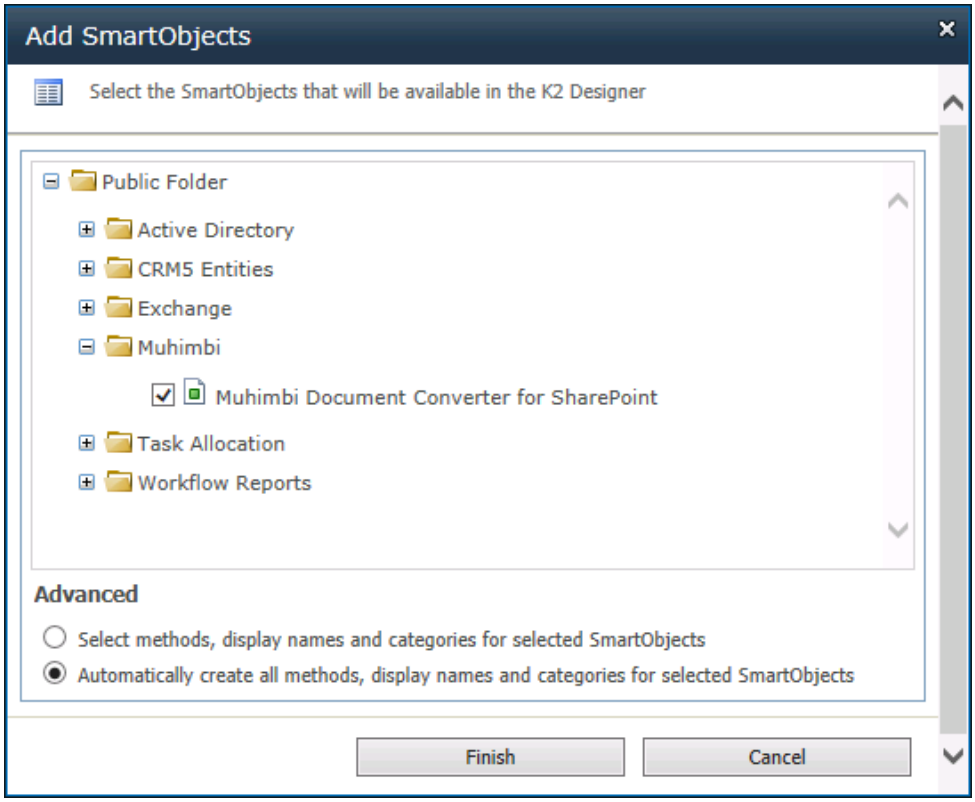
K2 blackpearl comes with a number of different workflow editors. The majority of K2 workflow designers will be most familiar with *K2 Studio*, but *Visual Studio* as well as web-based workflow editors are also available. This section describes how to create a basic workflow to convert documents to PDF using a *K2 Designer* workflow.

5.3.1 Configuring SmartObjects

The Muhimbi PDF Converter for SharePoint is exposed, in K2, as a series of SmartObjects. By default, SmartObjects are not available for use in *K2 Designer*, an administrator must add them. The steps to do so are as follows:

1. In the relevant Site Collection open *K2 Site Settings*. If this option is not available, then please make sure the relevant *K2 Designer* SharePoint Features have been enabled at the Site Collection and Site Level.
2. Under *K2 Designer for SharePoint Management* select *Configure SmartObject Access*.
3. If *Muhimbi Document Converter for SharePoint* is not already present in the list, click *Add new item*.

4. In the *Add SmartObject* Window open the *Muhimbi* Folder and tick the box next to *Muhimbi Document Converter for SharePoint*.
5. Under *Advanced*, select the option to *Automatically create all methods* and click *Finish* to complete the operation.



5.3.2 Creating the workflow

In this tutorial we will create a basic workflow to convert files to PDF and associate the workflow with a Document Library. This workflow is similar to the one we built using *K2 Studio* (See 5.2), but as *K2 Designer* works slightly differently, we have to jump though some additional hoops.

1. Create a new Document Library named *Tutorial2* in a site collection of your choice.
2. In the *Tutorial2* Library, select the Library ribbon tab and click *K2 Workflow*.
3. In the *Welcome* dialog, assuming it is displayed by default, select the *Create a new workflow* option.
4. Name the workflow *Tutorial2*, accept the default settings and, as this tutorial does not need any of the other Wizard screens, click *Finish*.
5. In order for a SmartObject to be able to convert a document a number of parameters are needed. These parameters (*EventDetails.SiteURL and EventDetails.ListItemRelativeURL*) are available from the Context Browser

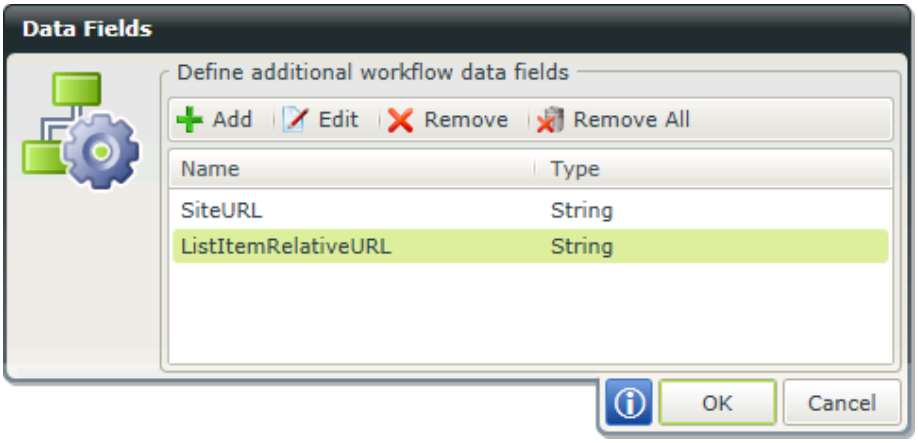
in *K2 Studio*, but in *K2 Designer* we have to manually create them. The steps are as follows:

a. In *K2 Designer* select *File* / *Configure Workflow Settings* / *Data Fields*.

b. Click the *Add* button and specify *SiteURL* as the name.

c. Although we could use a complex regular expression to determine this value at run time, let's keep it simple and specify the URL to the site collection as the Default Value. In our case http://portal.denallix.com/ (including the trailing slash!).

d. Click *OK* and add another Data Field named *ListItemRelativeURL*.



e. Accept the default settings, click *OK* and *OK* again to close the Data Fields dialog.

f. Hover the mouse over the *Start* line in the designer, a button will appear, click it and confirm the question to add a new step.

g. Select the *Workflow Steps* tab in the ribbon and drag the *Set Data Fields* activity onto the newly created step.

h. We already know the *SiteURL*, but we need to calculate the value of the *ListItemRelativeURL* by taking *Document Context.Document URL* and removing the *SiteURL* from the beginning. This is not difficult, just a bit fiddly.

   i. In the Context Browser open *Inline Functions* / *Text* and drag the *Mid(Text,Start)* on top of *ListItemRelativeURL*.

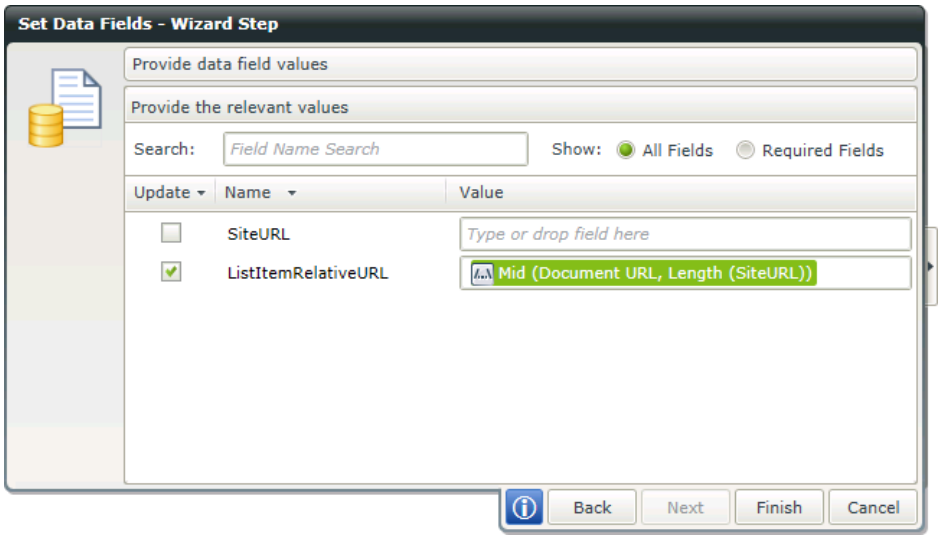
   ii. In the Editor that is opened drag and drop *Document Context* / *Document URL* onto the *Text* field.

   iii. Drag and drop *Inline Functions* / *Text* / *Length* onto the Start field.

   iv. In the Editor that is opened drag and drop *Data Fields* / *Site URL* onto the Text field.

   v. Click *OK* in the various Edit windows and verify that the *Set Data Fields Wizard* looks as per the screenshot below.

   vi. Click Finish to continue.



6. With the required data fields in place, select the *SmartObjects* tab and drag the *Convert Document* smart object onto the empty workflow container and fill out the fields:

   a. **Source URL:** The URL of the document to convert. (See *Appendix - Specifying path and file names*, please make sure that the web application name (http://yourwebapp) **IS NOT** included. In this tutorial we use the previously calculated data field by dragging *Context Browser / Data Fields / ListItemRelativeURL* onto the *SourceURL* field.

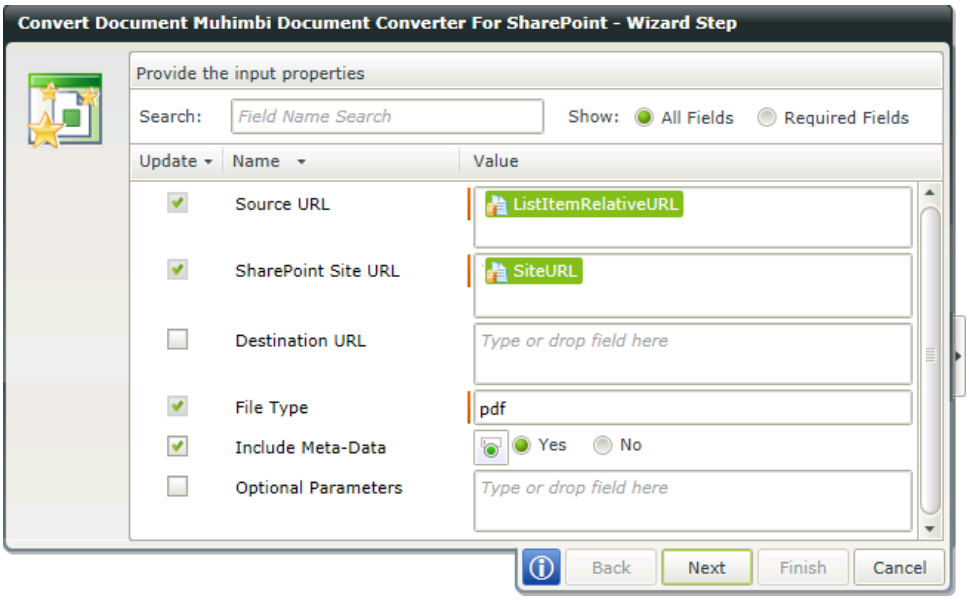
   b. **SharePoint Site URL:** Similar to K2's other SharePoint SmartObjects and Wizards, you will need to specify the URL of the site collection the workflow is acting on. The steps are identical to specifying the *Source Url*, just select *SiteURL* from *Data Fields*.

   c. **Destination URL:** The optional path and file name of where the converted file will be written to. When left empty the converted file will be saved in the same folder as the source file using the same file name, just with the extension of the specified file type. This field uses the same format and rules as the Source URL field. In this tutorial we'll leave this field empty. For details see *Appendix - Specifying path and file names*

   d. **File Type:** The extension of the file type we are converting to. In this case assign the **PDF** value.

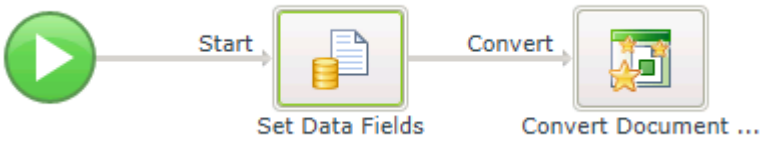
   e. **Include Meta-Data:** In this example we want to copy all meta-data available on the source document to the converted document. Please accept the default *Yes* value

   f. **Optional Parameters:** The Muhimbi PDF Converter is a very powerful product that allows many different settings to be specified. It is not feasible to make all of these settings available via individual field mappings, which is why we have developed a special XML syntax to populate these parameters. For this tutorial leave this field empty, you can find more details in *Appendix - Override default conversion settings*. Please keep in mind that *K2 Designer* does not provide support for entering line breaks in SmartObject mappings, so we recommend creating this XML in a regular code editor (or Notepad) and copy it from there into the *Optional Parameters* field.



Click *Next* followed by *Finish*, this tutorial does not use the return properties.

7. From the *File* option in the ribbon select the *Deploy* option. Click *Next* (twice) followed by *Finish*.



5.3.3 Testing the Workflow

Verify the workflow is working correctly by uploading an MS-Word file (or Excel, MSG, TIFF, PowerPoint, or any of the many other formats we support) into the *Tutorial2* Document Library and manually starting the workflow on the file. To manually start a workflow, in SharePoint open the context menu for the relevant file and select the *Workflows* option.

If all has been configured well, and the workflow has been created correctly then - within a few seconds - a PDF copy of the source file should appear in the *Tutorial2* library.

5.3.4 Troubleshooting

If the workflow does not work correctly then either use the *Process Overview* report in *K2 Workspace* to drill down into the workflow, or - and this is what we like to do - insert a Send E-mail step between the *Set Data Fields* and *Convert Document* steps and populate the email with the content of the various data fields. By sending the email to *Context Browser / Workflow Context / Originator E-mail* it is easy to verify that the workflow is actually running and get an overview of what is going on without having to dive into the *K2 Workspace* Reports.

6 Converting Documents / web pages using hyperlinks

In addition to generating PDF files manually, via workflows and web service calls, it is also possible to convert documents and web pages by simply invoking a URL. This functionality is particularly helpful if you want to place a link to a PDF copy of a file anywhere in a web page or in a *Data View Web Part*.

The latest version of this tutorial is available from the Muhimbi Blog at https://www.muhimbi.com/blog/convert-any-file-to-pdf-format-using-the-data-view-web-part-and-a-simple-hyperlink/

The URL is made up of the following elements:

_layouts/Muhimbi.PDFConverter/Convert.aspx?Action=ConvertAndDownload&ListId=*your_list_id*&ItemId=*your_item_id*&ConversionURL=*url_to_convert*

- **Action:** The action to carry out. Currently the only supported value is *ConvertAndDownload*.
- **ListId:** The ID (GUID) of the list that contains the item to convert.
- **ItemId:** The ID of the item to convert.
- **ConversionURL:** A fully qualified URL of the web page to convert.

Please specify a combination of *ListId* / *ItemId* or the *ConversionURL*. Mixing these parameters is not allowed.

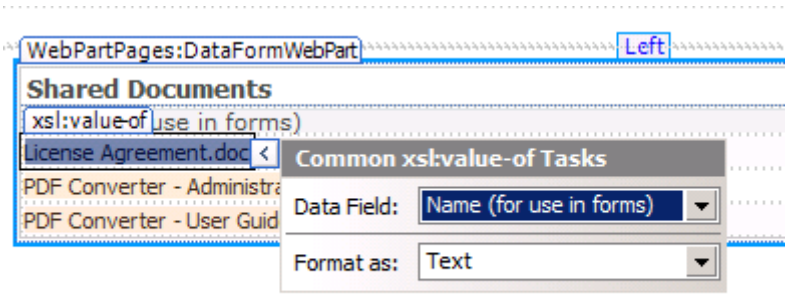
The best way to illustrate how this can be used is by example. In this example we want to make sure that users cannot see the original source files as we don't want them to have access to these documents. Instead, they can access each file via a special hyperlink that downloads the document in PDF format.

This can be achieved in many ways, but in this demonstration, we use SharePoint's *Data View Web Part* to automatically generate a list of documents and link each document to the *ConvertAndDownload* facility. Some familiarity with SharePoint Designer and the DVWP is assumed. For more information see Your First Data View Web Part

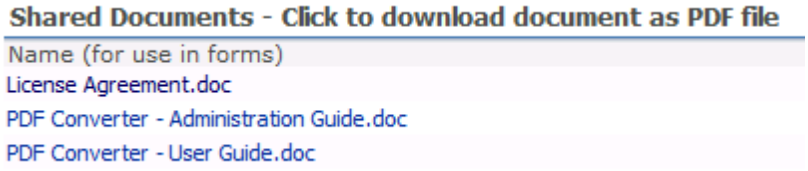
To implement the solution, carry out the steps below:

1. Make sure you have access to a Site Collection, a Document Library named *Shared Documents* and a web part page to insert the DVWP on (We use the home page named *default.aspx*).
2. Start SharePoint Designer and open the Site Collection to work in.
3. Open the web page to add the DVWP to, in our case *default.aspx*.
4. Select a Web Part Zone to add the DVWP to and choose *Insert Data View* from the *Data View* menu.
5. From the newly opened *Data Source Library* click the *Shared Documents* Library and select *Show Data*.

6. In the newly opened *Data Source Details tab* select *Name* (and any other fields you may be interested in) and choose *Multiple Items View* from the *Insert Selected Fields as*… dropdown menu.



7. Any available documents are automatically listed. Click the name of one of the documents followed by the '>' menu. Change *Format as Text* to *Format as Hyperlink.*
8. The previous action opens the *Edit Hyperlink* screen. Enter the following information in exactly this sequence:
    a. **Address:** _layouts/Muhimbi.PDFConverter/Convert.aspx?Action=ConvertAndDownload&ListId={$ListID}&ItemId={@ID}
    b. **Text to display:** {@FileLeafRef}
9. After closing the *Edit Hyperlink* screen, you will most likely see an error about the *ListID* variable not being recognised. Choose *Parameters* from the *Data View* menu to automatically create this ListID parameter and immediately close the *Data View Parameters* screen using the *OK* button.



That is all. Click *Save* in SharePoint Designer and open / refresh the page in your web browser. Click on a document's link to convert it to PDF format and open it in your browser.

7 Processing documents using Web Services

The Muhimbi PDF Converter for SharePoint ships with a powerful, yet friendly, web services interface for integration in your own applications. This same web service is leveraged by all of Muhimbi's SharePoint front end facilities so literally all functionality is exposed this way.

Full code samples for converting, watermarking, securing and OCRing documents are out of the scope of this document. However, full details are available at the following resources:

1. Document: Developer Guide
2. Blog Post: Converting Documents using a web service call (.NET)
3. Blog Post: Converting Documents using a web service call (Java WSimport)
4. Blog Post: Converting Documents using a web service call (Java Axis2)
5. Blog Post: Converting Documents using a web service call (VS2005)
6. Blog Post: Converting Documents using a web service call (PHP)
7. Blog Post: Converting Documents using a web service call (Ruby)
8. Blog Post: Watermarking sample (.NET)
9. Blog Post: Watermarking sample (Java)
10. Blog Post: PDF Merging sample (.NET)
11. Blog Post: PDF Merging sample (Java)
12. Blog Post: Split PDF sample (.NET)
13. Blog Post: OCR Images / scanned files to PDF (.NET)
14. Blog Post: OCR Images / scanned files to PDF (Java)
15. Blog Post: Convert PDF to PDF/A using a web service call.
16. Blog Post: Specifying PDF Viewer Preferences.
17. Blog Post: Set PDF Version, enable Fast Web Views, embed / strip fonts.
18. Blog Post: Adding a Table Of Contents.

More information and samples can be found in the Muhimbi Knowledge base.

8 Controling 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. For details about how to specify the view using a workflow see this blog post.

8.1 Use a special view for exporting to PDF

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

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

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

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

8.2 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.

8.3 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.

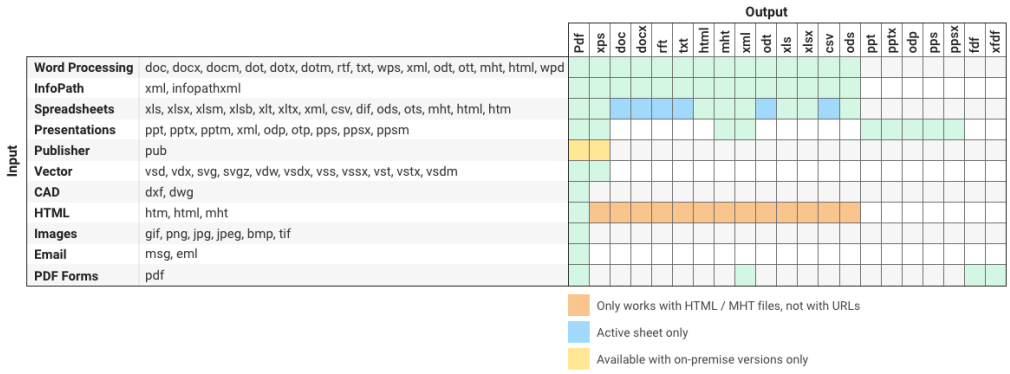
9 Cross-Converting between document types

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

So, how is this useful? Well, let's say that you have a large amount of legacy Office 97-2003 files, but your company now requires all files to be saved in the more modern, and open, Office 2007-2019 formats. By using the Muhimbi PDF Converter you can convert between these formats automatically using a SharePoint workflow, Nintex Workflow or a simple web service call using Java or .NET.

Conversion in the other direction is possible as well. For example, some users in an organisation may have legacy Office 2000 or 2003 installations, but may receive files in Office 2019 format, which no-one else can open. A simple SharePoint workflow 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.



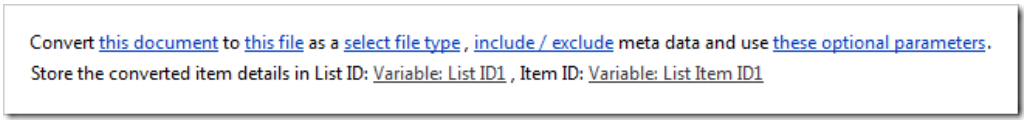
Some points of interest:

1. It is now possible to convert InfoPath files to MS-Word, Excel and HTML For details see section 9.4.
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.

9.1 Cross-Converting file types using SharePoint Designer

Converting a document using SharePoint Designer workflows works similar to converting to PDF. The main difference is that for conversion to non-PDF formats the *Convert Document* Workflow Activity is used rather than the *Convert to PDF* one.

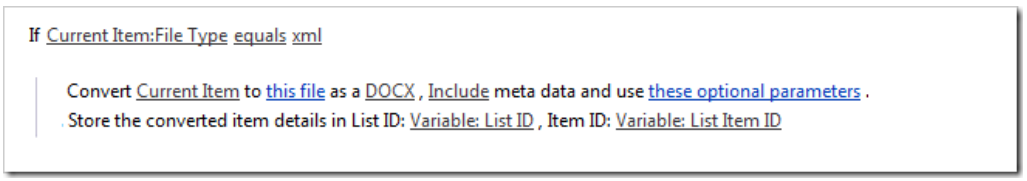
After adding the new activity to a workflow, the following *Workflow Sentence* is displayed.



The workflow sentence is consistent with Muhimbi's other Workflow Activities and is largely self-describing.

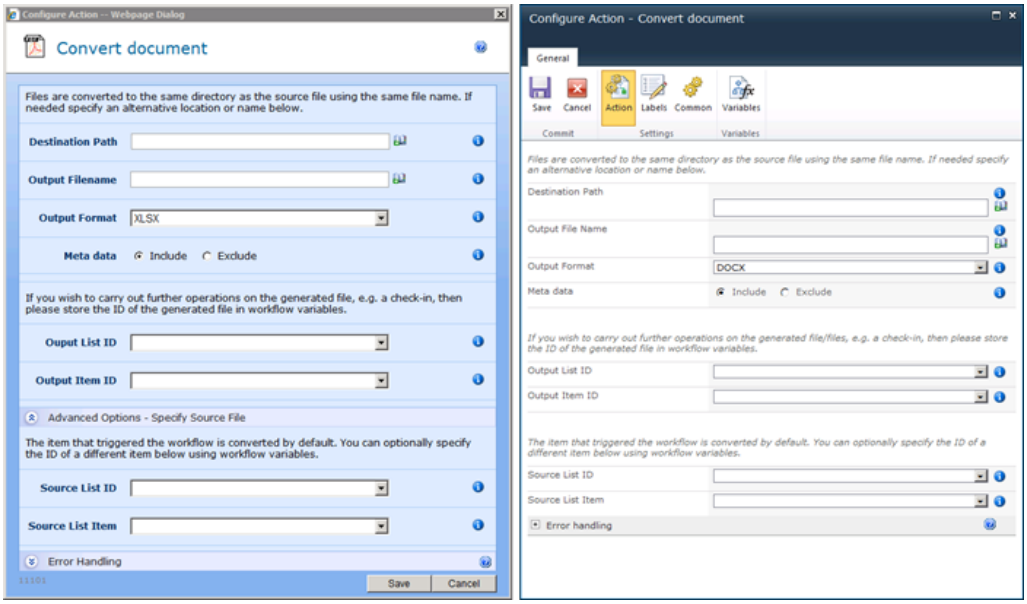
- **This document:** The document to convert. For most workflows selecting *Current Item* will suffice, but some custom scenarios (List or Site workflows) may require the look up of a different item.
- **This File:** An optional filename (and path) to write the converted document to. When not specified, the same name as the document that triggered the workflow will be used, just with a different extension. *Please make sure that the path does not include the host name, e.g., 'http://your site/…'. For more details see Appendix - Specifying path and file names*.
- **Select file type:** Select the type to convert to from the drop-down menu.
- **Include / exclude meta data:** In case of sensitive documents, you may want to strip any custom SharePoint columns from the file. For example, if your document library contains a column named 'Yearly sales forecast' then you may want to select 'Exclude'.
- **Optional parameters:** See *Appendix - Override default conversion settings.*
- **Parameter 'List ID'**: The ID of the list the converted file was written to. This can later in the workflow be used to perform additional tasks on the file such as performing a check-in or out.
- **Parameter 'List Item IDs'**: At the moment this workflow activity will always generate a single output file. However, in the future it will be possible to generate multiple output files in one go, in which case this parameter will return a string with ';' separated values of the generated item IDs. This list can then be used by other (custom) activities to process the individual files further.

A basic sample workflow is included below, by attaching this workflow to a Forms Library any InfoPath form saved in it will automatically be converted to an MS-Word 2007 file.



9.2 Cross-Converting file types using Nintex Workflow

Similar to Muhimbi's other Nintex Workflow activities, the *Convert Document* activity integrates with Nintex Workflow at a deep level. It supports SharePoint 2007-2019, allows errors to be handled and even supports integration with Nintex' iterators to deal with multiple items and loops.



As this Workflow Activity is similar to the generic PDF Conversion Activity it is worth looking at the example in Chapter 4 *Converting Documents using Nintex Workflow*. All parameters are identical with the following exceptions:

- **Output Format:** This field is specific to the Convert Document activity and allows the output format to be specified, e.g., *doc, xls, pdf, txt, csv, etc.*
- **Optional parameters:** See *Appendix - Override default conversion settings.*
- **Output Item ID:** The type of this field is *Text* rather than *Item ID*. The reason for this is that a future version of the software may return multiple, comma separated, values for certain actions.

Please note that you may need to make some small modifications if you intend to convert InfoPath to Excel, HTML or MS-Word. For details see section 9.4 *Convert InfoPath to MS-Word, Excel, XPS and PDF.*

9.3 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 chapter 7 *Processing documents using Web Services*.

9.4 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:

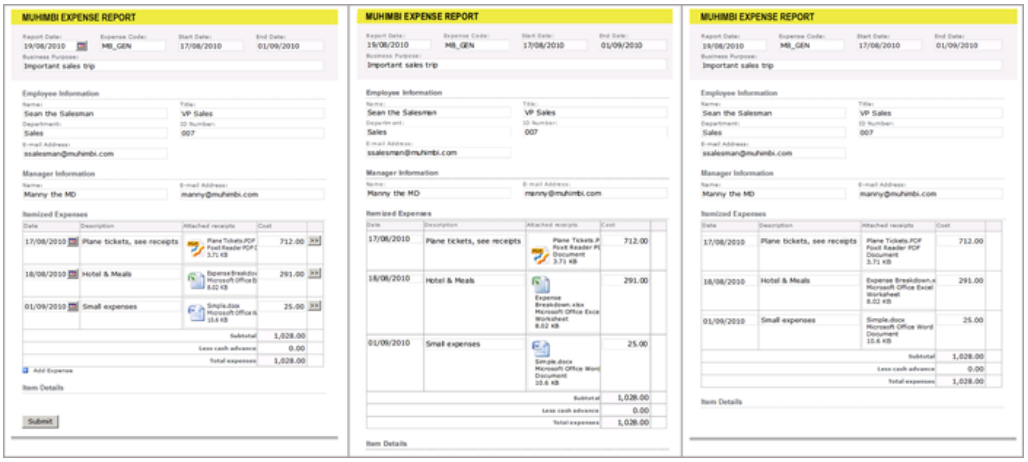
1. **Attachments**: When converting an InfoPath form to PDF the software also converts all attachments and merges them into the main PDF. This is possible because you can represent almost any file format in PDF and merge them together. Unfortunately, this is not possible when converting to HTML, MS-Word or Excel.
2. **View Selection:** The software provides a number of ways to specify which view or views to convert (See chapter 8). When converting to PDF it is possible to specify multiple views, which the converter then merges together into a single document. When converting to HTML, MS-Word or Excel it is only possible to convert a single view as these file formats don't support merging. As a workaround it is possible to create a 'conversion specific view' and combine the content of multiple views in it.

   Print Views are also ignored when converting to HTML, Word or Excel. Instead, you will need to use Muhimbi's View Selection facilities if you wish to convert any view other than the default View.
3. **Formatting:** PDF is a very flexible format that allows any content to be placed anywhere on the page. MS-Word, Excel and HTML are not necessarily this flexible. For example, Excel uses a 'cell-based approach' to display content. If an InfoPath form is not specifically designed for export to Excel, e.g., it uses nested tables or different column widths across a page, then you may need to optimise your InfoPath form for conversion or create a 'conversion specific view'.

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

InfoPath to HTML (MHT)

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

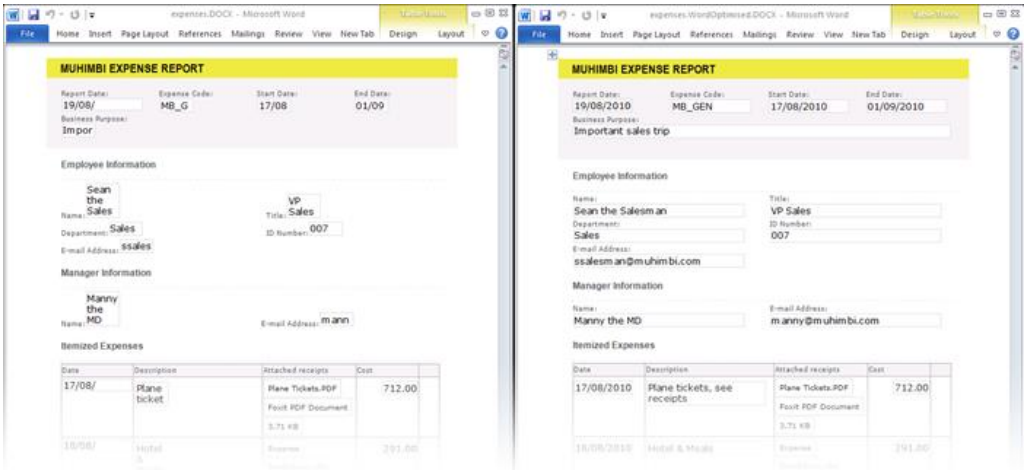


From left to right, the same Form in InfoPath, converted to PDF and converted to HTML

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

InfoPath to MS-Word

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



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

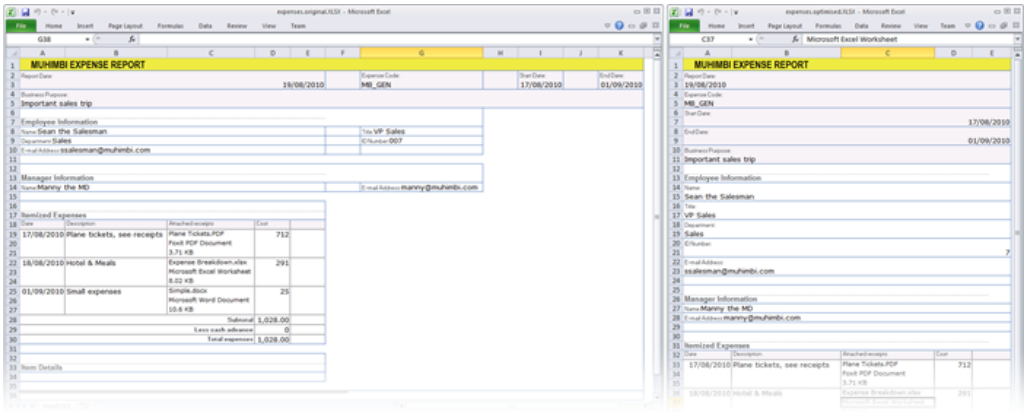
Looking at the ‘before optimisation’ conversion results in the image displayed above, there are 2 things that stand out:

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

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

InfoPath to Excel

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



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

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

1. **Column Widths**: As Excel uses a cell / grid based approach it is not possible to mix different column widths. The information in the form's header requires different column width and spans than the columns used in the repeating table further down the page. By changing the horizontally oriented fields in the header to individual rows we no longer have this problem.
2. **Number formats:** Depending on a cell's content, Excel sometimes tries to be 'clever'. Most of the time this works great, but in this case a field with value '007' is changed into a '7'. This could be fixed by changing the content of the InfoPath field into a formula and concatenating an apostrophe in front of it.

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

10 Merging multiple files into a single PDF

The PDF Converter comes with the ability to merge multiple files, PDF or otherwise, into a single PDF file, either manually using SharePoint's UI, via a SharePoint Designer workflow, a Nintex Workflow or via a Web Service call.

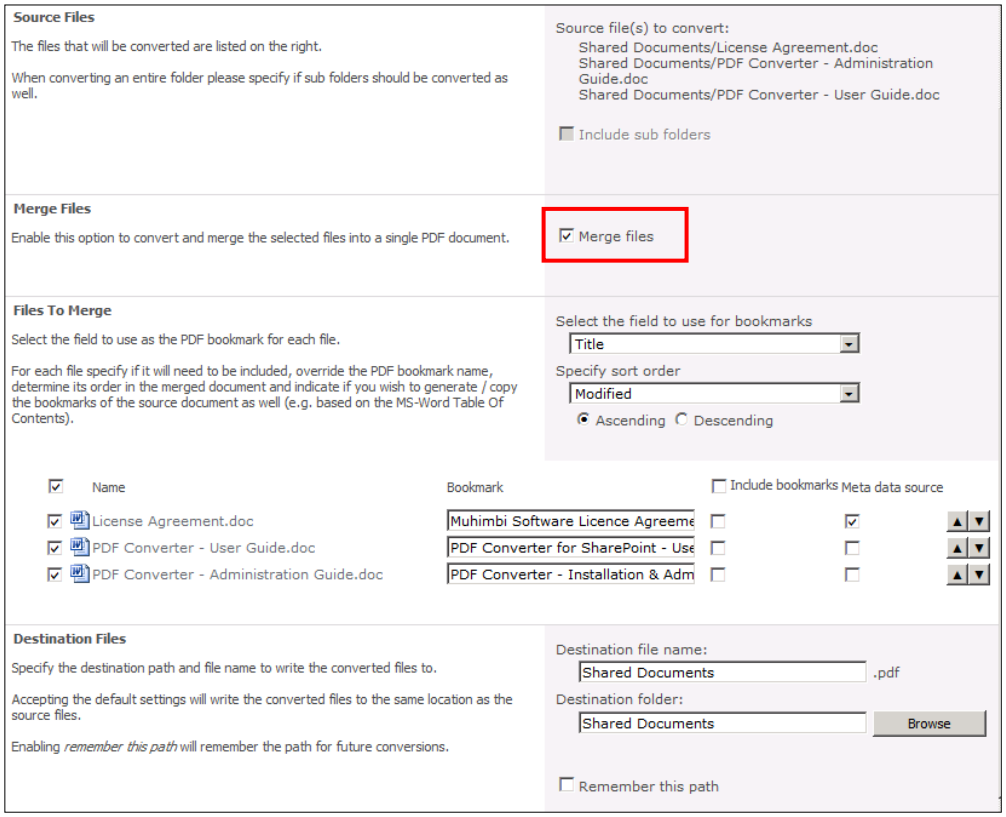
For details about starting each merged document on an *odd* or *even* page (for the purpose of double sided printing) see this blog post.

10.1 Merging files Using the SharePoint User Interface

Chapter 2 *Converting documents using the SharePoint U.I.* describes how to convert individual as well as multiple files to PDF format. When multiple input files are selected the default behaviour is to convert all files individually. However, clicking the '*Merge Files*' option allows the files to be combined into a single PDF file as per the screenshot below.

You can specify the following options:

- **Select the field to use for bookmarks:** If you wish you can automatically generate a PDF bookmark for each file that is merged. This makes navigating through the PDF file a lot easier. You can select any field defined on the document library or '– *Empty* – ' to skip the bookmarking process. The default field is '*Title*', which usually contains the most descriptive data.
- **Enabled / disable documents:** For each file you can specify if they should be included in the merged document or not. This can be very useful when selecting an entire folder to convert and merge.



- **Manually change bookmark text:** Selecting a field to populate the bookmarks with will automatically generate the text for each file's bookmark. However, if you wish you can manually change this value to anything you like. Please take into account that any manually entered values will be overwritten if you select a different bookmark field name.
- **Change sequence of documents:** By default, all documents are ordered by modification date, with the oldest one on top. You can change the sequence using the *up* and *down* arrows. It is also possible to select the name of the field to sort by, e.g., the *creation date*.
- **Include document bookmarks:** Each source document may already have its own bookmarks, e.g., existing PDF Files or MS-Word headings. By default, these are all stripped out, but if you wish you can select to merge the bookmarks as well. For each document they will automatically be moved to sit underneath any custom bookmark name specified for the file.
- **Copy meta-data source:** Specify which file to copy the meta-data from to the generated PDF file.
- **Change file name:** As we are dealing with multiple files, we cannot use the name of the source file to generate the destination file name. For the merged file we default the file name to the name of the folder the files are located in. If the files are in the root folder, then we use the name of the *Document Library*. You can override the default generated name and select a different folder if needed as well.

Further operations such as applying watermarks and PDF security can be carried out as a post processing step using simple workflows.

Please note that the maximum number of files that can be merged is 200. You will automatically receive a warning message when this threshold is exceeded.

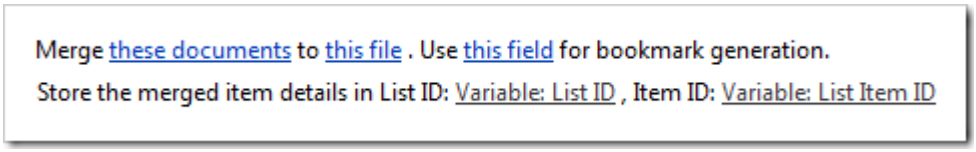
For details about changing the default PDF Bookmark and Sort fields see 19.11 *Changing the default merge bookmark and sort fields*.

10.2 Merging files Using a SharePoint Designer workflow

In addition to being able to convert and merge files using the SharePoint user interface it is also possible to automate this activity using a SharePoint Designer Workflow.

In this section we'll show how to create a workflow that automatically adds a cover page to a document whenever it is created or updated. If the document is not already in PDF Format, it will convert the file as part of the merging process as well. For a good introduction about using the PDF Converter in combination with SharePoint Designer workflows see chapter 3 *Converting documents via a SharePoint workflow*.

The SharePoint Designer Workflow Activity is named *Merge Documents into PDF.* After adding it to your workflow you will see the following *Workflow Sentence.*



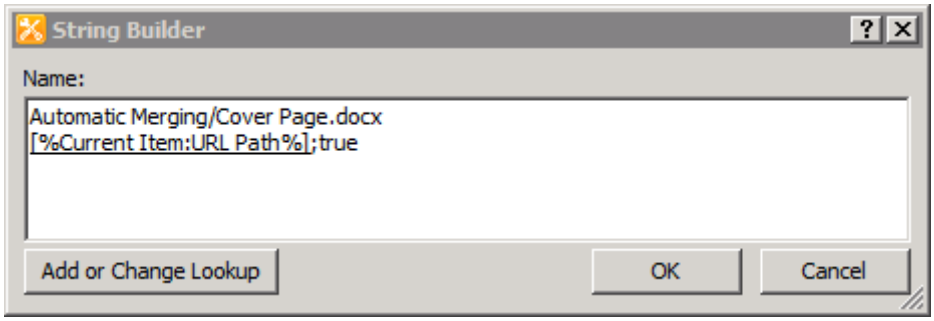
The following fields are available:

- **These Documents:** This field specifies the list of documents / URLs to merge (and convert if needed). Each file must be specified on a new line and each line may contains three ';' separated values:
  - **File Path:** The path to the file to merge or URL of web page to convert. This may already be a PDF file, but if not – and the file format is supported by the converter – then it will be converted first. The path can be relative to the current site (e.g., *Shared Documents/Some File.docx*) or absolute (e.g. */sites/Finance/ Shared Documents/Some Folder/Some file.docx*). SharePoint Designer Workflow Lookups are fully supported, which can be used to dynamically generate the path for the current file, e.g. [%Current Item:URL Path%]. ***In SharePoint 2010 (and later) always use forward slashes in your file path. When specifying a file, please make sure the path does not include the host name, e.g. 'http://your site/…'. If you wish to convert web pages then the path MUST start with http:// or https://.***
  - **Include Bookmarks:** The source file may already include PDF Bookmarks or may be able to generate such bookmarks as part of the conversion process, e.g., an MS-Word file. Specify *true* to copy these bookmarks to the merged PDF file, or *false* to strip out any bookmarks. This value is optional.
  - **Custom Bookmark value:** This field parameter (see below for details) can be overridden using the third parameter. Specify the name of the 'top level bookmark' for the file, specify "" to remove the bookmark for this document or don't specify anything at all to use the value stored in the column referenced by the this field parameter. For an example see the exercise below.

- **This File:** An optional filename (and path) to write the merged document to. When not specified, the same name as the document that triggered the workflow will be used with a '.pdf' file extension. Please make sure the path does not include the host name, e.g. '*http://your site/…'.* For details see *Appendix - Specifying path and file names*.
- **This Field:** In a way similar to the User Interface for the merge facility (see 10.1), it is possible to specify a name of the column which contents will be used to populate the PDF Bookmark for the merged document. For example, specifying *name* or *title* makes it very easy to jump between the various sections of the merged PDF file.
- **Start Page (Each Document):** Control if empty pages are added between documents to make sure each merged document always starts on the right-hand, or left-hand side when printing it double sided.
- **Variable: List ID**: If you wish to carry out further actions on the generated PDF file, e.g., perform a check-in, then you can optionally write the ID of the List the PDF was written to in a workflow variable.
- **Variable: List Item ID**: Similarly, to *List ID,* the Item ID of the generated PDF file can optionally be written to a workflow variable.

Let's create a sample to automatically add a cover page to each document. In this example we will use SharePoint Designer 2010, but the steps for SharePoint Designer 2007 & 2013 are nearly identical.

1. Make sure you have the appropriate privileges to create workflows on a site collection.
2. Create, or navigate to, the Document Library that will be used for this workflow and add a simple MS-Word file named *cover page.docx*. This file can be in any format and, if its content is static, it is recommended to use a PDF file for this purpose so it doesn't need to be converted repeatedly for every merge operation.
3. Open SharePoint Designer, open the relevant Site, create a new workflow, associate it with the Document Library used in step #4 and enable all *Start Options* to make sure the workflow is triggered when an item is created or updated.
4. Add a condition to check that the *type* property of the current item does not equal *pdf*. Otherwise, the workflow will recursively trigger whenever the merged file is written to the same document library.
5. Add the *Merge Documents into PDF* workflow Action and click the these documents parameter.



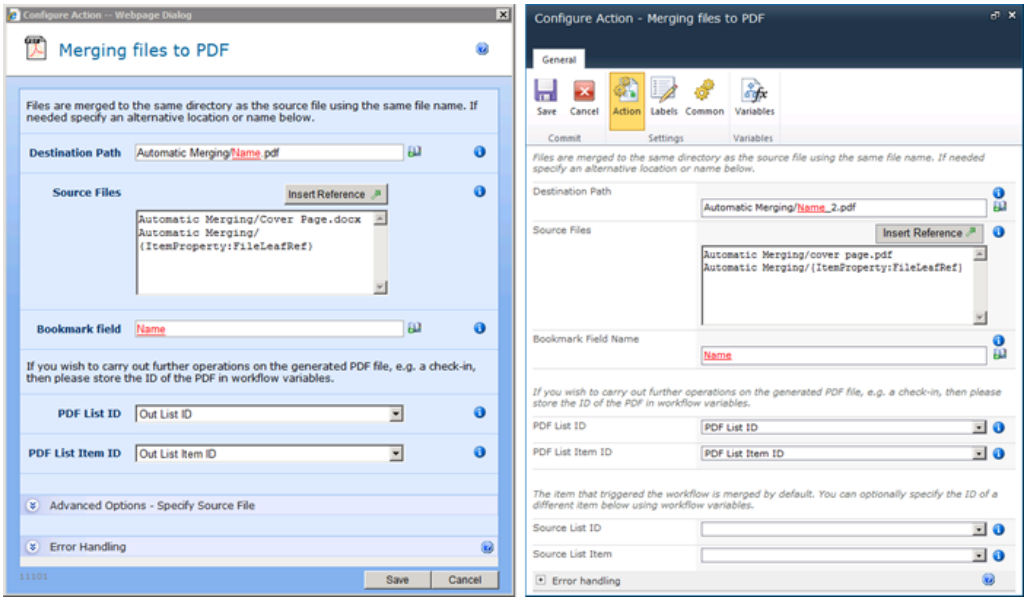
6. Add a line for the location of the cover page, note that our Document Library is named *Automatic Merging.* There is no need to specify the optional *Include Bookmark* and *Custom Bookmark* values, although you can if you wish to.
7. The second file to merge is the one that triggered the workflow. Position the cursor on the 2nd line of the String Builder and click the *Add or Change Lookup* button. Select *Current Item* as the *Data Source* and *URL Path* as the *Field from source*. Click *OK* to close the '*Lookup for String'* window. We want to include the source document's bookmarks so enter "*;true"* at the end of the second line.

   Your *String Builder* should now look similar to the screenshot under step #5. Click *OK* to accept the changes.
8. In this example we want the (merged) output file to use the same name and path as the source file that triggered the workflow, so there is no need to change the "this file" parameter.
9. Click the "this field" parameter and select the field you wish to use for the content of the PDF Bookmarks. In this example we'll select *Name*.

We are all done, publish the workflow and add an MS-Word file with the content to merge to the Document Library. After a few seconds a PDF file will be created consisting of the cover page and the content. Open the PDF file and use the Bookmark pane to quickly navigate to the start of the 2 documents.

10.3 Merging files Using a Nintex workflow

Similar to all other Nintex Activities provided by Muhimbi, the *Merging files (and URLs) to PDF* activity integrates with Nintex Workflow at a deep level. It supports SharePoint 2007-2019, allows errors to be handled and even supports integration with Nintex' iterators to deal with multiple items and loops. For a comprehensive example and details about how to enable the Nintex Workflow integration see Chapter 4. You may also want to look at section 0 for a tutorial about a similar SharePoint Designer workflow activity.



The fields supported by this Workflow Activity are as follows:

- **Destination Path:** Enter the path to write the merged file to, either:
  - Leave it empty to use the same filename (and path) as the file that triggered the workflow.
  - A relative path to a subsite / document library / folder, e.g., *Shared Documents/Some Folder/Some File.pdf.*
  - An absolute path to a different site collection, e.g., */sites/Finance/Shared Documents/Some Folder/Some File.pdf.* Please make sure the path does not include the host name, e.g.*, 'http://your site/…'.* For details see *Appendix - Specifying path and file names*.
- **Source Files:** This field specifies the list of documents to merge (and convert if needed). Each file must be specified on a new line and each line may contain three ';' separated values:
  - **File Path:** The path to the file, or URL, to merge. This may already be a PDF file, but if not – and the file format is supported by the converter – then it will be converted first. The path can be relative to the current site (e.g., *Shared Documents/Some File.docx*) or absolute (e.g., */sites/Finance/Shared Documents/Some Folder/ Some file.docx*). Nintex Workflow References are fully supported, which can be used to dynamically generated the path for the current

file, e.g. *{ItemProperty:FileLeafRef}*. Some string manipulation to generate the exact path to the file may be needed. *When specifying a file please make sure the path does not include the host name, e.g. 'http://your site/…'. When converting a web page then the path MUST start with http:// or https://*

- **Include Bookmarks:** The source file may already include PDF Bookmarks or may be able to generate such bookmarks as part of the conversion process, e.g., an MS-Word file. Specify *true* to copy these bookmarks to the merged PDF file, or *false* to strip out any bookmarks. This value is optional.
- **Custom Bookmark value:** The *Bookmark* field (see below for details) can be overridden using the third parameter. Specify the content of the 'top level bookmark' for the file, specify "" to remove the bookmark for this document or don't specify anything at all to use the value stored in the content specified in the *Bookmark* field.

- **Bookmark:** In a way similar to the User Interface for the merge facility (see 10.1), it is possible to specify a name of the column, or any kind of Nintex Reference, which contents will be used to populate the PDF Bookmark for the merged document. For example, specifying *Name* or *Title* (using the appropriate Nintex Workflow Syntax) makes it very easy to jump between the various sections of the merged PDF file.
- **Start Page (*Each Document*):** Control if empty pages are added between documents to make sure each merged document always starts on the right-hand, or left-hand side when printing it double sided.
- **PDF List ID:** If you wish to carry out further actions on the generated PDF file, e.g., merge additional documents or perform a check-in, then you can optionally write the ID of the List the PDF was written to in a workflow variable of type *String*.
- **PDF List Item ID:** Similarly, to *PDF List ID,* the Item ID of the generated PDF file can optionally be written to a workflow variable of type *Item ID* (in SharePoint 2007) or *Integer* (in SharePoint 2010 and later versions).
- **Source List ID & List Item:** The item that triggered the workflow is merged by default. You can optionally specify the ID of a different List and List Item using workflow variables. Please use the same data types as used by *PDF List ID* and *PDF List Item ID*.
- **Error Handling:** Similar to the way some of Nintex' own Workflow Activities allow errors to be captured and evaluated by subsequent actions, all of Muhimbi's Workflow Activities allow the same. By default, this facility is disabled meaning that any error terminates the workflow.

10.4 Merging files Using a K2 workflow

This section describes the *Merge Documents* method provided by Muhimbi's K2 SmartObjects. For a detailed tutorial about how to use the PDF Converter from K2 see sections 5.2 (*K2 Studio*) and 5.3 (*K2 Designer*).

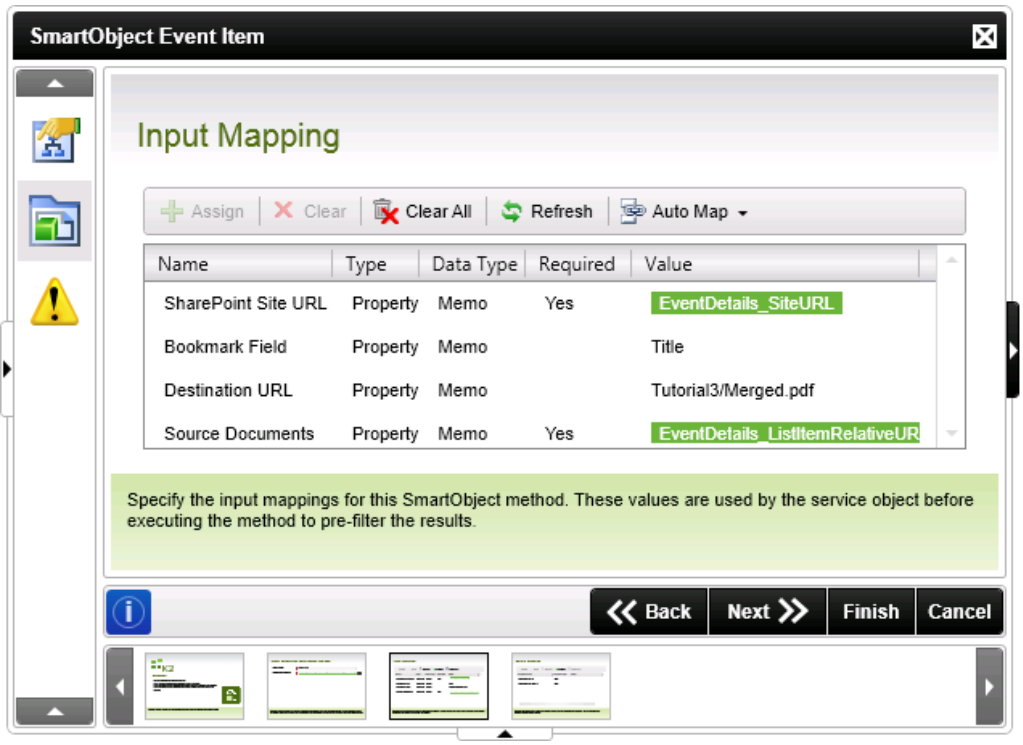
10.4.1 Prerequisites

Before creating the workflow, please make sure Muhimbi's K2 Integration facilities have been deployed as described in the Administration Guide, *Appendix – Deploying K2 Integration facilities*. Basic knowledge of creating workflows in K2 Studio / Designer, and having the privileges, is assumed.

This section was written based on SharePoint 2010. Muhimbi's PDF Converter for SharePoint integrates equally well with other SharePoint versions, but the actual steps for creating K2 workflows differ in each SharePoint version, particularly in SharePoint 2013 and later. Please refer to K2's tutorials and documentation for your particular environment.

10.4.2 The Merge SmartObject Method

The *Merge Documents* method can be added to a workflow by dragging the standard *K2 SmartObject* Event onto the workflow's design surface. Use the SmartObject Wizard's Context Browser to add *Environment tab / SmartObject Server(s) / SmartObject Server / Muhimbi / Muhimbi Document Converter for SharePoint / Methods / Merge Documents into PDF*.



The fields are as follows:

1. **SharePoint Site URL:** Similar to K2's other SharePoint SmartObjects and Wizards, the URL of the site collection the workflow is acting on must be specified. This value can be hardcoded, but the better solution is to use the Context Browser to select the *Process/Activity Data* tab (the 3rd one) and navigate to *XML Fields / [Your Project Name] / EventDetails / EventDetails / Site URL.*
2. **Bookmark Field:** In a way similar to our User Interface for merging file (See 10.1), it is possible to specify a column name which will be used to populate PDF Bookmarks. For example, specifying *Name* or *Title* makes it very easy to use a PDF Reader's Bookmarks pane to navigate between the various sections of the merged PDF file.
3. **Destination URL:** The path and file name to write the merged file to. For details about how to specify paths in Muhimbi's software see *Appendix - Specifying path and file names*.
4. **Source Documents:** This field specifies the list of documents to merge (and convert if needed). As *K2 Studio* does not support line breaks, files are separated using pipes (|) surrounded by a single space, each file may optionally contain three ';' separated values:
    a. **File Path:** The path to the file, or URL, to merge. This may already be in PDF format, but if not – and the file format is supported by the converter – then it will be converted first. The path can be relative to the current site (e.g., *Shared Documents/Some File.docx*) or absolute (e.g., */sites/Finance/Shared Documents/Some Folder/Some file.docx*). *When specifying a file please make sure the path does not include the host name, e.g., 'http://your site/…'. When converting a web page then the path MUST start with http:// or https://.* For details about how to specify paths see *Appendix - Specifying path and file names*.
    b. **Include Bookmarks:** The source file may already include PDF Bookmarks or may be able to generate such bookmarks as part of the conversion process, e.g., an MS-Word file. Specify *true* to copy these bookmarks to the merged PDF file, or *false* to strip out any bookmarks. This value is optional.
    c. **Custom Bookmark value:** The *Bookmark Field* parameter (see above for details) can be overridden using the third parameter. Specify the content of the 'top level bookmark' for the file, specify "" to remove the bookmark for this document or don't specify anything at all to use the value stored in the content specified in the *Bookmark* parameter.

Although a merge list will typically be generated by looping over files using a workflow, the end result passed into the *Source Documents* field will look similar to:

```
Shared Document/file1.docx | Shared Document/file2.pdf | … etc
```

Or

```
SomeLib/file1.docx;true;Introduction | SomeLib/file2.msg;;Summary
```

10.5 Merging files Using a web Service call

For a detailed example about how to convert and merge multiple files using Muhimbi's powerful web services interface see this Knowledge Base article.

11 Splitting PDFs into multiple documents

The PDF Converter comes with the ability to split PDF files up into multiple files using either a SharePoint Designer workflow or a Web Service call.

The key features of the PDF splitting facility are as follows:

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

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

For example, let's assume the following document:

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

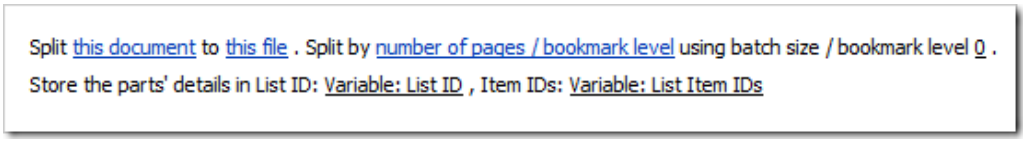
When splitting this document based on bookmarks using '1' as the batch size then the following files will be created:

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

11.1 Splitting files Using a SharePoint Designer workflow

In this section we'll show how to use a SharePoint Designer Workflow to automatically split up an existing PDF file into multiple files containing 10 pages each. This is quite a common scenario for organisations that deal with massive documents who frequently split up these kinds of files in batches of 100 pages to keep the files manageable. If your document is using a format other than PDF, then make sure your use our *Convert to PDF Workflow Activity* first as described in section 3.

The SharePoint Designer Workflow Activity is named *Split PDF*. After adding it to your workflow you will see the following *Workflow Sentence*.



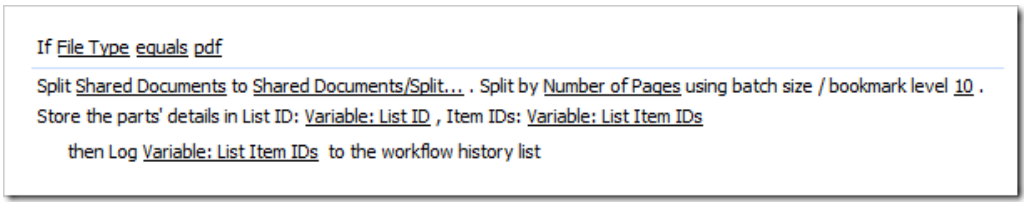
The workflow sentence is consistent with our other Workflow Activities and is largely self-describing. The following fields are available:

- **This document:** The document to split up. For most workflows selecting *Current Item* will suffice, but some custom scenarios may require a different item to be looked up. You may also want to check that the *file type* of the document is '*pdf'* before trying to split it up.
- **This file:** The name and location the split files will be written to are stored in this value. Leave this field empty to use the same folder and file name as the source file, but with sequential numbers added. However, you can optionally specify a path and / or filename template.
    - **Path**: Enter a path, including the Document Library and any folder names, to write the split files to. E.g., "*shared documents/split files/*". You can even specify a different site collection by starting the path with a '/' (never start with 'http:'!). When just specifying a path, without the file name, then make sure to use a trailing '/'.
    - **File Name:** The file name can be anything and allows the standard .NET string formatting facilities for numbering, 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.
- **Number of pages / bookmark level:** Specify if you wish to split based on the *number of pages* or the *level of the bookmark*.
- **Batch size:** When splitting based on the number of pages then this parameter must be set to the maximum number of pages to include in each split file. When splitting based on the bookmark level then this parameter should contain the 'depth' at which to split. E.g., specify '1' to split on top level chapters (Chapter 1, chapter 2, etc.) or a higher number to split at a deeper level (e.g., '2' splits on Chapter 1, 1.1, 1.2, 2, 2.1 etc.)
- **Output Parameter 'List ID':** The ID of the list the split files were written to. This can later in the workflow be used to perform additional tasks on the file such as performing a check-in or out.

- **Output Parameter 'List Item IDs':** Unlike our other workflow activities, this parameter will return a string with ';' separated values of the generated item IDs. This list can then be used by other (custom) activities to process the individual files further.

With all the theory out of the way, let's create a simple example to split up PDF files in batches of 10 pages.

1. Make sure you have the appropriate privileges to create workflows on a site collection.
2. Create a new workflow using SharePoint Designer.
3. Associate the workflow with the library of your choice. Do not tick any of the boxes next to the '*Automatically start….*' options, we want to start this workflow manually. If you wish to run this workflow automatically then you may want to add an extra column to determine if a file has been split before, similar to the technique used in 14.1 *Secure files using SharePoint Designer Workflows*.
4. Design the workflow as per the following screen. In summary it does the following:
   - Check if the file is in PDF format. Otherwise, it cannot be split.
   - The 'split' files are written to a folder named 'Split Files' so make sure this folder exists. e.g. "*Shared Documents/Split Files/spf-{0:D5}.pdf*". You can leave our sample file name or merge the file's name in using workflow lookups.
   - Log the generated list of Item IDs to the workflow history.



Publish the workflow and create / convert / upload a PDF file in the document library. From the file's context menu select 'Workflows' and run your workflow. Depending on the size of the document the split files will be generated in a matter of seconds.

11.2 Splitting files Using a web Service call

For a detailed example about how to split PDFs into multiple files using Muhimbi's powerful web services interface see this Knowledge Base article.

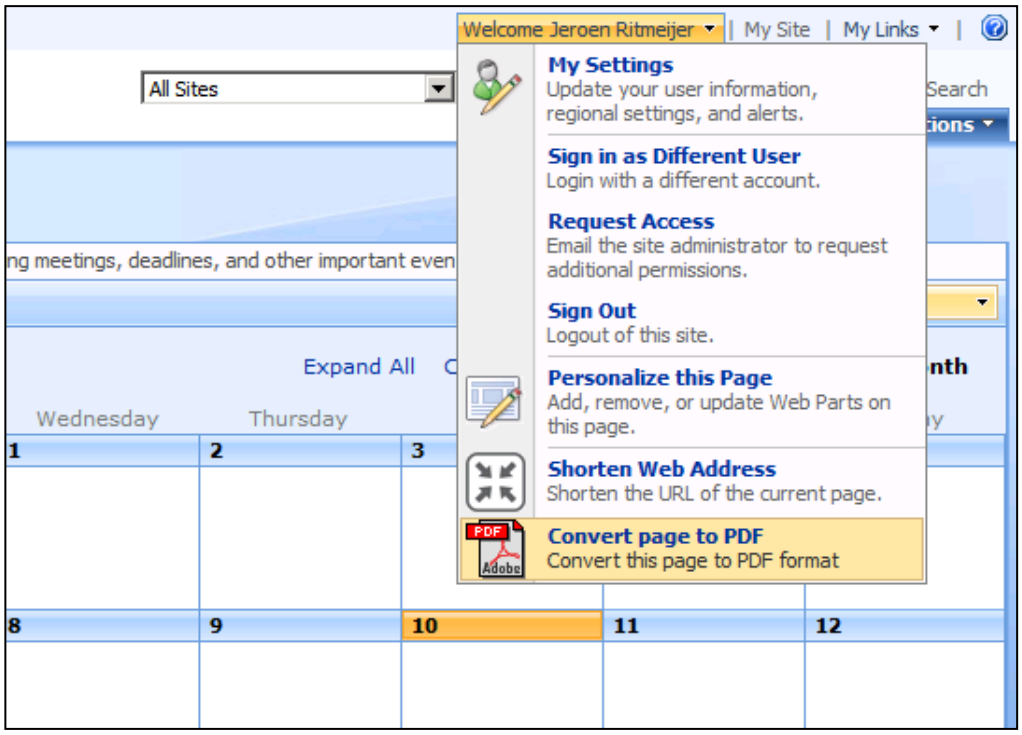
12 Converting HTML and web pages to PDF format

The PDF Converter comes with the ability to convert HTML fragments as well as entire web pages to PDF format, either manually, via a SharePoint Designer workflow, Nintex Workflow or via a Web Service call. For details about troubleshooting HTML conversions, see:

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

12.1 Manually converting a web page

When the *Muhimbi.PDFConverter.ConvertWebPage.Site* feature is enabled on a site collection then an option to convert the current web page to PDF format is automatically added to the *Personal Actions* menu.



Select this option to convert the current page and download it as a PDF file. This is perfect for quickly saving a list or web page as a PDF file.

The default page size, orientation, margin, scale mode and many other settings can be tweaked in the service's config file. For details see *HTML Specific switches* in the *Administration Guide*, subsection *Tuning the Document Conversion Settings*.

12.2 Converting HTML / web pages using SPD Workflows

The PDF Converter for SharePoint ships with a separate SharePoint Designer Workflow Activity that allows HTML fragments or web pages to be converted to PDF format. This chapter contains an example of how this functionality can be used to convert a SharePoint list to PDF format. This is only an example; the Workflow Activity can be used to convert any Web Page to PDF format and is not just limited to SharePoint pages[6].

The latest version of this tutorial can be accessed from the Muhimbi Blog at https://www.muhimbi.com/blog/how-to-convert-sharepoint-list-to-pdf-in-sharepoint-designer/

In this example we use a simple Tasks list, so open your SharePoint site and create a Tasks lists if one doesn't already exist.

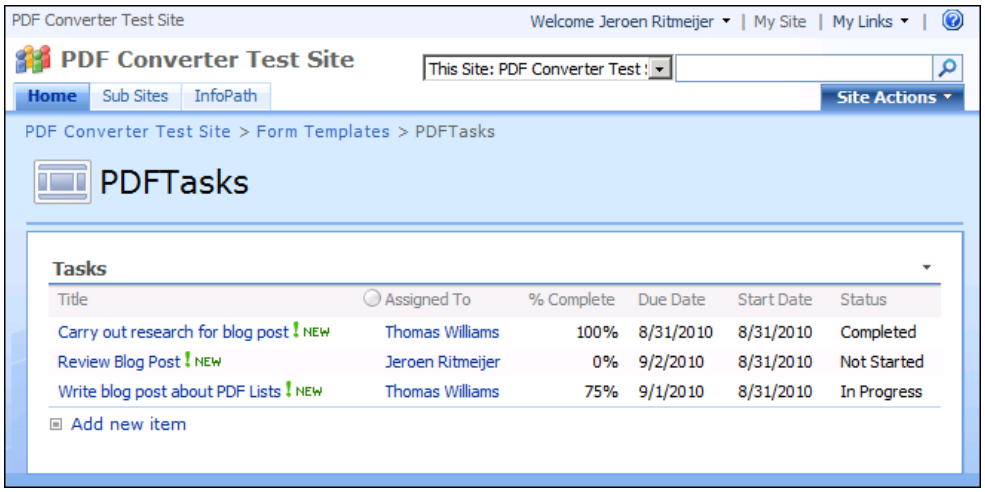
Next up we need to create a page that displays all items in the list with as little of the SharePoint User Interface around it as possible. We could just use the list's default view, but that would convert the Quick Launch menu to PDF as well, which doesn't look very clean. To create a new page without the Quick Launch menu follow these steps:

1. Navigate to the *View All Site Content* screen and click the Create button.
2. In the *Web Pages* column select *Web Part Page*.
3. Name the page *PDFTasks.aspx*, choose the *Full Page, Vertical* template and click the *Create* button.
4. On the newly created page click *Add a Web Part* and add the *Tasks* list.
5. Click the newly inserted Web Part's *Edit* button and select the *Modify Shared Web Part* option.
6. Click *Edit the current View* and select the columns you want to be included in the PDF file. For example, *% Complete, Due Data, Start Date* and *Status*. Do not close the screen yet.
7. Under the *Item Limit* section set the limit to an appropriately large number. We don't want to page through the data in batches as we want to include all items in the PDF file.
8. Click *OK* to save the changes.
9. Save the page's URL as we need it later. E.g.

   http://moss/sites/Management/FormServerTemplates/PDFTasks.aspx.

This new page will be used as the underlying layout for the PDF document. Feel free to modify it further in SharePoint designer / JQuery and remove more parts of the SharePoint user interface. You could also consider creating a minimalistic master page and applying that to the new PDFTasks page.

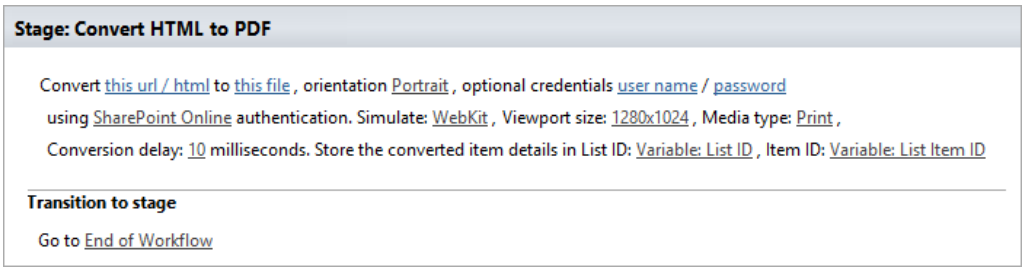
[6] Please note that the default page size, orientation, margin and scale mode can be set in the Muhimbi Service's config file. For details see *HTML Specific switches* in the *Administration Guide*, subsection *Tuning the Document Conversion Settings*.



With the page in place the next thing we need to do is setup the automatic PDF Conversion using a SharePoint Designer workflow. In this example we generate a PDF file whenever the Tasks list is modified. The generated PDF file will be stored in the *Shared Documents* library.

Create the workflow as follows:

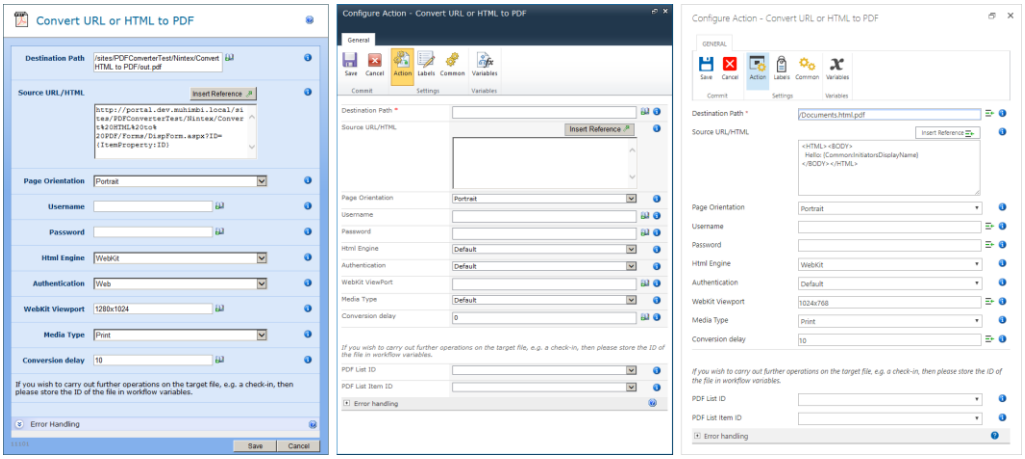
1. Make sure you have the appropriate privileges to create workflows on a site collection.
2. Create a new workflow using SharePoint Designer.
3. On the Workflow definition screen associate the workflow with the Tasks list, tick the boxes next to both '*Automatically start….*' options and proceed to the next screen.
4. Add the *Convert HTML to PDF* action and click on *this url* and enter the URL to the page that was created in a previous step, e.g., http://moss/sites/Management/FormServerTemplates/PDFTasks.aspx.
5. Click *this file* and enter the path and file name where the PDF file will be generated, e.g. *Shared Documents/Tasks.pdf*. **Contrary to the *Convert to PDF* activity, the name of the Document Library must be included in the destination path.**
6. Optionally change the generated page's orientation from *Portrait* to *Landscape*.
7. Optionally increase the *Conversion delay* to deal with JavaScript heavy pages, and the *Media Type* to make use of *Print* or *Screen* specific CSS.
8. The *user name* and *password* fields are optional. By default, the credentials the *Muhimbi Conversion Service* runs under will be used to open the web page. For now, leave it empty.
9. Click the *Finish* button to save and activate the workflow.



As a test enter one or more tasks. Every time a task is added or updated a PDF file is written to *Shared Documents/Tasks.pdf*. Open the PDF file to see the results.

12.3 Converting HTML / web pages using Nintex Workflow

Similar to all other Nintex Activities provided by Muhimbi, the *HTML to PDF Conversion* activity integrates with Nintex Workflow at a deep level. It supports SharePoint 2007-2019, allows errors to be handled and even supports integration with Nintex' iterators to deal with multiple items and loops. For a comprehensive example and details about how to enable the Nintex Workflow integration see chapter 4 *Converting Documents using Nintex Workflow*.



The fields supported by this Workflow Activity are as follows:

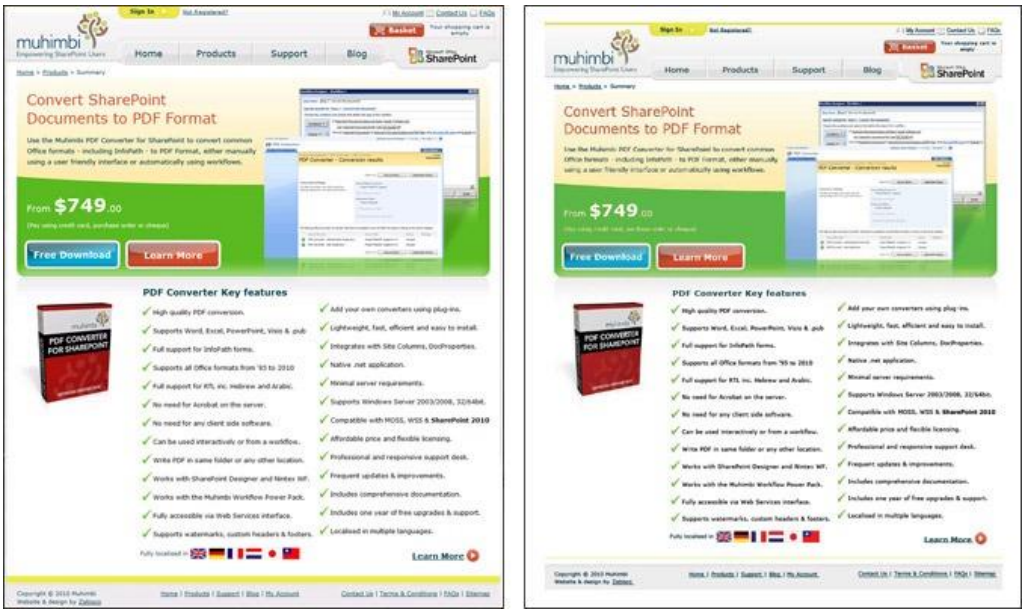
- **Destination Path:** Enter the path to write the converted file to, either:
  - A relative path to a subsite / document library / folder, e.g., *Shared Documents/Some Folder/Some File.pdf.*
  - An absolute path to a different site collection, e.g., /sites/Finance/ Shared Documents/Some Folder/Some File.pdf.

  Always use forward slashes ('/') in path names.
- **Source URL / HTML:** Either a fully qualified URL of the page to convert or an HTML fragment.

- **Page Orientation:** Specify if you wish to use *Portrait* or *Landscape* for the generated PDF file.
- **Username:** By default, all pages are requested by the conversion service using the account the service is running under. If that account has no privileges on the requested URL then specify an alternative user name in this field.
- **Password:** The password associated with the optional user name. Please note that any password entered here is displayed in clear text to allow field references to be added.
- **Html Engine:** Specify which HTML conversion engine to use. *Chromium* (the default on most modern systems) generally produces better results, especially with SharePoint 2013 and later, but for some rare scenarios the IE (Internet Explorer) or WebKit options may work better.
- **Authentication:** By default, the Muhimbi PDF Converter attempts to authenticate using standard Web / HTTP / Windows authentication. However, in order to convert SharePoint Online pages, a different authentication type will need to be specified.
- **WebKit Viewport:** Control the 'virtual browser size' to improve the rendering of responsive websites. (e.g., distinguish between mobile, tablet and desktop versions of a website)
- **Media Type:** HTML is primarily aimed at displaying information on screen, not on paper. However, modern web pages - including SharePoint - use CSS to define a different look and feel for the screen and the printer. Providing the default 'WebKit' based HTML Converter is used, the converter can use Print optimised CSS (the default) or Screen optimised CSS. (The IE based converter always uses the 'Screen' option).
- **Conversion Delay:** Modern web pages, including SharePoint 2013 and later, as well as SharePoint Online, use complex methods for rendering pages, often involving JavaScript. Under normal circumstances the Muhimbi Converter will convert HTML to PDF the moment the page has finished loading. However, you may want to add some milliseconds (a value of 1000 is 1 second) to allow all JavaScript to finish executing.
- **PDF List ID:** If you wish to carry out further actions on the generated PDF file, e.g., perform a check-in, then you can optionally write the ID of the List the PDF was written to in a workflow variable of type *String*.
- **PDF List Item ID:** Similarly, to *PDF List ID,* the Item ID of the generated PDF file can optionally be written to a workflow variable of type *Item ID* (in SharePoint 2007) or *Integer* (in SharePoint 2010 and later versions).
- **Error Handling:** Similar to the way some of Nintex' own Workflow Activities allow errors to be captured and evaluated by subsequent actions, all of Muhimbi's Workflow Activities allow the same. By default, this facility is disabled meaning that any error terminates the workflow.

12.4 Converting HTML / web pages using a Web Service call

HTML to PDF Conversion is accessible via the webservices based interface as well. The latest version of this tutorial is available from the Muhimbi Blog at https://www.muhimbi.com/blog/programmatically-converting-web-html-pages-to-pdf-format/



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

Listed below is a simple C# example[7] of how to carry out a conversion from your own code. The sample code is not complete as it calls into some shared functions from our main C# example to keep things short.

Our existing Java based examples can easily be extended to carry out the same type of conversions.

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

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

[7] Please note that the default page size, orientation, margin and scale mode can be specified in the *ConverterSpecificSettings* property or in the Muhimbi Service's config file. For details see *HTML Specific switches* in the *Administration Guide*, subsection *Tuning the Document Conversion Settings*.

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


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

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

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

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

        // ** Write the PDF file to the local file system.
        string destinationFileName = Path.GetDirectoryName(sourceFileName) + @"\" +
                                     Path.GetFileNameWithoutExtension(sourceFileName) +
                                     "." + conversionSettings.Format;
        using (FileStream fs = File.Create(destinationFileName))
        {
            fs.Write(convertedFile, 0, convertedFile.Length);
            fs.Close();
        }
         // ** Display the converted file in a PDF viewer.
        NavigateBrowser(destinationFileName);
    }
    finally
    {
        CloseService(client);
    }
}
```

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

12.5 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>
```

13 Applying watermarks to documents

The Muhimbi PDF Converter for SharePoint contains a powerful watermarking engine that can be used to add watermarks to Word, Excel, PowerPoint and PDF files, including headers, footers, page numbering and other data. A good overview of watermarking MS-office file formats, as well as some caveats you should be aware of, can be found in this blog post.

Multiple watermarks can be applied to the same page, and watermarks can be applied to page ranges, page intervals or certain page types such as *portrait* or *landscape*[8]. For each watermark it is possible to specify if it should always be displayed or only when printing8.

This chapter explains how simple and complex watermarks can be applied using SharePoint Designer workflows, Nintex Workflows, using the SharePoint User Interface as well as from your own code by invoking the PDF Converter's web services interface.

Section 13.6, *Watermarking field names*, contains an overview of the field names supported by the various watermarks. An overview of which fields are mandatory or optional for each of the possible watermark types as well as the possible values can be found in *Appendix - Watermark field matrix*.

13.1 Applying Individual watermarks using SPD workflows

The Muhimbi PDF Converter comes with a number of watermarking related workflow actions for SharePoint Designer.

These workflow actions allow elements such as text, rectangles, images, PDF files, QR Codes, barcodes as well other shapes to be added to a document either in front of or behind the document's content.

Each separate watermark workflow action is applied to the document separately. If multiple watermarks need to be applied in one go then please have a look at the separate *Composite Watermark* described in section 13.2.

The individual workflow actions are self-describing, but the following elements require special attention:

1. **this document:** The document to apply the watermark to. For most workflows selecting *Current Item* will suffice, but some scenarios require the look up of a different item. You may also want to check that the *file type* of the document is supported (pdf, docx, xlsx, pptx).
2. **this file:** The name and location of the watermarked file. Leave this field empty to overwrite the source file with the watermarked copy. Enter a path, including the Document Library and any folder names, to write the watermarked file to a separate location. E.g. "*shared documents/watermarked files/confidential.pdf*"
3. **List ID:** The ID of the list the watermarked file was written to. This can later in the workflow be used to perform additional tasks on the file such as a check-in or out.

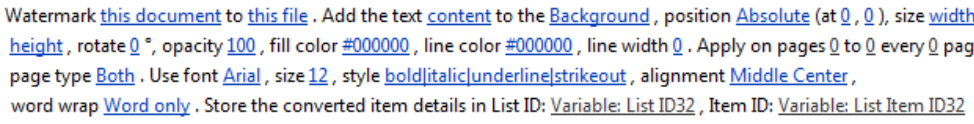
[8] If supported by the underlying file format. PDF is the most flexible, some of the Office formats are more restricted.

4. **Item ID:** The ID of the watermarked file. Can be used with the List ID.

13.1.1 Text watermark

This workflow action can be used to apply text to the foreground or background of one or more pages in the document with full control over the font, style, size, and color.

The SharePoint Designer workflow action is named *Add Text watermark to Document* and the parameters are as follows:



The text stored in the *content* field may contain embedded field codes such as the date or current page number8. For details see 13.7 *Embedding field codes in the Text element*.

SharePoint Designer lookup variables are also supported, which makes it possible for dynamic information stored in workflow variables, or in the Item's columns, to be embedded in the watermark.

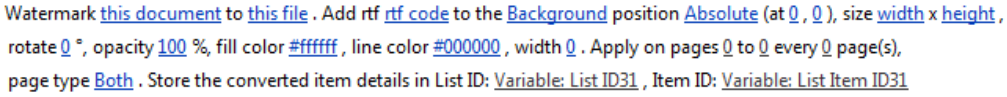
13.1.2 RTF watermark

The RTF watermark allows simple RTF encoded text to be added as a watermark. This allows more control over the look and feel of individual words in the watermark at the cost of added complexity.

An example of valid RTF is as follows:

```
{\rtf1\ansi{\fonttbl\f0\fswiss Helvetica;}\f0\pard
This is some {\b bold} text.\par
}
```

The SharePoint Designer workflow action is named *Add RTF watermark to Document* and the parameters are as follows:



Note that unlike the Text watermark, the RTF watermark does not support embedded field codes. However SharePoint Designer lookup variables[9] are fully supported to make it possible for dynamic information stored in workflow variables, or in the item's columns, to be embedded in the watermark.

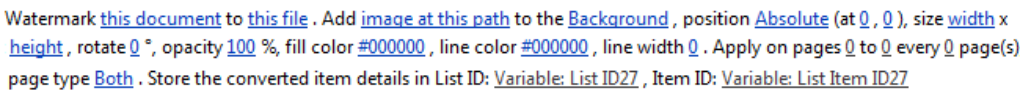
[9] Please note that due to a bug in SharePoint 2010, lookup variables in RTF text may have unexpected side effects. SP2007 works fine.

RTF Watermarks are displayed as plain text when applied to PowerPoint files.

13.1.3 Image watermark

Use the Image watermark to add common image types (BMP, JPG, GIF, PNG, TIFF, WMF, EMF / EMF+) as a watermark to a document.

The SharePoint Designer workflow action is named *Add Image watermark to Document* and the parameters are as follows:

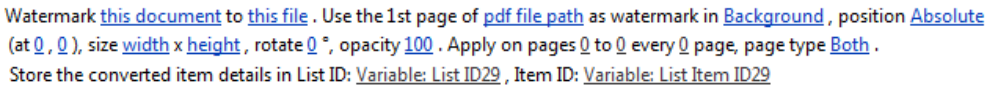


The *image at this path* parameter expects the full path of the image relative to the current site, e.g. ‘shared documents/images/company_logo.gif’.

13.1.4 PDF watermark

Existing PDF files can also be used as a watermark. This could be a dynamic file that is generated from, for example MS-Word, and then converted to PDF using the Muhimbi PDF Converter. Alternatively this could be a static PDF file that has been generated manually.

The SharePoint Designer workflow action is named *Add PDF watermark to Document* and the parameters are as follows:

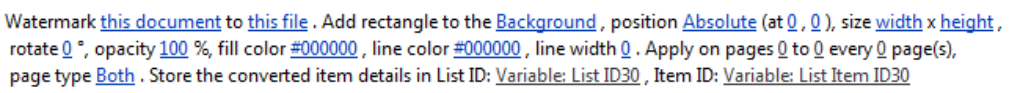


The *pdf file path* parameter expects the full path of the PDF name relative to the current site, e.g. ‘shared documents/static watermarks/company_logo.pdf’.

Note that PDF based watermarks can only be applied to other PDF documents. It is not possible to apply a PDF Watermark to Office file formats.

13.1.5 Rectangle watermark

A simple rectangle can be added as a watermark using the *Add Rectangle to Document* workflow action. The parameters are as follows:

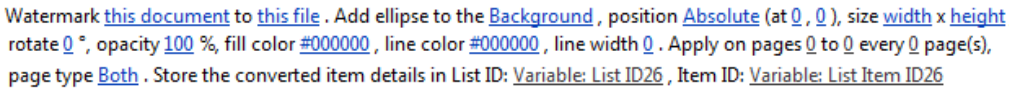


13.1.6 Line watermark

A line can be added as a watermark using the *Add Line to Document* workflow action. The parameters are as follows:

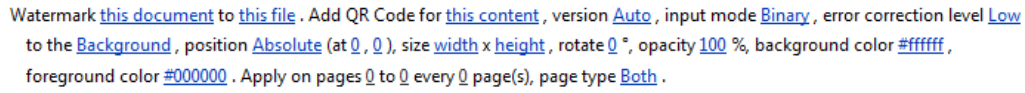
13.1.7 Ellipse Watermark

A circle or ellipse can be added as a watermark using the *Add Ellipse to Document* workflow action. The parameters are as follows:



13.1.8 QR Code Watermark

A QR Code can be added as a watermark using the *Add QR Code Watermark to Document* workflow action.

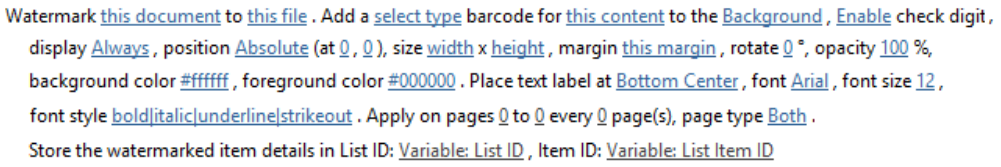


The fields are as follows:

- **Content:** The content to embed in the QR code. This will need to match the specified *input mode*.
- **Version:** Over the years many different QR versions have been introduced. Select the one appropriate to your needs, either *Auto* or *Version01 – Version40*.
- **Input mode:** Specify the appropriate mode for your content:
  - **Binary:** Any value including text, URLs etc.
  - **AlphaNumeric:** Numbers, (Upper case) characters and SPACE, $, %, *, +, -, ., /, :
  - **Numeric:** Numbers only
- **Error correction level:** Select the appropriate level for your needs: *Low, Medium, Quartile, High*

13.1.9 Barcode Watermark

A number of different barcode types can be added as a watermark using the *Add Linear Barcode* workflow action.



The fields, specific to this watermark type, 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.

13.2 Applying composite watermarks using SPD workflows

The individual watermarking workflow actions described in the previous section are very easy to use. The disadvantage however is that if you wish to combine multiple different shapes, e.g. a Circle, an Image and some dynamic text, then a separate watermarking cycle is carried out internally for each shape. This works well and will be fast enough for most occasions, but it is not the most efficient way to do it.

The separate *Add Composite Watermark to Document* workflow action allows multiple watermarks to be applied in one go with each watermark made up of 1 or more individual shapes.

The workflow parameters are as follows:

The real power comes as part of the *watermark.xml* field that stores the XML that describes the multiple watermarks and elements.

The XML for each individual shape is described below, but before we go into more details let's start with an example.

The following sample code describes three watermarks.

- The first adds an image of the company logo (in the foreground) to the top right of each page in the document.
- The second adds the login-id of the user who created / changed the document to a random location in the background in a semi-transparent way.
- The third adds an automatically generated page number to the bottom right of each page.

```
<watermarks>

  <!-- ** First watermark contains a single image element with the logo -->
  <watermark
    hPosition="right"
    vPosition="top"
    width="200"
    height="73"
    zOrder="1"
    opacity="100"
    pageOrientation="both">
    <image
      width="200"
      height="73"
      scaleMode="maintainaspectratio"
      imageFilePath="watermarking/muhimbi-logo.gif"/>
  </watermark>

  <!-- ** Second watermark places the user's name in the background -->
  <watermark
    hPosition="random"
    vPosition="random"
    width="300"
```

```
    height="200"
    zOrder="-1"
    rotation="-30"
    opacity="15"
    pageOrientation="both">
    <text
      width="200"
      height="200"
      fillColor="#000000"
      content="Insert 'modified by' field using SharePoint Designer"
      fontFamilyName="Times New Roman"
      fontSize="24"
      fontStyle="bold|italic"
      wordWrap="word"
      />
  </watermark>

  <!-- ** Third watermark adds page numbering -->
  <watermark
    hPosition="right"
    vPosition="bottom"
    width="100"
    height="40"
    zOrder="1"
    pageOrientation="both">
    <text
      hPosition="left"
      vPosition="top"
      width="100"
      height="40"
      content="Page {PAGE} of {NUMPAGES}"
      fontFamilyName="Arial"
      fontSize="11"
      hAlign="left"
      vAlign="top"
      />
  </watermark>

</watermarks>
```

13.2.1 Watermark element

As the previous example shows, the *watermarks* element may contain multiple *watermark* elements. This *watermark* element again can contain one or more individual shapes such as text and images.

The watermark element acts as a container that determines where the watermark will be located, on which pages, before or behind the text, opacity, print only, as well as which page orientations the watermark applies to.

The XML looks as follows[10].

```
<watermark
  hPosition="optional (absolute, random, left, center, right)"
  vPosition="optional (absolute, random, top, middle, bottom)"
  x="optional"
  y="optional"
  width="required"
  height="required"
  zOrder="optional"
  scaleMode="optional (absolute, exactFit, maintainaspectratio)"
  scaleX="optional"
  scaleY="optional"
  rotation="optional"
  opacity="optional"
  pageOrientation="optional (portrait, landscape, both)"
  startPage="optional"
  endPage="optional"
  pageInterval="optional"
  pageRange="optional"
  printOnly="optional (true, false)"/>
```

13.2.2 Text element

This watermark can be used to apply text to the foreground or background of one or more pages in the target document, with full control over the font, style, size and color.

The XML looks as follows[10]:

```
<text
  hPosition="optional (absolute, random, left, center, right)"
  vPosition="optional (absolute, random, top, middle, bottom)"
  x="optional"
  y="optional"
  width="required"
  height="required"
  zOrder="optional"
  scaleMode="optional (absolute, exactFit, maintainaspectratio)"
  scaleX="optional"
  scaleY="optional"
  rotation="optional"
  opacity="optional"
  lineColor="optional"
  lineWidth="optional"
  fillColor="optional"
  content="required"
  fontFamilyName="optional"
  fontSize="optional"
  fontStyle="optional (regular, bold, italic, strikeout, underline)"
  wordWrap="optional (none, character, word, wordonly)"
  hAlign="optional (left, center, right, justify)"
  vAlign="optional (top, middle, bottom)" />
```

[10] Please omit any attributes marked as *optional* from your code when they are not needed. Leaving them in with the 'optional' text will cause an error.

The text stored in the *content* attribute may contain embedded field codes such as the date or current page number. For details see 13.7 *Embedding field codes in the Text element*.

When a lot of content is anticipated then you may want to remove the *content* attribute and instead place it inside the *text* element, for example:

```
<text
  hPosition="optional (absolute, random, left, center, right)"
  ... rest omitted
  >
   Your content goes here, use a CDATA section if needed.
</text>
```

SharePoint Designer lookup variables are also supported, which makes it possible for dynamic information stored in workflow variables, or in the Item's columns, to be embedded in the watermark.

13.2.3 RTF element

The RTF watermark allows simple RTF encoded text to be added as a watermark. This allows more control over the look and feel of individual words in the watermark at the cost of added complexity.

An example of valid RTF is as follows:

```
{\rtf1\ansi{\fonttbl\f0\fswiss Helvetica;}\f0\pard
This is some {\b bold} text.\par
}
```

The XML looks as follows[11].

```
<rtf
  hPosition="optional (absolute, random, left, center, right)"
  vPosition="optional (absolute, random, top, middle, bottom)"
  x="optional"
  y="optional"
  width="required"
  height="required"
  zOrder="optional"
  scaleMode="optional (absolute, exactFit, maintainaspectratio)"
  scaleX="optional"
  scaleY="optional"
  rotation="optional"
  opacity="optional"
  lineColor="optional"
  lineWidth="optional"
  fillColor="optional"
  rtfData="required"/>
```

[11] Please omit any attributes marked as *optional* from your code when they are not needed. Leaving them in with the 'optional' text will cause an error.

When a lot of content is anticipated then you may want to remove the *rtfData* attribute and instead place it inside the *rtf* element, for example:

```
<rtf
  hPosition="optional (absolute, random, left, center, right)"
  ... rest omitted
  >
   Your content goes here, use a CDATA section if needed.
</rtf>
```

Note that unlike the Text watermark, the RTF watermark does not support embedded field codes. However, SharePoint Designer lookup variables are fully supported to make it possible for dynamic information stored in workflow variables, or in the Item's columns, to be embedded in the watermark.

RTF Watermarks are displayed as plain text when applied to PowerPoint files.

13.2.4 Image element

Use the Image watermark to add common image types (BMP, JPG, GIF, PNG, TIFF, WMF, EMF / EMF+) as a watermark to a document.

The XML looks as follows[12].

```
<image
  hPosition="optional (absolute, random, left, center, right)"
  vPosition="optional (absolute, random, top, middle, bottom)"
  x="optional"
  y="optional"
  width="optional"
  height="optional"
  zOrder="optional"
  scaleMode="optional (absolute, exactFit, maintainaspectratio)"
  scaleX="optional"
  scaleY="optional"
  rotation="optional"
  opacity="optional"
  lineColor="optional"
  lineWidth="optional"
  fillColor="optional"
  imageFilePath="required"/>
```

The *image at this path* parameter expects the full path of the image relative to the current site, e.g. 'shared documents/images/company_logo.gif'.

[12] Please omit any attributes marked as *optional* from your code when they are not needed. Leaving them in with the 'optional' text will cause an error.

13.2.5 QRCode element

A range of different QR Codes can be added to documents using watermarks, ideal for embedding the document ID, or any kind of other SharePoint metadata.

The XML looks as follows[13]:

```
<qrcode
     hPosition="optional (absolute, random, left, center, right)"
     vPosition="optional (absolute, random, top, middle, bottom)"
     x="optional"
     y="optional"
     width="optional"
     height="optional"
     zOrder="optional"
     rotation="optional"
     opacity="optional"
     fillColor="#optional"
     lineColor="#optional"
     scaleMode="optional (absolute, exactFit, maintainaspectratio)"
     scaleX="optional"
     scaleY="optional"
     text="required"
     errorCorrectionLevel="optional (Low, Medium, Quartile, High)"
     inputMode="required (Binary, AlphaNumeric, Numeric)"
     version="optional (Auto, Version01, Version02, Version03
                        ... Version40)" />
```

13.2.6 LinearBarcode element

A range of different barcodes can be added to documents, ideal for embedding tracking numbers, and other metadata.

The XML looks as follows[13]:

```
<linearBarcode
     hPosition="optional (absolute, random, left, center, right)"
     vPosition="optional (absolute, random, top, middle, bottom)"
     x="optional"
     y="optional"
     width="optional"
     height="optional"
     zOrder="optional"
     rotation="optional"
     opacity="optional"
     fillColor="#optional"
     lineColor="#optional"
     scaleMode="optional (absolute, exactFit, maintainaspectratio)"
     scaleX="optional"
     scaleY="optional"
     text="required"
     barcodeType="required (Codabar, Code11, Code32, Code39,
                            Code39Extended, Code93, Code93Extended, Code128,
                            Code128A, Code128B, Code128C, GS1Code128)"
     omitStartStopSymbols="optional"
     disableCheckDigit="optional"
     showCheckDigit="optional"
```

[13] Please omit any attributes marked as *optional* from your code when they are not needed. Leaving them in with the 'optional' text will cause an error.

```
  textLocation="optional (Bottom, None, Top)"
  barcodeToTextGapHeight="optional"
  fontFamilyName="optional"
  fontSize="optional"
  fontStyle="optional (regular, bold, italic, strikeout, underline)"
  textAlignment="optional (Default, Center, Justify, Left, Right)"
  margin="optional" />
```

13.2.7 PDF element

Existing PDF files can also be used as a watermark. This could be a dynamic file that is generated from, for example MS-Word, and then converted to PDF using the Muhimbi PDF Converter. Alternatively this could be a static PDF file that has been generated manually.

The XML looks as follows[14]:

```
<pdf
  hPosition="optional (absolute, random, left, center, right)"
  vPosition="optional (absolute, random, top, middle, bottom)"
  x="optional"
  y="optional"
  width="optional"
  height="optional"
  zOrder="optional"
  scaleMode="optional (absolute, exactFit, maintainaspectratio)"
  scaleX="optional"
  scaleY="optional"
  rotation="optional"
  opacity="optional"
  lineColor="optional"
  lineWidth="optional"
  fillColor="optional"
  pdfFilePath="required"/>
```

The *pdf file path* parameter expects the full path of the PDF name relative to the current site, e.g. ‘shared documents/static watermarks/company_logo.pdf’.

Note that PDF based watermarks can only be applied to other PDF documents. It is not possible to apply a PDF Watermark to Office file formats.

13.2.8 Rectangle element

A simple rectangle can be added as a watermark. The XML looks as follows[14].

```
<rectangle
  hPosition="optional (absolute, random, left, center, right)"
  vPosition="optional (absolute, random, top, middle, bottom)"
  x="optional"
  y="optional"
  width="required"
  height="required"
  zOrder="optional"
  scaleMode="optional (absolute, exactFit, maintainaspectratio)"
  scaleX="optional"
```

[14] Please omit any attributes marked as *optional* from your code when they are not needed. Leaving them in with the ‘optional’ text will cause an error.

```
  scaleY="optional"
  rotation="optional"
  opacity="optional"
  lineColor="optional"
  lineWidth="optional"
  fillColor="optional"/>
```

13.2.9 Line element

A line can be added as a watermark. The XML looks as follows[15].

```
<line
  hPosition="optional (absolute, random, left, center, right)"
  vPosition="optional (absolute, random, top, middle, bottom)"
  x="required"
  y="required"
  endX="required"
  endY="required"
  zOrder="optional"
  scaleMode="optional (absolute, exactFit, maintainaspectratio)"
  scaleX="optional"
  scaleY="optional"
  rotation="optional"
  opacity="optional"
  lineColor="optional"
  lineWidth="optional"/>
```

13.2.10 Ellipse element

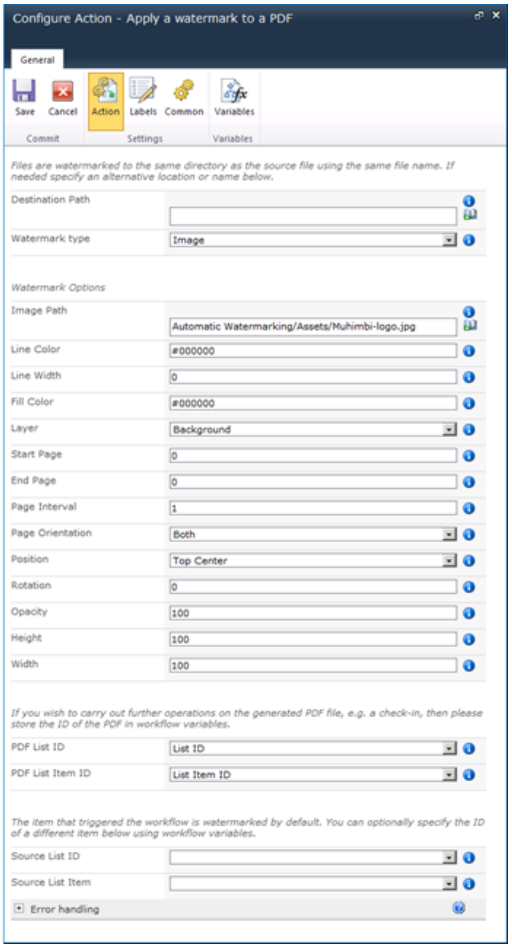
A circle or ellipse can be added as a watermark[15].

```
<ellipse
  hPosition="optional (absolute, random, left, center, right)"
  vPosition="optional (absolute, random, top, middle, bottom)"
  x="optional"
  y="optional"
  width="required"
  height="required"
  zOrder="optional"
  scaleMode="optional (absolute, exactFit, maintainaspectratio)"
  scaleX="optional"
  scaleY="optional"
  rotation="optional"
  opacity="optional"
  lineColor="optional"
  lineWidth="optional"
  fillColor="optional"/>
```

[15] Please omit any attributes marked as *optional* from your code when they are not needed. Leaving them in with the 'optional' text will cause an error.

13.3 Applying watermarks using Nintex Workflow

Similar to all other Nintex Activities provided by Muhimbi, the *Watermark Document* activity integrates with Nintex Workflow at a deep level. It supports SharePoint 2007-2019, allows errors to be handled and even supports integration with Nintex' iterators to deal with multiple items and loops. For a comprehensive example and details about how to enable the Nintex Workflow integration see chapter 4 *Converting Documents using Nintex Workflow*.



The fields supported by this Workflow Activity are as follows:

- **Destination Path:** Enter the path to write the watermarked file to, either:
  - Leave it empty to use the same filename (and path) as the file that triggered the workflow.
  - A file name, without the full path, to write a differently named file to the same location as the source file.
  - A relative path to a subsite / document library / folder, e.g., *Shared Documents/Some Folder/Some File.pdf.*
  - An absolute path to a different site collection, e.g., */sites/Finance/Shared Documents/Some Folder/Some File.pdf.* Please

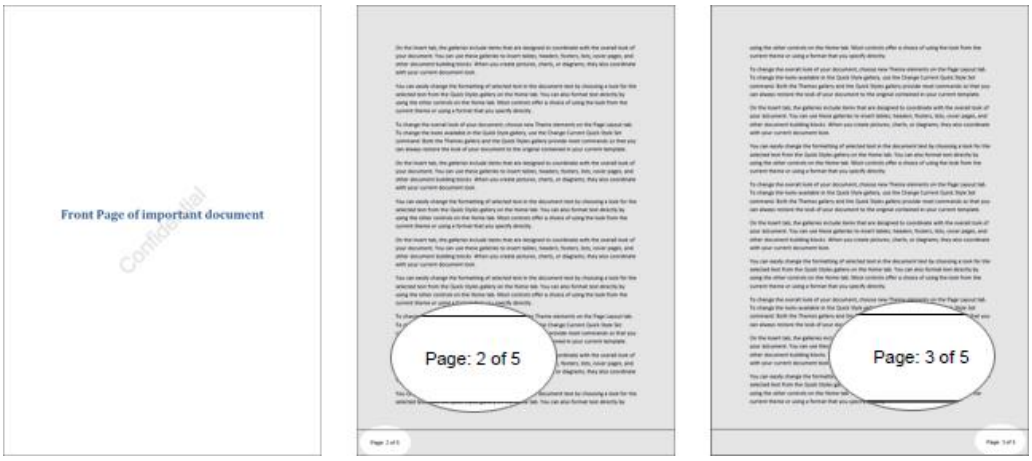
make sure the path does not include the host name, e.g., *'http://your site/…'.* For details see *Appendix - Specifying path and file names.*

- **Watermark type:** A number of different watermark types are supported. Unlike our individual SharePoint Designer workflow activities, all watermark types have been rolled up into a single Nintex Activity. The functionality is largely the same so have a look at section 13.1 for more details. The following watermark types are supported.
  - **Text:** Add a text-based watermark with full control over the font type, size, style, rotation and field codes such as {PAGE}. For details about field codes see *Appendix - Merge codes*
  - **PDF:** Add the first page of another PDF file as the watermark.(Not supported in Office file types)
  - **QR Code:** Add a QR Code as the watermark. For details about the various fields see 13.1.8 *QR Code Watermark*.
  - **Linear Barcode:** Add a barcode watermark. For details about the various fields see 13.1.9 *Barcode Watermark*.
  - **Image:** Use a BMP, JPG, GIF, PNG, TIFF, WMF or EMF / EMF+ file as a watermark.
  - **RTF:** Use RTF based text as a watermark.
  - **Ellipse:** Add an ellipse based watermark.
  - **Line:** Add a line based watermark.
  - **Composite:** For complex watermarks, or watermarks consisting of multiple elements such as line AND text AND images, use the Composite Watermark in combination with our XML based watermarking syntax. For details see section 13.2 *Applying composite watermarks using SPD workflows*.
- **Display:** Either display the watermark *Always*, or only *When Printing*. (When supported by target file format)
- **PDF List ID:** If you wish to carry out further actions on the watermarked file, e.g., send it by email or perform a check-in, then you can optionally write the ID of the List the file was written to in a workflow variable of type *String*.
- **PDF List Item ID:** Similarly, to *PDF List ID,* the Item ID of the watermarked file can optionally be written to a workflow variable of type *Item ID* (in SharePoint 2007) or *Integer* (in SharePoint 2010 and later versions).
- **Source List ID & List Item:** The item that triggered the workflow is watermarked by default. You can optionally specify the ID of a different List and List Item using workflow variables. Please use the same data types as used by *PDF List ID* and *PDF List Item ID*.
- **Error Handling:** Similar to the way some of Nintex' own Workflow Activities allow errors to be captured and evaluated by subsequent actions, all of Muhimbi's Workflow Activities allow the same. By default, this facility is disabled meaning that any error terminates the workflow.

13.4 Applying watermarks using a webservice call

The C# sample below 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.

```
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 output folder contains an MS-Word file.
5. Compile and execute the application.

13.5 Automatically applying watermarks using the SharePoint UI

Although applying watermarks programmatically or via a SharePoint Workflow can be very powerful, not everyone is a programmer, and many people may not actually have the relevant workflow tools such as SharePoint Designer installed on their system.

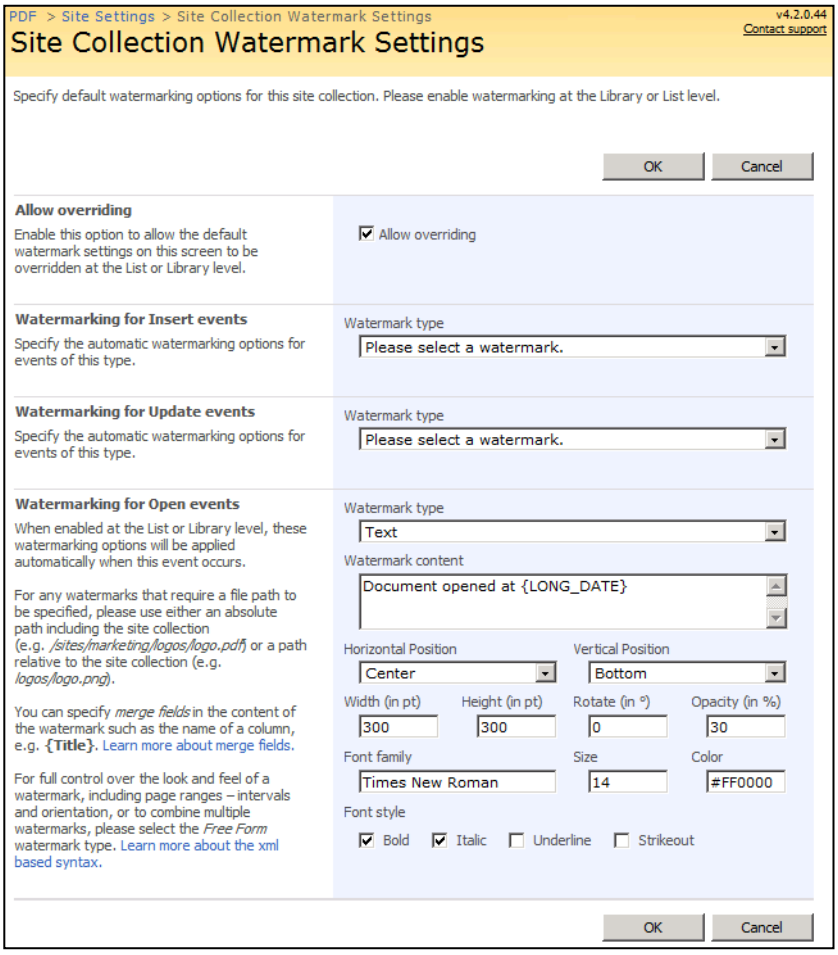
Fortunately, the PDF Converter for SharePoint allows watermarks to be applied using the standard SharePoint user interface. This powerful functionality allows watermarks to be applied to Documents and List Item attachments, and even supports the application of custom, user specific watermarks, when a document is opened, something that cannot be achieved by any other means.

Please note that by default the SharePoint screens associated with this watermarking facility are disabled. They can be activated by enabling the *Muhimbi PDF Converter - Automatic Document Processing User Interface* Feature at either the Web Application or Site Collection level (but not both).

13.5.1 Configuring predefined watermarks at the Site Collection level

To allow the same watermarks to be shared between multiple lists and libraries, the system allows default watermarks to be specified at the Site Collection level using the *Site Actions / Watermarking Settings* screen.

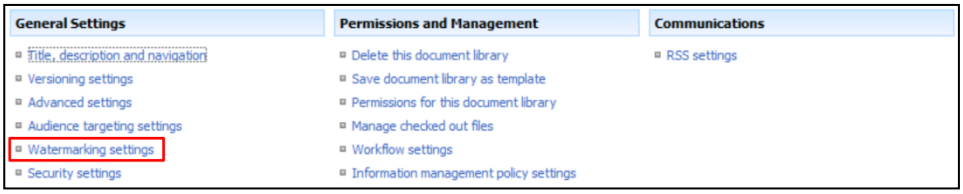
Specifying 'default watermarks' does not automatically enable watermarking on a Library, that must be enabled separately, see 13.5.2.



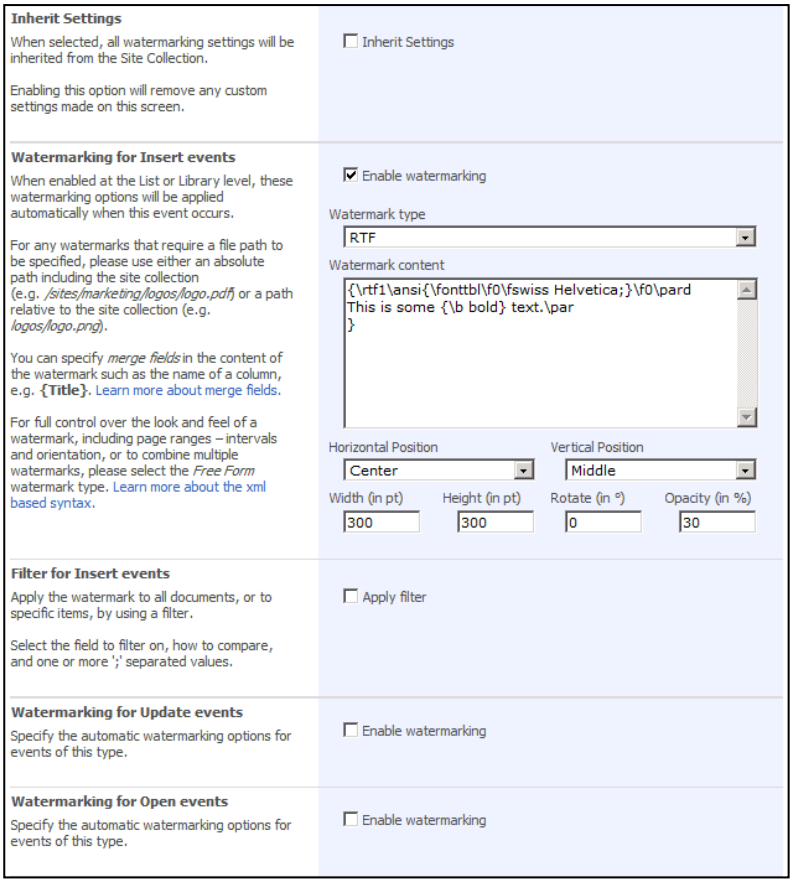
A different default watermark can be specified for *Insert, Update* and *Open* watermarking events. It is even possible to lock down the look and feel of the watermarks to prevent them from being changed at the list or library level.

13.5.2 Configuring automatic watermarking at the List / library level

Even though default watermarks can be configured at the Site Collection level, the actual automatic watermarking must be enabled at the individual List or Library level. The relevant screen, *Watermarking settings*, can be opened from the List or Library's settings screen.



The *Settings screen* allows automatic watermarking to be enabled for *Insert*[16], *Update* and *Open* events. Each event type can use a separate watermark and selection filter (see 13.5.4). Different watermark types can be selected including Text, RTF, Image, PDF and Composite ones (see 13.2).



[16] Due to limitations in SharePoint 2007, automatic watermarking for Insert and Update events are not supported on Document Libraries. They are supported on Lists and in SharePoint 2010. Use Workflows as an alternative (see 13.1)

13.5.3 Merging dynamic data into watermarks

The watermarking framework provided by the PDF Converter for SharePoint comes with a powerful set of *merge codes* that allow dynamic data to be 'merged in' at run-time. For example

```
PDF Opened by '{REMOTE_USER}' on '{LONG_DATE}' from IP '{REMOTE_ADDR}'
```

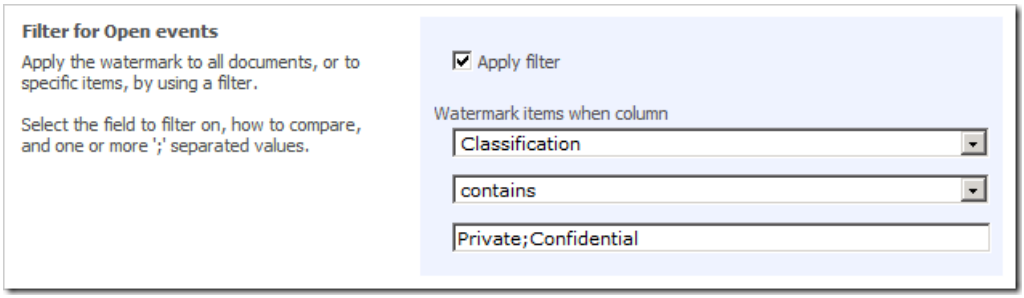
Which results in the following watermark:

Merge codes are not exclusive to the 'watermark on open' functionality, there are several other ways that watermarks can be applied, for example using *workflows* (See 13.1) or *web services* (see 13.4). However, field codes are not always available for a particular interface as, for example, the web service has no knowledge about anything that happens in SharePoint, and workflows are not always associated with an HTTP Context. For a full overview of the available merge codes see *Appendix - Merge codes*.

13.5.4 Specifying filtering criteria when automatically applying watermarks

Similar to the way SharePoint views can be filtered to only display certain list items, Muhimbi's automatic watermarking facility allows filters to be created that determine which documents should be watermarked automatically, and which should not. The possibilities are endless, but some obvious examples are as follows:

- Apply watermarks to *draft* documents, but not to *approved* documents.
- Always apply watermarks to a document created by a user from a particular domain or authentication provider (e.g., FBA)
- Watermark a document, based on the current date, for embargoed documents.
- Enable watermarks on all documents with 'Confidential' in the Title.
- Apply watermarks to *minor versions (*e.g., 1.3*)* of a document, but not to *major versions* (e.g. 2.0).
- Only apply watermarks to certain content types.

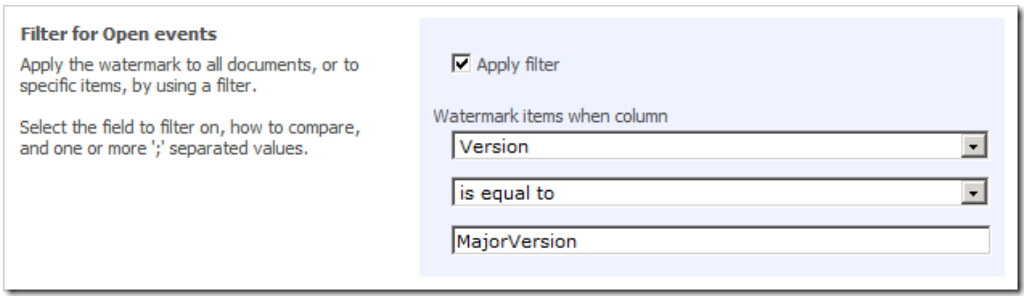


Fortunately, this powerful functionality is very easy to use. Each watermarking section comes with a separate Filter that contains the following fields:

- **Apply Filter:** A simple check box that allows filtering to be toggled. When disabled, watermarks are automatically added to all documents.
- **Field List:** A list of fields that can be used to apply filters on e.g., *Title, Author, Content Type*, etc. This is a 'sanitised' list similar to the one displayed in SharePoint's own *View filter*. Nonsensible 'for internal use only' fields are automatically stripped out.
- **Comparison Type:** Specify the kind of comparison to use, e.g., '*is equal to', 'is greater than', 'contains', 'begins with'* etc.
- **Comparison Value:** The value(s) to compare the selected field with. Full details are provided below.

It is in the *Comparison Value* field where things become interesting as all comparisons take the underlying type of the column into account. All common data types are supported, specifically:

- **Boolean:** True: *True, Yes, -1, 1* and *False, No, 0* as False.
- **Currency:** Allows currency comparisons. Please use numbers only, do not specify the currency symbol.



- **DateTime:** Allows date and time fields to be compared. Enter [Today] (including the square brackets) to compare a date field to the current date, e.g. to apply watermarks to documents modified today. Similarly [Now] can be used to compare a field with the current date and time.
- **Lookup:** Compare lookup fields, including lookups that may contain multiple values.
- **User:** Compares fields that contain a reference to a user account, e.g., the *Modified By* field. Filters of this type expect a value recognised by the underlying Authentication Provider. This will usually be in the '*domain\user_name*' format, but could also be '*fbaprovidername:username*'.

  Enter [me] (including the square brackets) to evaluate the field against the current user. This allows, for example, watermarks to be added to all documents not created by the current user.
- **Single and Multiple choice fields:** Allows single and multiple values to be matched, particularly useful with the 'contains' comparison type. Examples and details are provided below.
- **Numerical fields:** Carries out numerical comparisons.
- **URL:** Can compare fields of type URL.
- **Text:** Any text field or custom field type that uses a text-based representation.

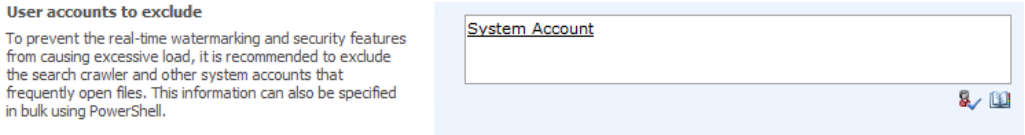
Some additional notes of interest:

- All comparisons are case insensitive.
- The filter for the 'Version' field accepts the '[*MajorVersion*]' and '[*MinorVersion*]' values to distinguish between a major (1.0, 2.0) and minor (1.1, 1.2, 2.2) versions. It also allows a specific version number to be specified.
- When filtering multiple choice / multi select fields you can specify multiple values. For example:
    - "Brett;Ben" Contains "Ben" = True
    - "Brett;Jeroen;Ben" Contains "Brett;Ben" = True
    - "Brett;Jeroen" Contains "Brett;Ben" = False
    - "Brett;Jeroen" is equal to "Jeroen;Brett" = True (The sequence in which values are specified does not matter)
- If a field is deleted from a list after it has been specified in a filter, then the filter will always equate to *false* resulting in the watermark not being applied.
- It is not possible to specify a Filter for Insert events on a Document Library as the fields do not contain any values at this time. This is not a problem for inserting documents attached to a list.
- When comparing fields that contain both Date and Time elements (e.g., the Modified Date) against such values as [Today] then please consider that the value represented by [Today] does not include a time element. Therefore 'Created Date = [Today]' will never evaluate to true unless it is midnight. Instead use 'Created Date >= [Today]'. For Date fields that don't include the Time element, e.g. a birthday, this will not be a problem.

13.5.5 Filtering out system users (e.g. Search indexer)

When using the *Watermark On Open* facility on a list or library, watermarks are applied every time a file is opened, which is exactly what is expected. However, these watermarks are also applied when the SharePoint Search Indexer opens these files. This places excessive load on the system (bulk indexing results in bulk watermarking), so it is recommended to exclude certain accounts from the automatic watermarker.

Starting with version 8.4 of the Muhimbi PDF Converter, a *PDF real-time settings* screen is available from the *Site Actions* menu. Enter a list of accounts to exclude, the *System account* is excluded by default.



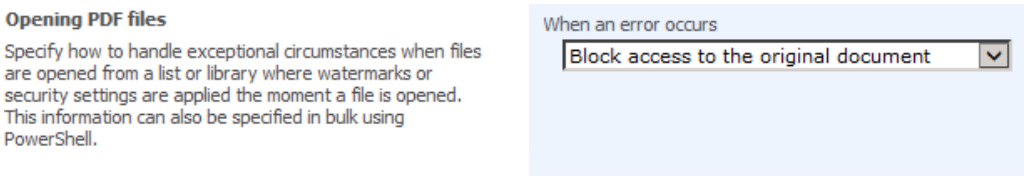
13.5.6 Dealing with watermark errors

Automatically watermarking documents works great, but in the real world you always have to cater for the worst possible situation. In some situations,

watermarks cannot be applied, either because the document is encrypted, corrupt, or an unexpected error occurs, e.g. a temporary network error.

Depending on your exact requirements, you need to define how the system should behave under exceptional circumstances. Should the user still be given access to the document, even though watermarks could not be added, or should access to the document be blocked?

Starting with version 10.0 of the Muhimbi PDF Converter, a *Real-time settings* screen is available from the *Site Actions* menu. Use this screen to specify if access to the document should be blocked in case of an error, or if the unprocessed (pre-watermarked) document should be passed to the user.



For details about how to apply Security to these dynamically watermarked files, see 14.3 *Securing Documents using the SharePoint User Interface*.

13.6 Watermarking field names

Regardless of the method chosen to apply watermarks, you will be presented with various watermarking options such as *EndPage*, *PageInterval*, *PageOrientation* etc. This section explains what each of these names mean.

An overview of which fields are mandatory or optional for each of the possible watermark types as well as the possible values can be found in *Appendix - Watermark field matrix*.

Name	Description
EndPage[17]	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	For MS-Word, the index of the last section this watermark applies to. It is not possible to target individual pages in MS-Word files.
EndX	The 2nd X-coordinate for the Line watermark.
EndY	The 2nd Y-coordinate for the Line watermark.
FillColor	The color of the element's fill in *#rrggbb* or *#aarrggbb* format where *aa* represents the alpha / transparency.
FontFamilyName	The name of the font to use. When the font is not found the system will throw an exception.
FontSize	The size of the font.
FontStyle	The style of the text. Multiple values can be combined.
HAlign	Horizontal alignment of text stored in a *Text* element.
Height	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	The horizontal position of the element.
LineColor	The color of the element's line in *#rrggbb* or *#aarrggbb* format where *aa* represents the alpha / transparency.
LineWidth	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).
Opacity	The Opacity of the element. Note that workflow watermarks use *Opacity* whereas direct web service calls use the opposite term *Transparency*.
PageInterval[17]	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[17]	Specifies what page orientation the watermark applies to: *Portrait*, *Landscape* or *Both*.
PageRange[17]	An optional string representation of the range of pages

[17] Only supported by PDF and PPTX

	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	In case of MS-Word, the kinds of header the watermark applies to *Default*, *First* or *Even*, providing these options have been set for the section header. These values can be combined, so *First* + *Even* is possible.
PrintOnly[18]	Boolean value indicating if the watermark should always be displayed (False) or only when printing (True).
Rotation	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	The behaviour to use when scaling the element, e.g., maintain Aspect Ratio or ExactFit.
ScaleX	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	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.
SectionRange	For MS-Word, an optional string representation of the range of sections the watermark applies to. For example, “1,3,7,10-15”. If specified, this is in addition to the values stored in the StartSection and EndSection properties.
StartPage[17]	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	In case of MS-Word, the index of the first section this watermark applies to.
Transparency	The element’s transparency where 1 means *opaque* and 0 is completely transparent. Note that workflow watermarks use *Opacity* whereas direct web service calls use the opposite term *Transparency*.
VAlign	Vertical alignment of text stored in a *Text* element.
VPosition	The vertical position of the element.
Width	The width of the element. Note that this field is of type *string* to allow the unit of measure to be specified (future version).
WordWrap	The word wrapping behaviour of text stored in a *Text*

[18] PDF Only

	element.
X	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	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	For the watermark, not for individual elements, a negative or 0 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.

13.7 Embedding field codes in the Text element

The *Text* element 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.

For full details see *Appendix - Merge codes*.

14 Securing PDF, Word, Excel and PowerPoint Files

14.1 Secure files using SharePoint Designer Workflows

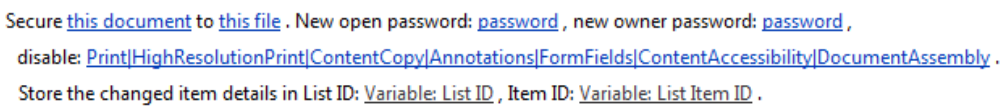
The PDF Converter comes with a SharePoint Designer Workflow Activity that allows security settings such as the *Open Password*, *Owner Password* and document restrictions such as *Printing*[19] and *Content Copying* to be applied from a friendly SharePoint Designer workflow.

The latest version of this chapter is available on-line at the following address:

https://www.muhimbi.com/blog/securing-pdf-files-using-sharepoint-designer-workflows/

The best way to show how this works is by example. What follows is a walkthrough of how to create a SharePoint Designer Workflow to automatically apply security settings to any new or modified PDF file in a Document Library. In this example we use SharePoint Designer 2010, but it works just as well in other SharePoint Designer versions.

Before we begin, let's have a look at the options provided by the *Secure Document* workflow activity.



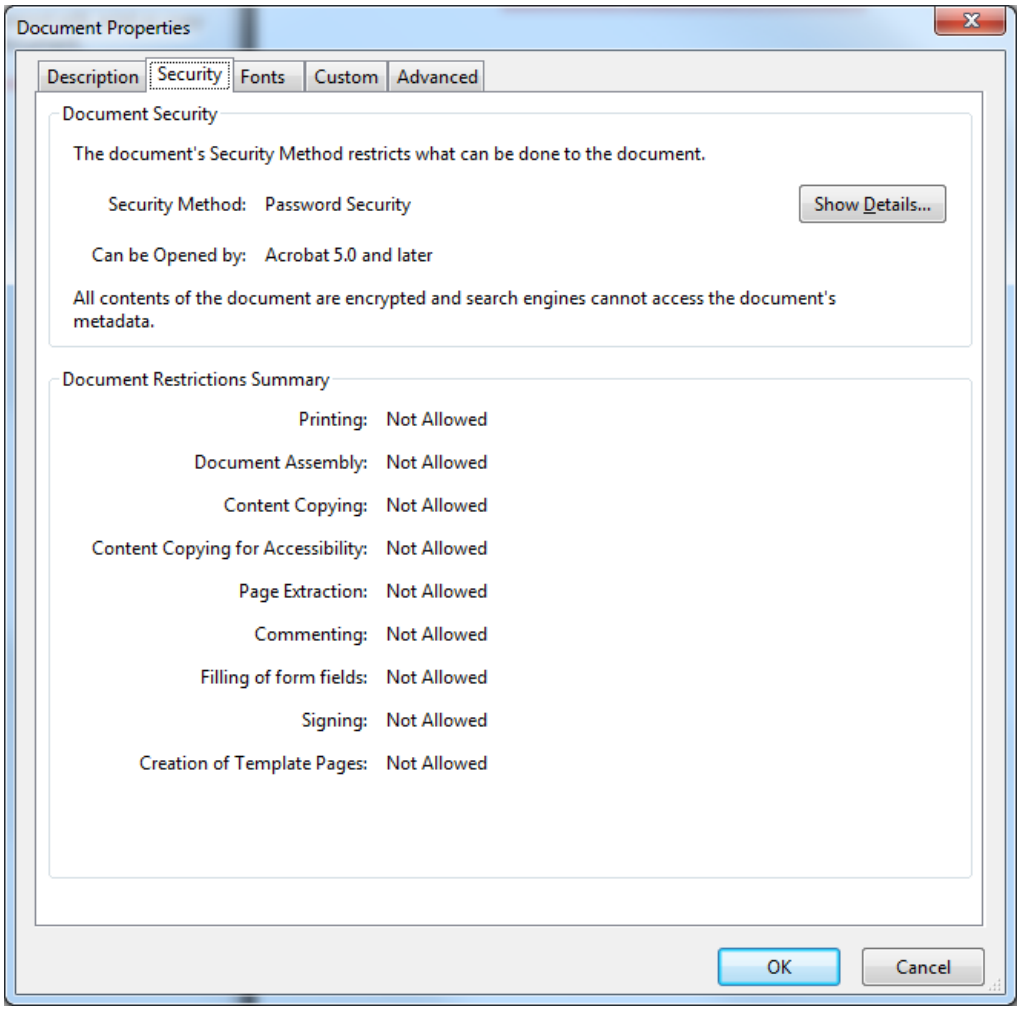
In typical *Muhimbi* fashion, the workflow sentence is consistent with our other Workflow Activities, and is largely self-describing.

1. **this document:** The document to apply the security settings to. For most workflows selecting *Current Item* will suffice, but some custom scenarios may require the look up of a different item. You may also want to check that the *file type* of the document is '*pdf*' before trying to apply security.
2. **this file:** The name and location of the secured file. Leave this field empty to overwrite the source file with the secured copy. Enter a path, including the Document Library and any folder names, to write the secured file to a separate location. E.g., "*shared documents/secured files/confidential.pdf*". You can even specify a different site collection, see *Appendix - Specifying path and file names*.
3. **open password**: When specified, anyone who wants to open the PDF file will need to enter this password.
4. **owner password**: When specified, anyone who wants to change the security settings on the PDF file will need to enter this password.
5. **disable options**: One or more restrictions to apply to the PDF file, separated by a pipe '|' character. By default, it applies all restrictions (*Print|HighResolutionPrint|ContentCopy|Annotations|FormFields|ContentAccessibility|DocumentAssembly*), but any combination is allowed. Enter the

[19] The PDF Format supports all functionality. Word, Excel and PowerPoint support a subset.

word *Nothing* to not apply any restrictions. In order to activate these settings, you must supply an *owner password*.

6. **List ID:** The ID of the list the secured file was written to. This can later in the workflow be used to perform additional tasks on the file such as a check-in or out.
7. **Item ID:** The ID of the secured file. Can be used with the List ID.



Properties of a Secured PDF File.

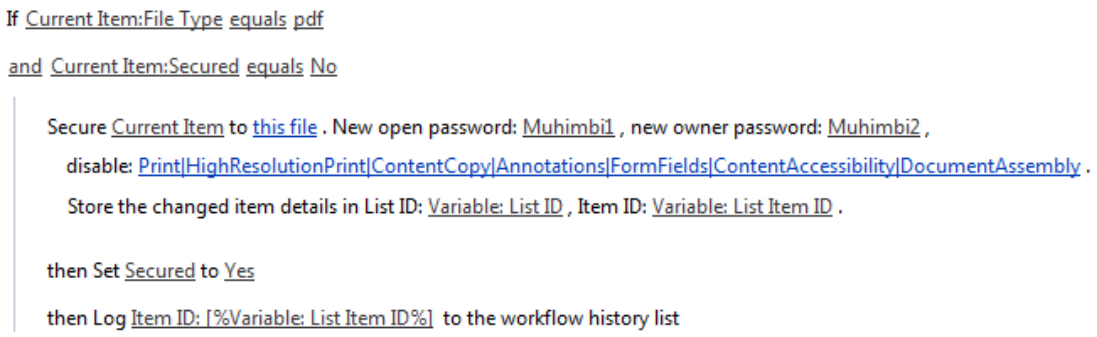
Create the workflow as follows:

1. Download and install the Muhimbi PDF Converter for SharePoint.
2. Make sure you have the appropriate privileges to create workflows on a site collection.
3. Create a new workflow using SharePoint Designer.
4. Associate the workflow with the library of your choice, tick the boxes next to both '*Automatically start….*' options and proceed to the next screen.
5. Because we are running the workflow when PDF files are created as well as modified, a new *Yes/No* column named *Secured* will need to be added to the document library using a default value of *No*. This way we can mark

a document as *secured* and stop workflows from recursively triggering. Alternatively, if files are secured in place, you can decide to just trigger the workflow when new files are added.

6. Design the workflow as per the following screen. In summary it does the following:

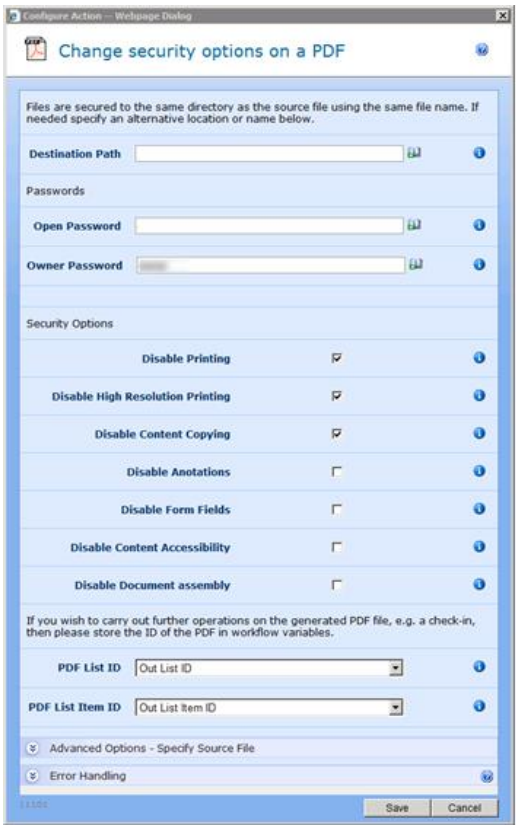
    1. Check if the file is in PDF format. Otherwise, security cannot be applied.
    2. Check if the file has already been secured. If it has then it doesn't need to be secured again.
    3. The PDF File is secured in place. Both Open and Owner passwords are applied, and all restrictions are set.
    4. The file is marked as *Secured* so the workflow doesn't repeatedly run.
    5. A status message is written to the workflow history.

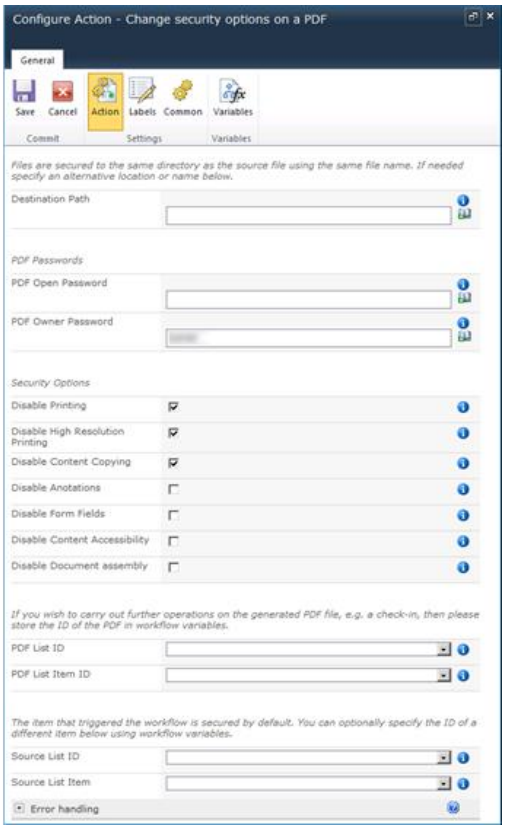


Publish the workflow and create / convert / upload a new PDF file in the Document Library. After a few seconds the workflow column will change to *Completed*, indicating that the file has been secured successfully.

14.2 Secure Documents using Nintex Workflow

Similar to all other Nintex Activities provided by Muhimbi, the *Secure PDF* activity integrates with Nintex Workflow at a deep level. It supports SharePoint 2007-2019, allows errors to be handled and even supports integration with Nintex' iterators to deal with multiple items and loops. For a comprehensive example and details about how to enable the Nintex Workflow integration see chapter 4 *Converting Documents using Nintex Workflow*. You may also want to look at section 14.1 for a tutorial about using a similar Workflow Activity in SharePoint Designer Workflows.





The fields supported by this Workflow Activity are as follows:

- **Destination Path:** Enter the path to write the secured file to, either:
  - Leave it empty to use the same filename (and path) as the file that triggered the workflow.
  - A file name, without the full path, to write a differently named file to the same location as the source file.
  - A relative path to a subsite / document library / folder, e.g. *Shared Documents/Some Folder/Some File.pdf.*
  - An absolute path to a different site collection, e.g. */sites/Finance/Shared Documents/Some Folder/Some File.pdf.* Please make sure the path does not include the host name, e.g. 'http://your site/…'.

  For details see *Appendix - Specifying path and file names.*

- **Open Password:** An optional password that the user must enter in order to open the document. *Please note that any password entered here is displayed in clear text to allow Nintex field references to be added.*
- **Owner Password:** An optional password that the user must enter in order to change the PDF restrictions. When specifying PDF Restrictions then this password must be set. *Please note that any password entered here is displayed in clear text to allow Nintex field references to be added.*
- **Individual PDF Restrictions:** Select the individual restrictions such as *Disable Printing* or *Disable Content Copying*.
- **PDF List ID:** If you wish to carry out further actions on the secured document, e.g. send it by email or perform a check-in, then you can optionally write the ID of the List the file was written to in a workflow variable of type *String*.
- **PDF List Item ID:** Similarly to *PDF List ID,* the Item ID of the secured file can optionally be written to a workflow variable of type *Item ID* (in SharePoint 2007) or *Integer* (in SharePoint 2010 and later versions).
- **Source List ID & List Item:** The item that triggered the workflow is secured by default. You can optionally specify the ID of a different List and List Item using workflow variables. Please use the same data types as used by *PDF List ID* and *PDF List Item ID*.
- **Error Handling:** Similar to the way some of Nintex' own Workflow Activities allow errors to be captured and evaluated by subsequent actions, all of Muhimbi's Workflow Activities allow the same. By default this facility is disabled meaning that any error terminates the workflow.

14.3 Securing Documents using the SharePoint User Interface

Section 13.5 *Automatically applying watermarks using the SharePoint UI* explains how user-specific watermarks can be added the moment a file is downloaded or accessed. This works well, but as watermarks cannot be applied to secured documents these files will need to be secured after watermarks have been added.

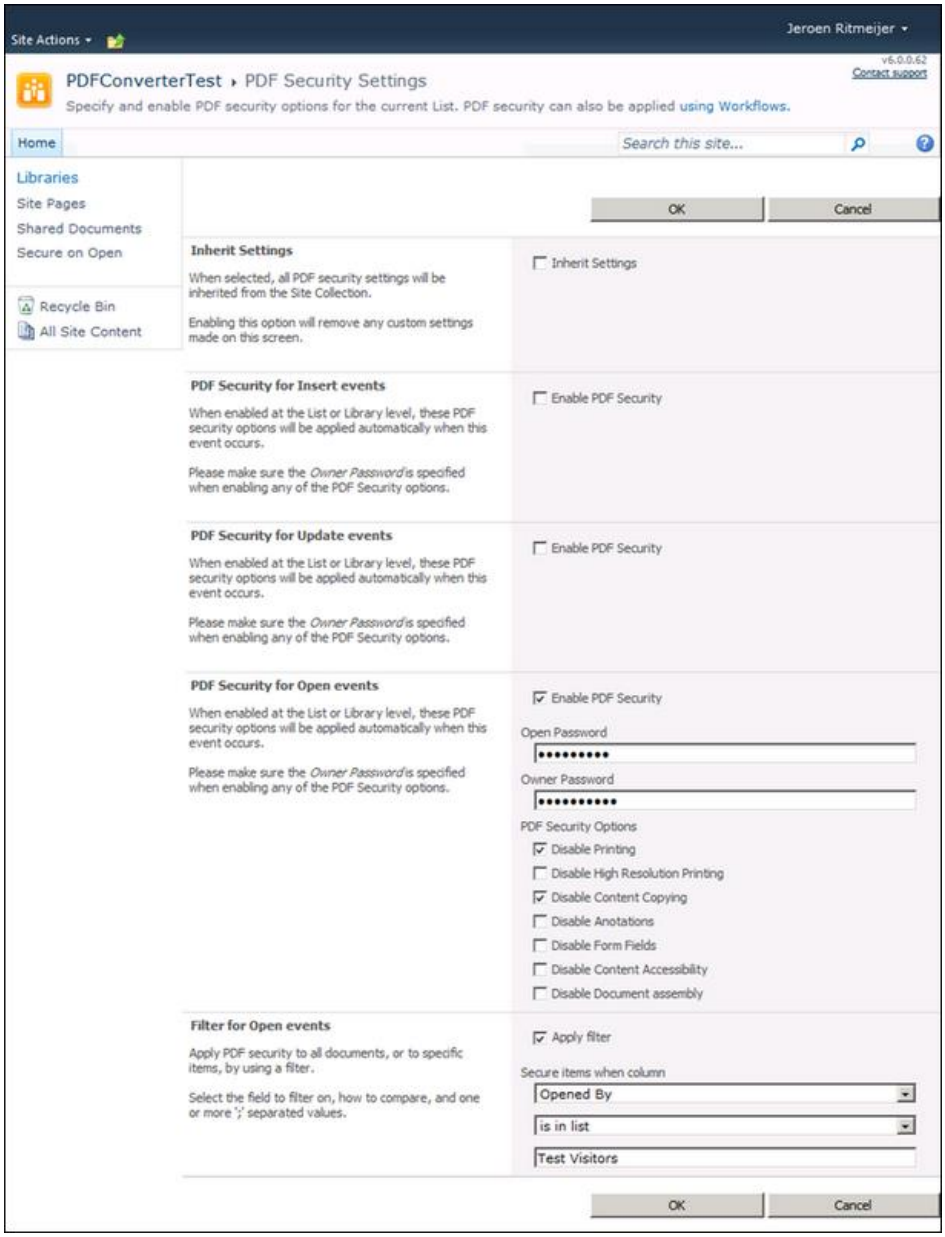
The facility explained in this section can be used for exactly that purpose. The latest version of this section, including user feedback, can be found on the Muhimbi Blog. Additional details about securing Office files can be found here.

The key features are as follows:

- Apply security to Word, Excel, PowerPoint and PDF Files.
- Apply security after user specific watermarks have been applied.
- Apply typical PDF Security including *Open Password, Owner Password, Prevent Printing, Prevent Copy, Prevent Document assembly*, etc.
- Allow filters to be specified and only apply security when a condition is met, e.g. a *Status* field is set to *Approved*, or the user that is accessing the document is in a specific group.
- Apply security to files in Document Libraries as well as files attached to individual list items.
- Works on all SharePoint 2007 and later versions.

Let's work through an example to show how easy it is to set this up.

1. By default the *Secure / Watermark on open* facility is disabled so use SharePoint Central Administration to enable the *Muhimbi PDF Converter - Automatic Document Processor* Feature at the relevant Web Application. Note that this is a Web Application Scoped Feature, not a Farm or Site Collection scoped one. You also need to enable the *Muhimbi PDF Converter - Automatic Document Processing User Interface* Feature at either the Web Application Level (to enable the screen on all Site Collections) or at the individual Site Collection level.
2. Once enabled, a new menu named *Security settings* can be found in the *Site Actions / Site Settings* screen as well as the *List Settings* screen on each individual List and Document Library. Default security settings can optionally be specified at the Site Collection level, which can then be inherited at the individual List or Library Level, which is displayed in the following screen.



3. As can be seen in the screenshot there are also options to enable security during *Insert* and *Update* events. However, the focus of this example is to *Secure On Open*. In this screenshot we have specified both an *Open* and an *Owner Password*. The *Owner Password* must be set when any of the *PDF Security Options* are selected, the *Open Password* is optional.
4. In the same screenshot we have also specified a filter to only secure documents when the person opening the file is in the *Test Visitors* SharePoint group. Please note that you can only use SharePoint Group names, not Windows Group names.

That is all there is to it. When a PDF or Office file is opened from the Document Library, and the user opening it is a member of the *Test Visitors* group, then security will be applied automatically to the file without modifying the original in the List or Document Library.

Please note that securing files this way is a *real-time* action and adds some overhead. If there is no need to apply security in combination with user specific watermarks, or based on a user specific filter, then we recommend applying security using a *SharePoint Designer Workflow* (See 14.1) or *Nintex* Workflow (See 14.2) the moment a file is created or modified.

Use the *Filtering out system users (e.g. Search indexer)* facility, described in section 13.5.5 to skip processing for certain accounts.

15 Carry out OCR (Optical Character Recognition)

The Muhimbi PDF Converter for SharePoint provides support for Optical Character Recognition (OCR), which can be used to convert image-based content such as scans or faxes into fully searchable and indexable PDFs. In addition, it can be used to recognise text and extract content from these images and use it for further processing. This can be useful to retrieve invoice numbers or other textual content from documents that conform to a particular template.

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.

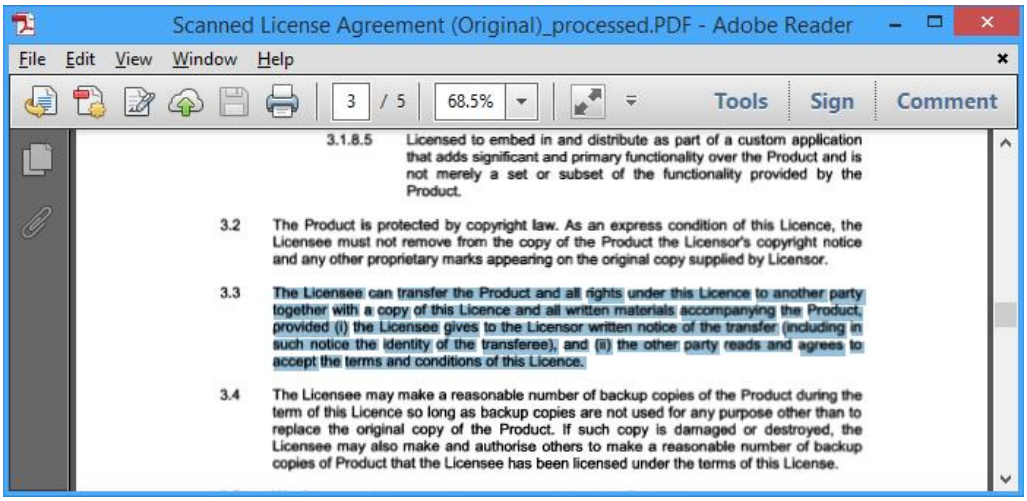
15.1 Convert Images & Scans to Searchable PDFs

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

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

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

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



Scanned Document with OCRed text selected

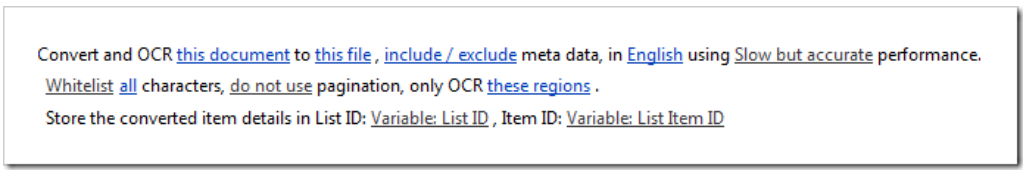
The key features are as follows:

- Server based solution, accessible via a modern Web Service interface (*Java, C#, Ruby, PHP* etc)
- Integrates with SharePoint Designer and Nintex workflows.
- 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*).
- 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.

Although very powerful, 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.

15.1.1 OCR files using SharePoint Designer workflows

It is possible to carry out OCR using our standard *Convert Document* workflow activity, but that requires knowledge of our XML syntax, which - although powerful - is less than user friendly. To make life easier we have created a separate Workflow Activity named *Convert to OCRed PDF.* This is what it looks like.



The workflow sentence is consistent with the other Muhimbi Workflow Activities and largely self-describing:

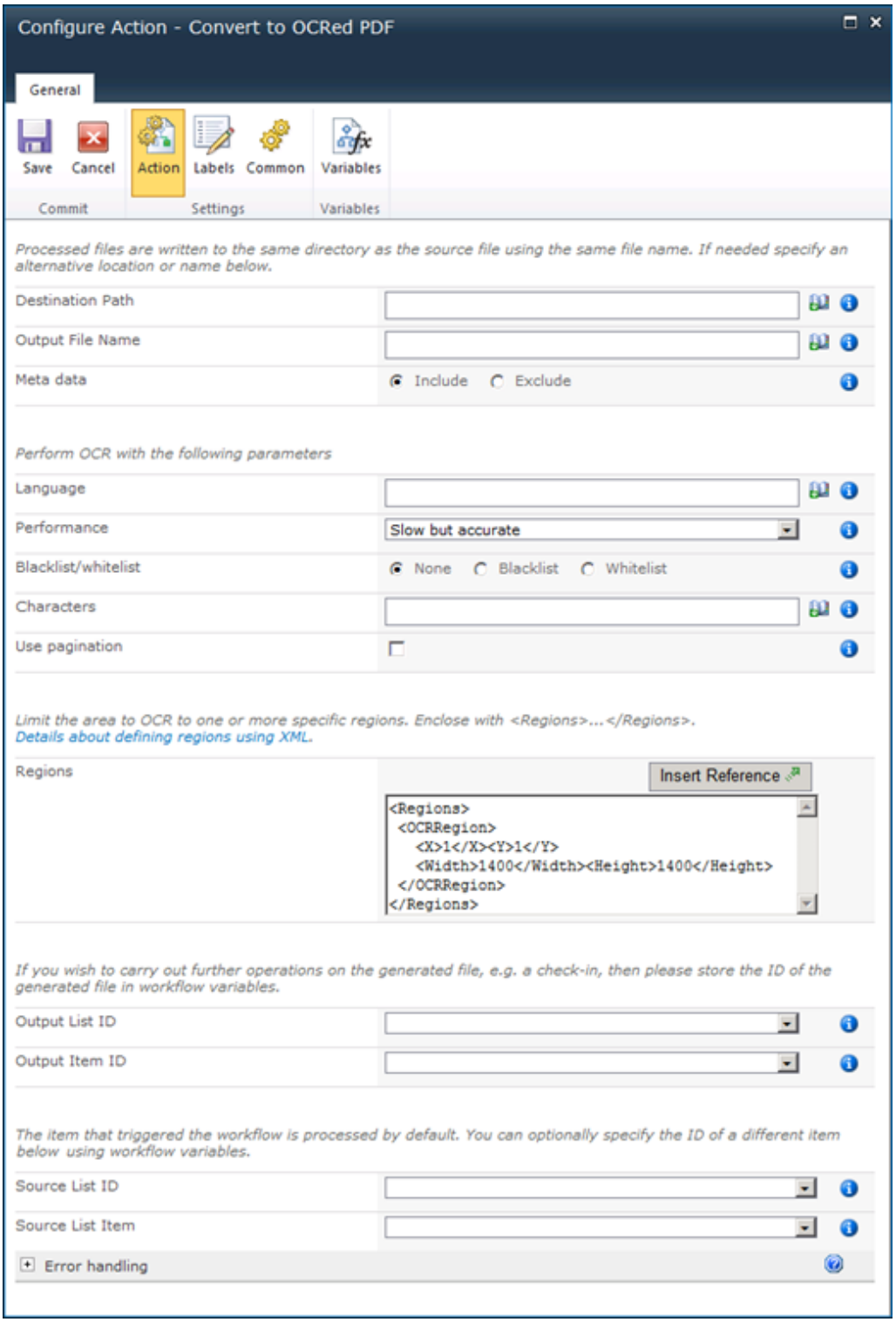
1. **this document:** The source document to *Convert and OCR*. For most workflows selecting *Current Item* will suffice, but some scenarios may require the look up of a different item.
2. **this file:** The name and location to write the generated file to. Leave this field empty to use the same location and name as the source file. Please

note that if your source file is already in PDF format then leaving this field empty will overwrite it. For details about how to specify paths to different libraries / site collections see *Appendix - Specifying path and file names*.

3. **include / exclude meta data**: Control if the source file's SharePoint meta-data is copied to the destination file.
4. **OCR language**: The language the source document is written in. It defaults to English, but we currently support *Arabic, Danish, German, English, Dutch, Finnish, French, Hebrew, Hungarian, Italian, Norwegian, Portuguese, Spanish* and *Swedish*.
5. **OCR Performance**: Specify the performance / accuracy of the OCR engine. It is recommended to leave this on the default *Slow but accurate* setting.
6. **Whitelist / Blacklist**: Control which characters are recognised. For example, limit recognition to numbers by whitelisting 1234567890. This prevents, for example, a 0 (zero) to be recognised as the letter o or O.
7. **Pagination**: In some specific cases a single image spans multiple pages. Enable *pagination* for those cases.
8. **Regions**: By default, the entire page is OCRed. To limit OCR to certain parts of a page, e.g., a header and/or footer, you can specify one or more regions using our XML syntax. Have a look at this blog post, but only use the part that starts with (and includes) <Regions>…</Regions>.
9. **List ID:** The ID of the list the processed file was written to. This can later in the workflow be used to perform additional tasks on the file such as a check-in or out.
10. **Item ID:** The ID of the processed file. Can be used with the List ID.

15.1.2 OCR files using Nintex Workflow

It is possible to carry out OCR using our standard *Convert Document* workflow activity, but that requires knowledge of our XML syntax, which - although powerful - is less than user friendly. To make life easier we have created a separate Workflow Activity named *Convert to OCRed PDF.* It is compatible with Nintex Workflow 2007-2019 and this is what it looks like.



For a comprehensive example and details about how to enable the Nintex Workflow integration see chapter 4 *Converting Documents using Nintex Workflow*.

The fields supported by this Workflow Activity are as follows:

1. **Destination Path:** The location to write the generated file to. Leave this field empty to use the same location as the source file. For details about how to specify paths to different libraries / site collections see *Appendix - Specifying path and file names.*
2. **Output File Name:** The name of the generated file. Leave this field empty to use the same name as the source file. Please note that if your source file is already in PDF format, and the Destination Path is the same as the Source Path, then leaving this field empty will overwrite it.
3. **Meta data**: Control if the source file's SharePoint meta-data is copied to the destination file.
4. **Language**: The language the source document is written in. It defaults to English, but we support *Arabic, Danish, German, English, Dutch, Finnish, French, Hebrew, Hungarian, Italian, Norwegian, Portuguese, Spanish* and *Swedish*.
5. **Performance**: Specify the performance / accuracy of the OCR engine. It is recommended to leave this on the default *Slow but accurate* setting.
6. **Whitelist / Blacklist**: Control which characters are recognised. For example, limit recognition to numbers by whitelisting 1234567890. This prevents, for example, a 0 (zero) to be recognised as the letter o or O.
7. **Pagination**: In some specific cases a single image spans multiple pages. Enable *pagination* for those cases.
8. **Regions**: By default, the entire page is OCRed. To limit OCR to certain parts of a page, e.g., a header and/or footer, you can specify one or more regions using our XML syntax. Have a look at this blog post, but only use the part that starts with (and includes) <Regions>…</Regions>.
9. **Output List ID:** If you wish to carry out further actions on the generated file, e.g., send it by email or perform a check-in, then you can optionally store the ID of the List the file was written to in a workflow variable of type *String*.
10. **PDF List Item ID:** Similarly, to *Output List ID,* the Item ID of the generated file can optionally be stored in a workflow variable of type *Item ID* (in SharePoint 2007) or *Integer* (in SharePoint 2010 and later versions).
11. **Source List ID & List Item:** The item that triggered the workflow is processed by default. You can optionally specify the ID of a different List and List Item using workflow variables. Please use the same data types as used by *Output List ID* and *Output List Item ID*.
12. **Error Handling:** Similar to the way some of Nintex' own Workflow Activities allow errors to be captured and evaluated by subsequent actions, all of Muhimbi's Workflow Activities allow the same. By default, this facility is disabled meaning that any error terminates the workflow.

15.1.3 OCR files using a web Service call

For a detailed example about how to use the Web Services interface to carry out OCR and generate fully searchable and indexable PDF files see this Knowledge Base article.

15.2 Extract text from image based content

A new OCR related facility has been added to the product with the introduction of version 7.2. This new facility allows text on (part of) a page to be recognised and returned to the workflow for further processing.

A common use for this is to extract a particular area of text from documents that all use a common template or layout. For example, if a reference number can always be found at the top right corner of scanned documents, then that text can be extracted and stored in a SharePoint column from where it can be included in searches or be used in further workflow steps.

15.2.1 Extract text using OCR and SharePoint Designer Workflows

Once the PDF Converter is installed you will find a number of new Workflow Activities in SharePoint Designer. One of these activities is named *Extract Text using OCR* and looks as follows.

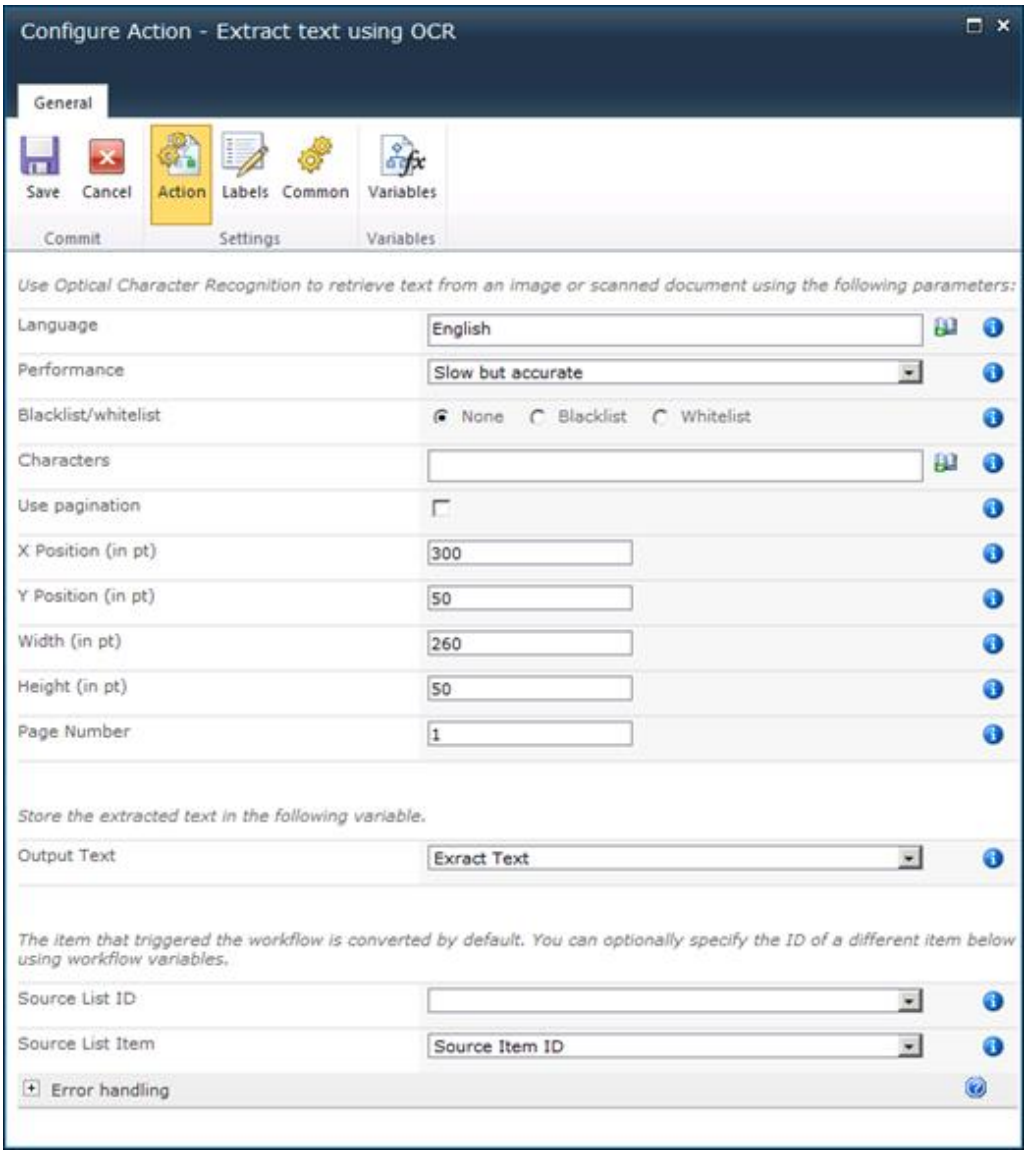
The workflow sentence is consistent with the other Muhimbi Workflow Activities and largely self-describing:

1. **this document:** The source document to OCR and extract text from. For most workflows selecting *Current Item* will suffice, but some scenarios may require the look up of a different item.
2. **OCR language**: The language the source document is written in. It defaults to English, but we support *Arabic, Danish, German, English, Dutch, Finnish, French, Hebrew, Hungarian, Italian, Norwegian, Portuguese, Spanish* and *Swedish*.
3. **OCR Performance**: Specify the performance / accuracy of the OCR engine. It is recommended to leave this on the default *Slow but accurate* setting.
4. **Whitelist / Blacklist**: Control which characters are recognised. For example, limit recognition to numbers by whitelisting 1234567890. This prevents, for example, a 0 (zero) to be recognised as the letter o or O.
5. **Pagination**: In some specific cases a single image spans multiple pages. Enable *pagination* for those cases.

6. **Region**: Specify the *x, y, width* and *height* of the region to retrieve text from. The unit of measure (UOM) is *pt*, 1/72nd of an inch. When extracting text from non-PDF files, e.g., a TIFF or PNG, then please take into account that internally the image is first converted to PDF, which may add margins around the image but guarantees that a single – unified - UOM is used across all file formats. If you are not sure how internal conversion affects the dimensions of your image or scan, then use our software to convert the file to PDF and open it in a PDF reader.
7. **Page:** By default, text is extracted from all pages and concatenated. To extract the text from a specific page, specify the page number in this field.
8. **Result:** The recognised text will be stored in this variable (type String)

15.2.2 Extract text using OCR and Nintex Workflow

The new *Extract text using OCR* activity is compatible with Nintex Workflow 2007 and later versions.



For a comprehensive example and details about how to enable the Nintex Workflow integration see chapter 4 *Converting Documents using Nintex Workflow*.

The fields supported by this Workflow Activity are as follows:

1. **Language**: The language the source document is written in. It defaults to English, but we support *Arabic, Danish, German, English, Dutch, Finnish, French, Hebrew, Hungarian, Italian, Norwegian, Portuguese, Spanish* and *Swedish*.
2. **Performance**: Specify the performance / accuracy of the OCR engine. It is recommended to leave this on the default *Slow but accurate* setting.
3. **Whitelist / Blacklist**: Control which characters are recognised. For example, limit recognition to numbers by whitelisting 1234567890. This prevents, for example, a 0 (zero) to be recognised as the letter o or O.
4. **Pagination**: In some specific cases a single image spans multiple pages. Enable *pagination* for those cases.
5. **Region**: Specify the *x, y, width* and *height* of the region to retrieve text from. The unit of measure (UOM) is *pt*, 1/72nd of an inch. When extracting text from non-PDF files, e.g., a TIFF or PNG, then please take into account that internally the image is first converted to PDF, which may add margins around the image but guarantees that a single - unified - UOM is used across all file formats. If you are not sure how internal conversion affects the dimensions of your image or scan, then use our software to convert the file to PDF and open it in a PDF reader.
6. **Page Number:** By default, text is extracted from all pages and concatenated. To extract the text from a specific page, specify the page number in this field.
7. **Output Text:** The recognised text will be stored in this variable (type String).
8. **Source List ID & List Item:** The item that triggered the workflow is processed by default. You can optionally specify the ID of a different List and List Item using workflow variables. Please use data type *string* for the List ID workflow variable. For the Item ID use type *Item ID* (in SharePoint 2007) or *Integer* (in SharePoint 2010 and later versions)
9. **Error Handling:** Similar to the way some of Nintex' own Workflow Activities allow errors to be captured and evaluated by subsequent actions, all of Muhimbi's Workflow Activities allow the same. By default, this facility is disabled meaning that any error terminates the workflow.

16 Copying Metadata

Some of the product's Workflow Activities and conversion screens provide the ability to copy-meta as part of the conversion process. This works well, but it is an 'all or nothing' approach.

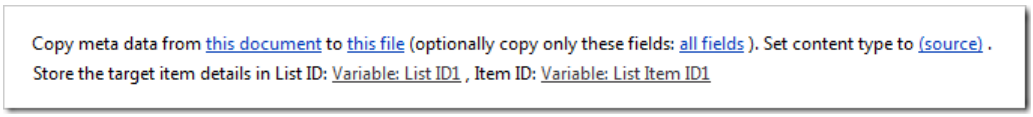
To provide users with a more flexible way of copying meta-data, and setting an item's content type, we provide a separate SharePoint Designer and Nintex Workflow Activity to deal with this kind of operation.

The high-level functionality is as follows:

- Standalone Workflow Activity that can be used in combination with any of Muhimbi's Workflow Activities, or without them.
- Copy all meta-data or only selected fields.
- Copy meta-data to files in different folders or site collections.
- Change the content type to either the source file's, destination file's, the default content type for the library or a specific named content type.
- Copy content of *Author*, *Created* and *Modified* fields by explicitly specifying these field names. This information is not copied when the default 'copy all fields' option is enabled. It is not possible to copy the *Editor* field as that is always overwritten by the workflow.

16.1 Copying Metadata from SharePoint Designer Workflows

To insert the *Copy Meta-Data* Activity into a workflow, click the *Action* button in the SharePoint Designer Workflow editor and select *Copy Meta Data*. This inserts the following *workflow sentence*.

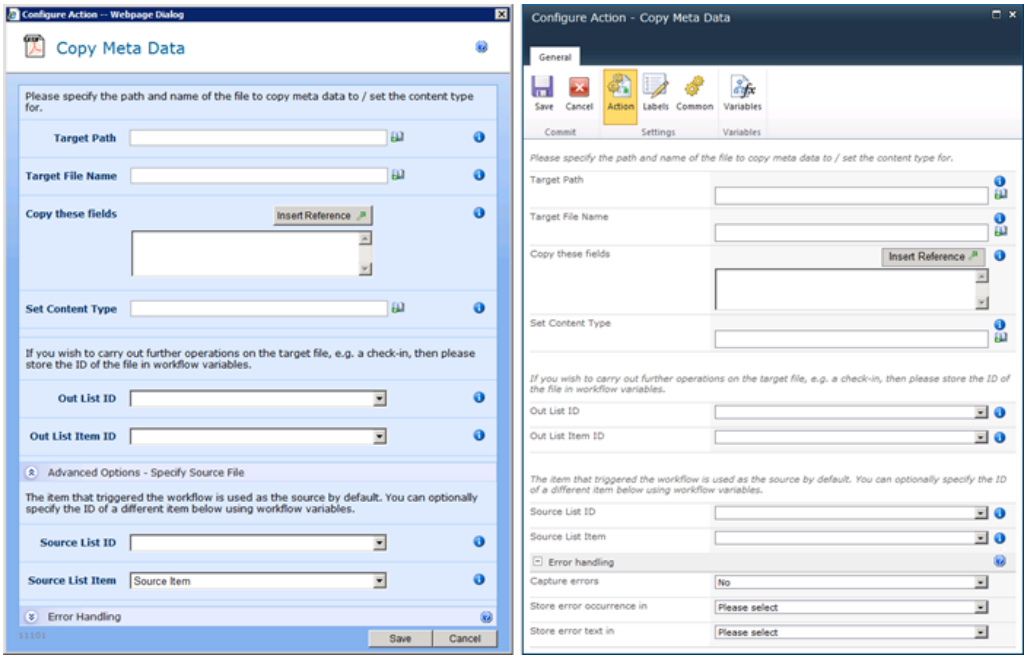


The parameters are largely self-describing and use the same format as the other Workflow Activities.

- **This document:** The source document to copy the meta-data from. For most workflows selecting *Current Item* will suffice, but some custom scenarios (e.g., Site workflows) may require the look up of a different item.
- **This File:** The path and file name to copy the meta-data to. *Please make sure that the path does not include the host name, e.g., 'http://your site/…'. For more details see* Appendix - Specifying path and file names.
- **Fields:** By default, the content of all fields is copied to the destination file. However, if you wish to copy only specific fields then the field names can be specified in this list. You can separate fields using line breaks, ',' or ';' and you can use both *internal* and *display* field names.

- **Content type:** While copying meta-data you have full control over the content type of the destination file. The following options can be specified:
  - **(source)**: The content type of the source file is copied to the destination file. *Please include the round brackets.*
  - **(target)**: The content type of the destination file is not modified and remains what it was before the copy operation. *Please include the round brackets.*
  - **(default)**: The default content type for the document library is applied to the destination file. *Please include the round brackets.*
  - **Name of Content type**: The destination file is set to a specific, named, content type. *Please **do not** use round brackets around the name of the content type.*
- **Parameter 'List ID'**: The ID of the file the meta-data was copied to. This can later in the workflow be used to perform additional tasks on the file such as performing a check-in or out.
- **Parameter 'List Item ID'**: The ID of the list that holds the file that the meta-data was copied to.

16.2 Copying Metadata using a Nintex Workflow



For a full example of how to use Muhimbi's Workflow Activities using Nintex Workflow see Chapter 4 *Converting Documents using Nintex Workflow*. The fields supported by this Workflow Activity are as follows:

- **Target Path:** Enter the path of the target file to copy meta-data to, either:
  - Leave empty to use the same directory as the source file.
  - A relative path to a sub site / document library / folder.

- An absolute path to a different site collection.
  Please make sure the path does not include the host name, e.g., 'http://your site/…'. For more details see *Appendix - Specifying path and file names.*

- **Target File Name:** The name of the file to copy the meta-data to. Leave empty to use the source file, which can be useful if you just wish to set the content type of a file.
- **Copy these fields:** By default, the content of all fields is copied to the destination file. However, if you wish to copy only specific fields then the field names can be specified in this list. You can separate fields using line breaks, ',' or ';' and you can use both *internal* and *display* field names.
- **Set Content type:** While copying meta-data you have full control over the content type of the destination file. The following options can be specified:
  - **(source)**: The content type of the source file is copied to the destination file. *Please include the round brackets.*
  - **(target)**: The content type of the destination file is not modified and remains what it was before the copy operation. *Please include the round brackets.*
  - **(default)**: The default content type for the document library is applied to the destination file. *Please include the round brackets.*
  - **Name of Content type**: The destination file is set to a specific, named, content type. *Please **do not** use round brackets around the name of the content type.*
- **Out List ID:** If you wish to carry out further actions on the target file, e.g., perform a check-in, then you can optionally write the ID of the List that holds the target file to a workflow variable of type *String*.
- **Out List Item ID:** Similarly, to *Out List ID,* the Item ID of the target file can optionally be written to a workflow variable of type *Item ID* (in SharePoint 2007) or *Integer* (in SharePoint 2010 and later versions).
- **Source List ID & List Item:** The item that triggered the workflow is used as the source item by default. You can optionally specify the ID of a different List and List Item using workflow variables. Please use the same data types as used by *Out List ID* and *Out List Item ID*.
- **Error Handling:** Similar to the way some of Nintex' own Workflow Activities allow errors to be captured and evaluated by subsequent actions, all of Muhimbi's Workflow Activities allow the same. By default, this facility is disabled meaning that any error terminates the workflow.

17 Building a Table Of Contents

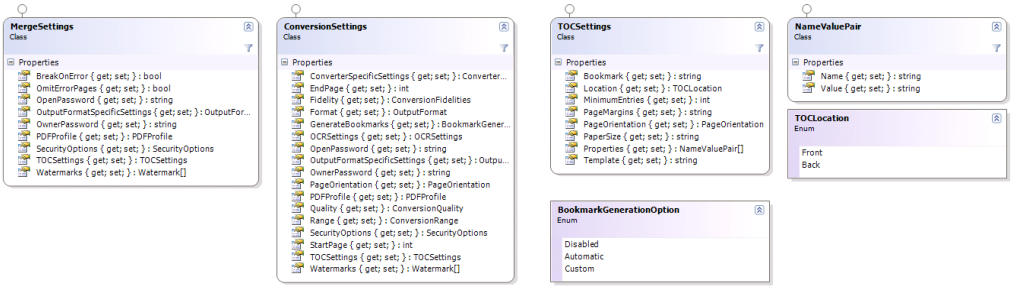
One of the more popular features provided by the PDF Converter is the ability to convert and merge multiple documents into a single PDF, all in one operation (See chapter 10 for details). Although this facility works very well, and even includes the ability to generate PDF bookmarks to aid with navigation inside the merged document, a common request is to add a full Table of Contents (TOC) to the merged document as well.

In this chapter the focus is on generating a TOC via our API (See our separate Developer Guide). However, if you wish to use this functionality from a SharePoint Workflow, then please continue reading as well. We might have to get a little bit technical, but once you're familiar with the concepts, you can use our XML based workflow syntax – as described in *Appendix - Override default conversion settings* - to generate a table of contents from SharePoint Designer, Nintex and K2 based workflows.

A copy of this chapter can also be found on our Blog.

17.1 Object Model

The classes relevant to dealing with TOCs are as follows



- **MergeSettings:** When merging multiple files and generating a single table of contents, follow the normal procedure for merging files (sample code) and populate the *MergeSettings.TOCSettings* property as per the sample code below.
- **ConversionSettings:** To generate a table of contents for a single document (so not as part of a merge operation), follow the normal procedure for converting or processing a single file (sample code) and populate *ConversionSettings.TOCSettings* as per the sample code below.
- **TOCSettings:** All settings related to the generation of the TOC can be found in this class. The available properties are as follows:
  - **Bookmark:** The TOC itself can have its own PDF bookmark to aid with navigation. Specify the text in this property.
  - **Location:** TOCs can be added to the *Front* or *Back* of the document. Enter the relevant option here.

- **MinimumEntries:** For certain, simple, documents that only have one or 2 bookmarks, it may not make sense to add a table of contents. Use this property to specify the minimum number of entries before a TOC is generated. The default value is '0', which will always create a TOC regardless of the number of entries.
- **PageMargins:** Page margins in the format set out below. It defaults to a uniform half inch margin.

  "#{dim}" - for a uniform margin or
  "#{dim},#{dim},#{dim},#{dim}" - for individual margins
  where
  - # is numeric value
  - {dim} is dimension. Either empty (meaning inches) or "mm", "in", "in.", "inch" or "inches".
- **PageOrientation:** The orientation used by the TOC. *Portrait*, *Landscape* or *Default*. The *Default* option uses the same orientation as the page following (or preceding) the TOC depending on the value specified in *Location*.
- **PaperSize:** A *named* paper size such as *A4* or *Letter* (See MSDN) or a custom size in "{width}{dim}{sep}{height}{dim}" format where:

  - {width} and {height} are numerical values (please use a colon '.' as the decimal separator) .
  - {dim} is the dimension which can be 'mm', 'in.' or 'inches'. (It defaults to inches when nothing is specified)
  - {sep} separates the width and the height, either 'by', comma (,) or the letter 'x' Example: "8.5 in. by 6 in."
- **Properties:** Optional properties to pass to the XSL template for display or processing purposes. For details see below.
- **Template:** The XSL template (See 17.3) to use for formatting purposes. This can either be a string containing all the XSL, a path - ***local*** to the server running the conversion service - to the location of the XSL file, or a URL to the XSL file on a web (or SharePoint) server.

- **NameValuePair:** A single value that can be passed into the XSL using *TOCSettings.Properties*.
- **TOCLocation:** Used by *TOCSettings.Location* to determine where the TOC should go.
- **BookmarkGenerationOption:** As explained in *XML Source Data* (17.2), the TOC system is based on the content and structure of PDF Bookmarks. It is therefore essential that during the conversion of the source documents *ConversionSettings.GenerateBookmarks* is set to *Automatic*.

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

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

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

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

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

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

17.2 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 13).

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.

17.3 XSL Transformation

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

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <xsl:stylesheet version="1.0"
3                  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
4                  xmlns:msxsl="urn:schemas-microsoft-com:xslt"
5                  exclude-result-prefixes="msxsl">
6
7    <xsl:output method="html" indent="yes"/>
8
9    <xsl:template match="/toc">
10     <html>
11       <head>
12         <style type="text/css">
13           ul.toc
```

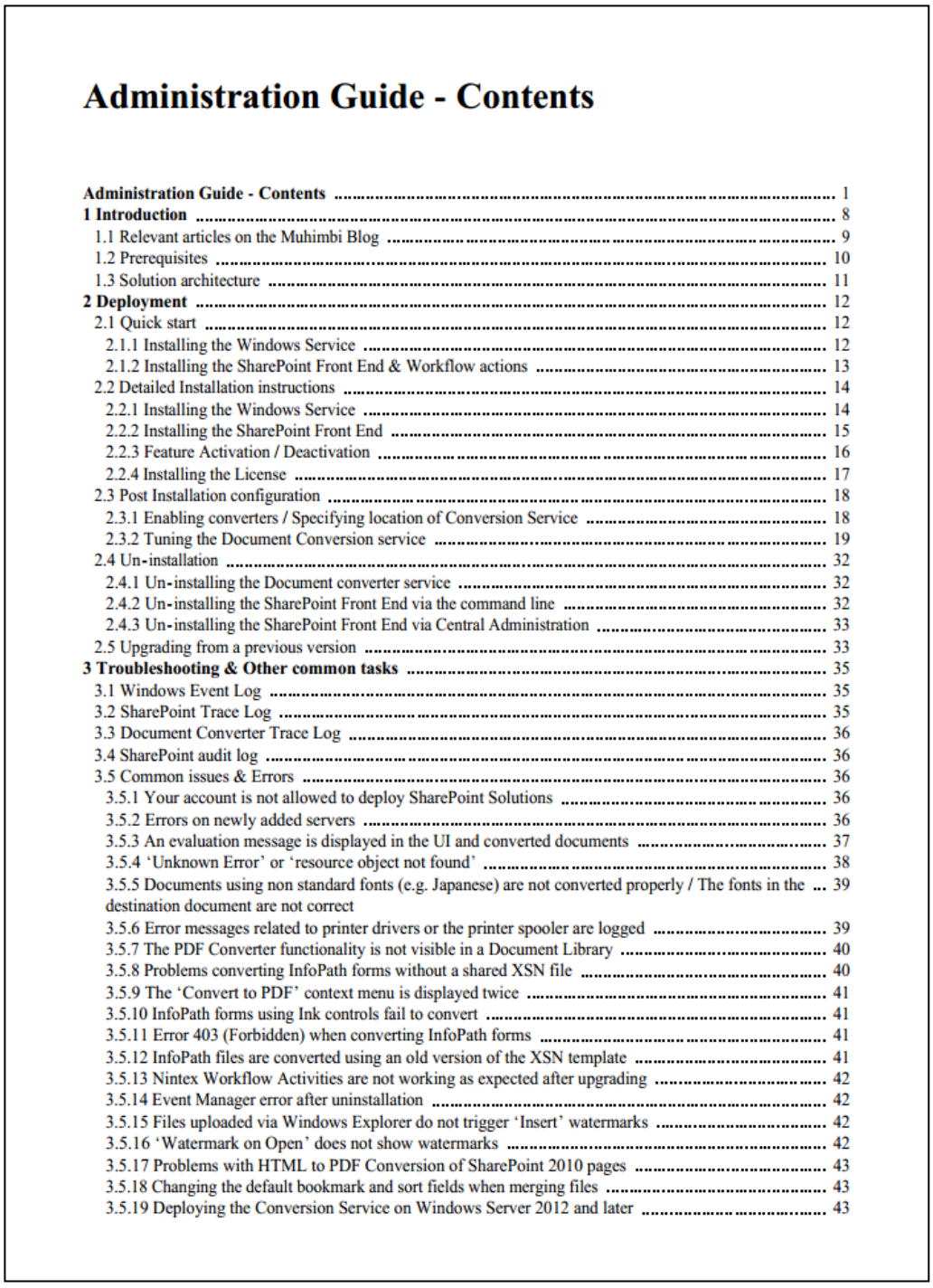
```
14          {
15              margin: 0;
16              padding: 0;
17              list-style: none;
18          }
19          ol.toc
20          {
21              margin: 0;
22              padding: 0;
23              margin-left: 10px;
24              list-style: none;
25          }
26          ul.toc li
27          {
28              clear: both;
29              overflow: hidden;
30          }
31          ol.toc li
32          {
33              overflow: hidden;
34          }
35          span.title
36          {
37              float: left;
38              padding-right: 4px;
39          }
40          span.page
41          {
42              float: right;
43              padding-left: 4px;
44          }
45          span.dots
46          {
47              font-size: 0px;
48              width:100%;
49              border-bottom: 2px dotted black;
50          }
51          a.toc
52          {
53            text-decoration: none;
54            color: #000;
55          }
56        </style>
57      </head>
58      <body>
59        <h1>
60          <xsl:value-of select="properties/property[@name='title']"/>
61        </h1>
62        <br/>
63        <br/>
64        <xsl:apply-templates/>
65      </body>
66    </html>
67  </xsl:template>
68
69  <xsl:template match="topics">
70    <ul class="toc">
71      <xsl:apply-templates/>
72    </ul>
73  </xsl:template>
74
75  <!-- Empty template so properties are not appearing -->
76  <xsl:template match="properties"></xsl:template>
77
78  <xsl:template match="topic[@level='0']">
79    <li>
80      <xsl:element name="a">
81        <xsl:attribute name="href">
82          <xsl:value-of select="@target"/>
83        </xsl:attribute>
84        <xsl:attribute name="class">toc</xsl:attribute>
```

```
85          <span class="title" style="font-weight: 900;">
86            <xsl:value-of select="@title"/>
87          </span>
88          <span class="page">
89            <xsl:value-of select="@page"/>
90          </span>
91          <span class="dots"></span>
92        </xsl:element>
93      </li>
94      <ol class="toc">
95        <xsl:apply-templates/>
96      </ol>
97    </xsl:template>
98
99    <xsl:template match="topic">
100     <li>
101       <xsl:element name="a">
102         <xsl:attribute name="href">
103           <xsl:value-of select="@target"/>
104         </xsl:attribute>
105         <xsl:attribute name="class">toc</xsl:attribute>
106         <span class="title">
107           <xsl:value-of select="@title"/>
108         </span>
109         <span class="page">
110           <xsl:value-of select="@page"/>
111         </span>
112         <span class="dots"></span>
113       </xsl:element>
114     </li>
115     <ol class="toc">
116       <xsl:apply-templates/>
117     </ol>
118   </xsl:template>
119
120 </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.

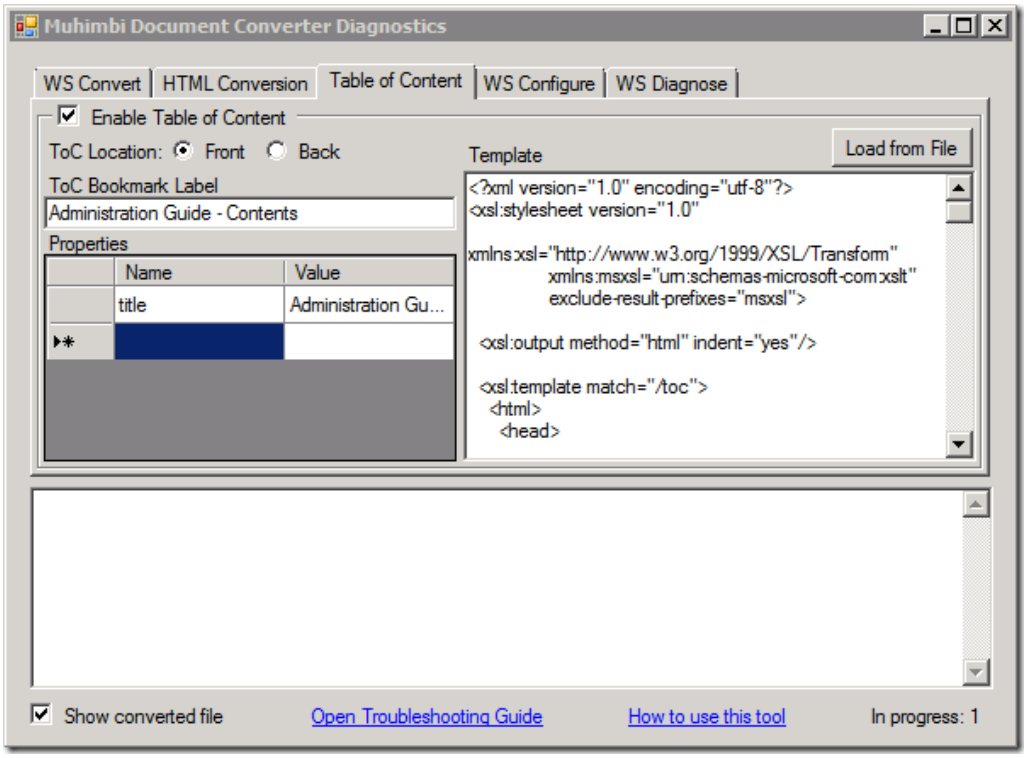
If your experience with XML and XSL is limited, then we recommend using the XSL sample provided above. As can be seen in the following screenshot, the results look very good.



Administration Guide - Contents

17.4 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 TOC facility. While this might be merely a handy test tool, not the official user interface for the TOC facility, it can be incredibly helpful in quickly testing various XSL template designs before integrating them into your solution.



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

17.5 Generating a TOC from a SharePoint Workflow

The previous sections explain the underlying TOC mechanism in detail, but the example uses the Muhimbi API, which is of little use to SharePoint Workflow developers. For an example about how to apply the same concept, but generate the TOC from a SharePoint Workflow, see the relevant example in *Appendix - Override default conversion settings*.

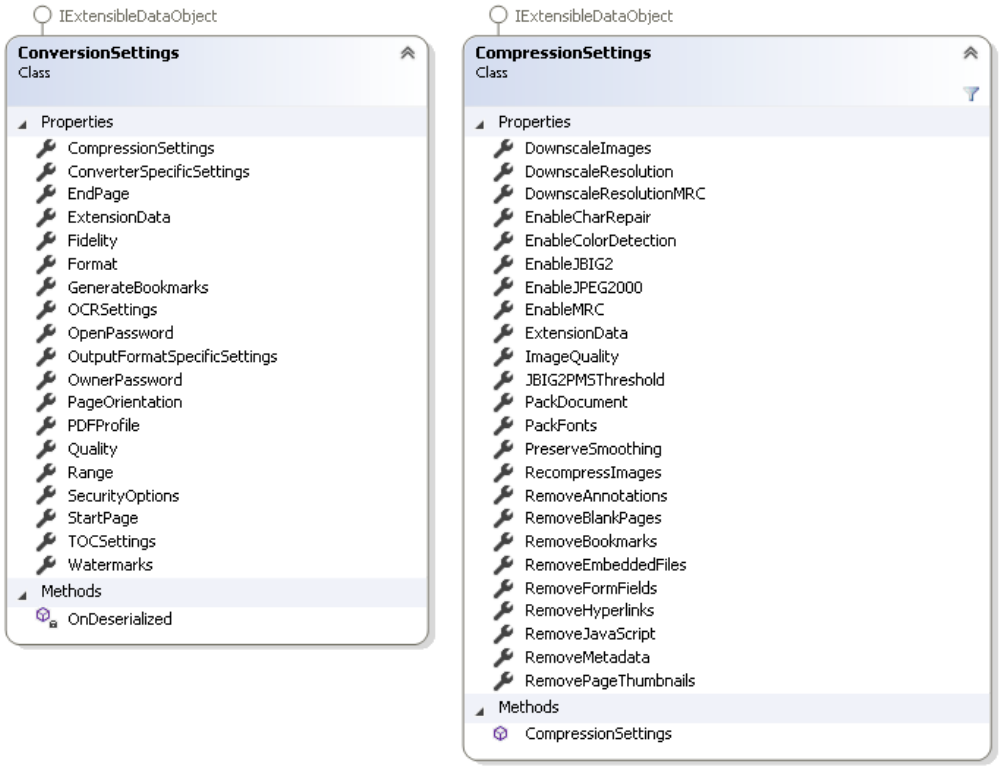
18 Compressing output files

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.

18.1 Object Model

The classes relevant to dealing with compressing files are as follows:



Other objects (such as OpenSettings) may be relevant to your use case.

18.2 Specifying the options

The following options are available:

Option	Description
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
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 is 200
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 to the following resolution (DPI)
DownscaleResolution	Resolution used to downscale images. Default 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 is 85

* Tri-state: Default/True/False. Values ARE case sensitive.

18.3 Compression using Override XML

The following Override XML undertakes compression only. The CompressionSettings node can be added to an existing ConversionSettings node to combine all other operations available via Override XML.

```
<Override>
    <ConversionSettings>
        <CompressionSettings>
          <RemoveAnnotations>Default</RemoveAnnotations>
          <RemoveBlankPages>Default</RemoveBlankPages>
          <RemoveBookmarks>Default</RemoveBookmarks>
          <RemoveEmbeddedFiles>Default</RemoveEmbeddedFiles>
          <RemoveFormFields>Default</RemoveFormFields>
          <RemoveJavaScript>Default</RemoveJavaScript>
          <RemoveMetadata>Default</RemoveMetadata>
          <RemovePageThumbnails>Default</RemovePageThumbnails>
          <PackFonts>Default</PackFonts>
          <PackDocument>Default</PackDocument>
          <RecompressImages>Default</RecompressImages>
          <EnableMRC>Default</EnableMRC>
          <PreserveSmoothing>Default</PreserveSmoothing>
          <DownscaleImages>Default</DownscaleImages>
          <EnableColorDetection>Default</EnableColorDetection>
          <EnableCharRepair>Default</EnableCharRepair>
          <EnableJPEG2000>Default</EnableJPEG2000>
          <EnableJBIG2>Default</EnableJBIG2>
          <JBIG2PMSThreshold>85</JBIG2PMSThreshold>
         <DownscaleResolution>150</DownscaleResolution>
          <DownscaleResolutionMRC>150</DownscaleResolutionMRC>
          <ImageQuality>ImageQualityDefault</ImageQuality>
        </CompressionSettings>
    </ConversionSettings>
</Override>
```

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.

19 Troubleshooting & Other common tasks

Although the PDF Converter is a user friendly, robust and intuitive application, 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.

19.1 The PDF Converter functionality is not available

When the PDF Conversion functionality is not displayed on a file's context menu, or in the Actions menu of the document library, then this may be due to the following:

1. The PDF Converter has either not been installed on the server or has not been activated.

   Please ask your SharePoint administrator to enable the *Muhimbi PDF Converter* on your Web Application.

2. The PDF Converter has been installed, but you are using a non-standard Document or Forms Library.

   Please ask your SharePoint administrator to consult section 3.5.7 of the PDF Converter Administration Guide.

19.2 Converting documents takes a very long time

In general, the PDF Converter is a very fast application. However, depending on the size and complexity of the documents that are being converted, the conversion process may take some time.

When converting multiple large documents in one go, you may want to consider converting them in batches as the maximum processing time – via the SharePoint user interface - is 30 minutes, after which an error will be displayed.

19.3 The PDF file does not look exactly the same as the source file

Although the PDF Converter for SharePoint 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 does not deal well with certain fonts, even when these fonts have been installed on the server. Try using a different font or create a separate InfoPath *Print View*.

19.4 How can I see who has converted a document?

Providing SharePoint Auditing is enabled, specifically the *Copy* and *View* audit types, an audit entry is written for each converted document.

The entry is written to the audit log for the Site Collection the source file belongs to. The following XML is added to each audit entry:

```
<DocumentConversion>
  <Source>
    <Version><Major>{0}</Major><Minor>{1}</Minor></Version>
  </Source>
  <Destination>
    <Url>{2}</Url>
   <Version><Major>{3}</Major><Minor>{4}</Minor></Version>
  </Destination>
</DocumentConversion>
```

Where

1. {0} and {1} are the version number of the source document,
2. {2} is the URL to the converted PDF file
3. {3} and {4} are the version number of the destination document.

To control auditing and view the SharePoint audit logs use Muhimbi SharePoint Audit, available at https://www.muhimbi.com/.

19.5 An evaluation message is displayed in the UI and converted documents

When an *evaluation* message is displayed on each screen and in each converted document then something may be wrong with your license or your license has not been installed. Please contact your SharePoint administrator who will be able to find additional information in the Administration Guide for this product.

19.6 InfoPath Forms fail to convert

When InfoPath documents fail to convert then please contact your SharePoint administrator and ask him to consult section 3.5 in the Administration Guide.

19.7 Converting file formats that are not supported

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*.

19.8 Nintex Workflow Activities are not working as expected after upgrading

If you encounter any Nintex Workflow related issues after upgrading to a new version of the Muhimbi PDF Converter for SharePoint then make sure the client side browser cache is cleared. If problems persist then please ask your SharePoint Administrator to review section 3.5.13 in the Administration Guide.

Also make sure the Muhimbi Nintex Workflow Feature is enabled on the relevant Web Application.

19.9 Files uploaded via Windows Explorer do not trigger 'Insert' watermarks

When using the watermarking facilities to automatically apply a watermark when a new file is uploaded then the appropriate event is not triggered by SharePoint 2010 when uploading the file using Windows File Explorer. This is related to a bug in SharePoint, the *Update* event is triggered instead.

As a workaround either make sure all files are uploaded using the web browser or apply watermarks to new files using a SharePoint Designer Workflow, Visual Studio Workflow or Nintex Workflow.

19.10 'Watermark on Open' does not show watermarks

If the *Watermark on Open* facility is used and watermarks do not show up when opening a PDF file then please check the following:

1. If the PDF File was previously opened by a user before this facility was enabled, the browser may cache the previous version of the document. To solve this problem clear the browser cache.
2. The Web Application scoped Feature named '*Muhimbi PDF Converter - Automatic Document Processor*' must be enabled. For details please ask your SharePoint administrator to enable the appropriate feature as described in section 2.2.3 of the Administration Guide.

19.11 Changing the default merge bookmark and sort fields

The PDF Converter ships with a powerful facility that allows multiple files to be merged together. It even allows fields to be selected to sort files by and to use as PDF bookmarks. However, the default options (sort by *Modified*, use *Title* as PDF bookmark) may not be suitable for your purposes. For details on how to change this behaviour see section 3.5.18 in the Administration Guide.

Appendix - Web Services Object Model

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

As outlined in the image below, the web service contains 3 main methods:

- **Convert:** Convert the file in the *sourceFile* byte array using the specified *openOptions* and *conversionSettings*. The generated PDF or XPS file is returned as a byte array as well.
- **GetConfiguration:** Retrieve information about which converters are supported and the associated file extensions. Consider calling this service once to retrieve a list of valid file extensions and check if a file is supported before it is 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.

The *ApplySecurity*, *ApplyWatermark* and *ProcessChanges* methods are identical at this moment in time and are provided for convenience only. They all take 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 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

A full discussion of the entire object model is out of the scope of this document. For full details see the PDF Converter Services User & Developer Guide as well as this blog post.

Appendix - Merge codes

Field Name	Description	Available from				
		Web Service	Work flows	WM on Open	WM on Insert[20]	WM on Update[20]
{LONG_DATE}	Long version of the date, e.g. *18 April 2011*.	V	V	V	V	V
{LONG_TIME}	Long version of the current time, e.g. *12:35:48*.	V	V	V	V	V
{DATE}	Short version of the date, e.g. *7/03/2011*.	V	V	V	V	V
{TIME}	Short version of the current time, e.g. *12:35*.	V	V	V	V	V
{PAGE}	The number of the current page in the PDF or PPTX file. This value is automatically updated for every page. (Not supported for DOCX and XLSX)	V	V	V	V	V
{NUMPAGES}	The number of pages in the PDF or PPTX file.	V	V	V	V	V
Any column name	Any SharePoint column / field defined on the list such as {Title}, {Author}. Please use (case sensitive) internal field names. A full list can be found at https://aarebrot.net/blog/2008/07/frodes-awesome-list-of-sharepoint-column-field-ids-for-sharepoint-2007/		*Using workflow fields*	V	V (Lists only)	V
{HTTP_HOST}	Returns the name of the Web server. This may or may not be the same as SERVER_NAME depending on type of name resolution you are using on your Web server (IP address, host header).			V		
{HTTP_REFERER}	Returns a string that contains the URL of the page that referred the request to the current page using an HTML <A> tag. Note that the URL is the one that the user typed into the browser address bar, which may not include the name of a default document.			V		
{HTTP_URL}	Returns the raw, encoded URL, for example, "/vdir/default.asp?querystring".			V		
{HTTP_USER_AGENT}	Returns a string describing the browser that sent the request.			V		
{LOGON_USER}	The Windows account that the user is impersonating while connected to your Web server. Use REMOTE_USER to view the raw user name that is contained in the request header.			V		
{REMOTE_ADDR}	The IP address of the remote host (identifying the user) that is making the request.			V		
{REMOTE_HOST}	The name of the host that is making the request. If the server does not have this information, it will set REMOTE_ADDR and leave this empty.			V		
{REMOTE_USER}	The name of the user as it is derived from the authorization header sent by the client, before the user name is mapped to a Windows account. If you have an authentication filter installed on your Web server that maps incoming users to accounts, use LOGON_USER to view the mapped user name.			V		
{SERVER_NAME}	The server's host name, DNS alias, or IP address as it would appear in self-referencing URLs.			V		
{URL}	Gives the base portion of the URL, without any querystring or extra path information, for example, "/vdir/default.asp".			V		

[20] Due to limitations in SharePoint 2007, automatic watermarking for Insert and Update events are not supported on Document Libraries. They are supported on Lists and in SharePoint 2010. Use Workflows as an alternative (see 13.1)

Field Name	Description	Available from				
		Web Service	Work flows	WM on Open	WM on Insert[20]	WM on Update[20]
{USER_NAME}	The user's name, if available (requires 7.2+).			V		
{USER_EMAIL}	The user's email, if available (requires 8.0+).			V		

Appendix - Override default conversion settings

The *Muhimbi PDF Converter for SharePoint* is based on an extremely flexible central conversion engine. This engine supports many more options than can realistically be displayed in the limited space available in a SharePoint Designer or Nintex Workflow Activity. As a result, the software defaults to the most common options.

Although it has always been possible to override these default settings (See the *Administration Guide*, *section 2.3.2*), those changes are global and affect all operations across the SharePoint Farm. To improve this situation version 6.1 introduces a new facility that allows the default settings to be overridden on a request-by-request basis.

The *Convert Document* workflow activity (See chapter 9 *Cross-Converting between document types*) has been extended and the previously reserved *Optional Parameters* field can now be used to specify 'override settings' using XML based syntax.

Although very powerful, the XML syntax and possible values for each XML Element may not immediately be obvious to all users. For full information see the class diagrams and details of the *OpenOptions* and *ConversionSettings* classes in the Developer Guide.

The latest version of this Appendix can be found in this blog post.

The values that can be overridden are as follows.

```
<Override>

  <OpenOptions>
    <AllowExternalConnections>false</AllowExternalConnections>
    <AllowMacros>None</AllowMacros>
    <FileExtension></FileExtension>
    <OriginalFileName></OriginalFileName>
    <UserName></UserName>
    <Password></Password>
    <RefreshContent>true</RefreshContent>
  </OpenOptions>

  <ConversionSettings>

    <ConverterSpecificSettings type="ConverterSpecificSettings_WordProcessing">
      <ProcessDocumentTemplate>false</ProcessDocumentTemplate>
      <RevisionsAndCommentsMarkupMode>InLine</RevisionsAndCommentsMarkupMode>
      <RevisionsAndCommentsDisplayMode>Original</RevisionsAndCommentsDisplayMode>
    </ConverterSpecificSettings>

    <OutputFormatSpecificSettings type="OutputFormatSpecificSettings_PDF">
      <FastWebView>false</FastWebView>
      <EmbedAllFonts>false</EmbedAllFonts>
      <SubsetFonts>false</SubsetFonts>
      <PostProcessFile>false</PostProcessFile>
      <ViewerPreferences>
        <CenterWindow>true</CenterWindow>
        <NavigationPane>Bookmarks</NavigationPane>
      </ViewerPreferences>
    </OutputFormatSpecificSettings>
```

```
<OCRSettings>
  <Language>English</Language>
  <Performance>Slow</Performance>
  <Paginate>true</Paginate>
  <WhiteList></WhiteList>
 <BlackList></BlackList>
 <Regions>
   <OCRRegion>
     <X>100</X><Y>100</Y><Width>200</Width><Height>50</Height>
     <StartPage>0</StartPage><EndPage>0</EndPage>
     <PageInterval>1</PageInterval><PageRange></PageRange>
  </OCRRegion>
 </Regions>
</OCRSettings>
<CompressionSettings>
     <RemoveAnnotations>Default</RemoveAnnotations>
     <RemoveBlankPages>Default</RemoveBlankPages>
     <RemoveBookmarks>Default</RemoveBookmarks>
     <RemoveEmbeddedFiles>Default</RemoveEmbeddedFiles>
     <RemoveFormFields>Default</RemoveFormFields>
     <RemoveJavaScript>Default</RemoveJavaScript>
     <RemoveMetadata>Default</RemoveMetadata>
     <RemovePageThumbnails>Default</RemovePageThumbnails>
     <PackFonts>Default</PackFonts>
     <PackDocument>Default</PackDocument>
     <RecompressImages>Default</RecompressImages>
     <EnableMRC>Default</EnableMRC>
     <PreserveSmoothing>Default</PreserveSmoothing>
     <DownscaleImages>Default</DownscaleImages>
     <EnableColorDetection>Default</EnableColorDetection>
     <EnableCharRepair>Default</EnableCharRepair>
     <EnableJPEG2000>Default</EnableJPEG2000>
     <EnableJBIG2>Default</EnableJBIG2>
     <JBIG2PMSThreshold>85</JBIG2PMSThreshold>
     <DownscaleResolution>150</DownscaleResolution>
     <DownscaleResolutionMRC>150</DownscaleResolutionMRC>
     <ImageQuality>ImageQualityDefault</ImageQuality>
 </CompressionSettings>
 <StartPage>1</StartPage>
 <EndPage>1</EndPage>
 <Fidelity>High</Fidelity>
 <Format>PDF</Format>
 <GenerateBookmarks>Disabled</GenerateBookmarks>
 <OpenPassword></OpenPassword>
 <OwnerPassword></OwnerPassword>
 <SecurityOptions>DisableContentAccessibility</SecurityOptions>
 <PageOrientation>Default</PageOrientation>
 <PDFProfile>PDF_A1B</PDFProfile>
 <Quality>OptimizeForPrint</Quality>
 <Range>VisibleDocuments</Range>
 <TOCSettings>....</TOCSettings>
</ConversionSettings>
```

```
</Override>
```

Please note:

- When entering values only specify those fields you want to override, leave all other fields completely out (don't just provide empty values, delete the entire line. (See the examples below)
- All values are case sensitive.
- Boolean values (true / false) need to be in *all lowercase*.
- There is no need to specify the name of the enumeration, e.g. in *RevisionsAndCommentsMarkupMode*
- In the *ConverterSpecificSettings* element you must specify the 'type' attribute to specify the exact type.
- When specifying multiple values, e.g. in the *SecurityOptions* element, then please separate these options using a blank space.
- The OutputFormatSpecificSettings property requires version 7.0 of the PDF Converter or newer.
- The *TOCSettings* property requires version 7.3 of the PDF Converter or newer.

A number of examples are provided below.

Make output format dynamic

If a workflow must have the ability to convert to different file formats, depending on a workflow parameter or column value, then you cannot use the normal drop down menu to pre-select the output format. Instead specify the following XML, position the cursor after <Format> and add a reference to the field that holds the *output format*, e.g. XLS, PDF, DOCX etc.

```
<Override>
    <ConversionSettings>
        <Format></Format>
    </ConversionSettings>
</Override>
```

Convert MS-Word files with different revision tracking options

MS-Word files support *revision tracking*, which is ideal for visualising what has changed in a document. By default the PDF Converter displays the *Final* version of the document and doesn't show the individual revisions. However, in some situations you may want to display these revisions and control how they are displayed (in-line or in balloons).

The XML to override these settings is as follows

```
<Override>
 <ConversionSettings>
  <ConverterSpecificSettings type="ConverterSpecificSettings_WordProcessing">
   <ProcessDocumentTemplate>true</ProcessDocumentTemplate>
```

```
    <RevisionsAndCommentsMarkupMode>InLine</RevisionsAndCommentsMarkupMode>
    <RevisionsAndCommentsDisplayMode>OriginalShowingMarkup
    </RevisionsAndCommentsDisplayMode>
   </ConverterSpecificSettings>
  </ConversionSettings>
</Override>
```

The possible values for *RevisionsAndCommentsMarkupMode* are:

- **InLine:** Show all revisions Inline.
- **Balloon:** Show all revisions in balloons.
- **Mixed:** Show only comments and formatting in balloons.

The possible values for *RevisionsAndCommentsDisplayMode* are:

- **FinalShowingMarkup:** Show the document with all proposed changes highlighted.
- **Final:** Show the document with all proposed changes included.
- **OriginalShowingMarkup:** Show the original document with all proposed changes highlighted.
- **Original:** Show the document before any changes were made.
- **SimpleMarkup:** Shows the final document in simple mark-up, with revisions incorporated, but with no mark-up visible.

Trim page numbers when converting to PDF

In some situations you may not be interested in all pages of the converted file, e.g you only want to convert the cover page (Page 1). To achieve this just set the *StartPage* and *EndPage* to *1* as follows:

```
<Override>
    <ConversionSettings>
        <StartPage>1</StartPage>
        <EndPage>1</EndPage>
    </ConversionSettings>
</Override>
```

When the start or end page should not be limited, specify 0 inside the element or remove the element. Do not include empty elements, e.g. <EndPage> </EndPage>, as this does not represent a valid number. For example, to start conversion at the second page, and include all pages to the very last page, use the following:

```
<Override>
    <ConversionSettings>
        <StartPage>2</StartPage>
    </ConversionSettings>
</Override>
```

Specify ‘open’ passwords for secured documents

If your MS-Word files have been saved using a password then these files cannot be converted. However, if the password is known then it can be specified as follows.

```
<Override>
    <OpenOptions>
        <Password></Password>
    </OpenOptions>
</Override>
```

Disable refreshing of content

By default the PDF Converter refreshes all content in an MS-Word file, e.g. embedded fields, table of contents, smart parts, etc. If this is not desired then specify the following XML:

```
<Override>
    <OpenOptions>
        <RefreshContent>false</RefreshContent>
    </OpenOptions>
</Override>
```

Specify PDF profile, e.g. PDF/A or a specific PDF Version

The PDF Converter provides full support for PDF/A output (See Administration Guide), however if you want to use it from workflows it is an *all or nothing* approach using a flag in the configuration file. To control PDF/A on a request by request basis either use our Web Services Interface or the following XML:

```
<Override>
    <ConversionSettings>
       <!-- Set the output profile -->
       <PDFProfile>PDF_A1B</PDFProfile>
       <!-- Force post processing -->
       <OutputFormatSpecificSettings type="OutputFormatSpecificSettings_PDF">
            <FastWebView>false</FastWebView>
            <EmbedAllFonts>true</EmbedAllFonts>
            <SubsetFonts>false</SubsetFonts>
            <PostProcessFile>true</PostProcessFile>
       </OutputFormatSpecificSettings>
    </ConversionSettings>
</Override>
```

This will make sure that all converted files conform to the PDF/A standard. However, if the source file is already in PDF format, and needs converting to PDF/A, then the *SkipPDFFiles* setting will need to be disabled first.

The PDFProfile element supports the following values, *please note that the use of this functionality requires a Muhimbi PDF Converter Professional license.*

- **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_A3U:** Use the PDF/A3u standard (experimental).
- **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:** PDF 1.5 output (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).

Change conversion range

Some document formats such as Excel and PowerPoint allow sheets to be hidden. By default the PDF Converter converts all Visible Sheets / Slides, but perhaps you are only interested in the Active / Selected spreadsheet. This can be controlled using the following XML:

```
<Override>
    <ConversionSettings>
        <Range></Range>
    </ConversionSettings>
</Override>
```

The possible values for *Range* are:

- **VisibleDocuments:** Skips, in case of Excel and PowerPoint, any hidden tabs or slides.
- **AllDocuments:** Export all tabs or slides in a workspace.
- **ActiveDocuments:** Exports, in case of Excel, the selected tabs

Specify Converter Specific Settings

Quite a few of our converters support settings that are specific to that particular file format. For an example see the Revision Tracking example above. Although enhancements are made all the time, at the time of writing the following Converter Specific Settings are available.

- ConverterSpecificSettings_Cad
- ConverterSpecificSettings_CommandLineConverter
- ConverterSpecificSettings_HTML
- ConverterSpecificSettings_Image
- ConverterSpecificSettings_InfoPath
- ConverterSpecificSettings_MSG
- ConverterSpecificSettings_PDF
- ConverterSpecificSettings_Presentations
- ConverterSpecificSettings_Spreadsheets
- ConverterSpecificSettings_TIFF
- ConverterSpecificSettings_WordProcessing

For more detail see the Class Diagrams in the Developer Guide.

Specify Viewer Preferences

The PDF Converter allows *PDF Viewer Preferences* to be specified, e.g. Center the PDF Reader Window, Hide the Menu and Toolbars, display the bookmarks pane, etc. For full details see this blog post.

An example that shows how to set these preferences is provided below. Naturally it can be combined with the other 'overrides' discussed previously.

```
<Override>
    <ConversionSettings>
       <OutputFormatSpecificSettings type="OutputFormatSpecificSettings_PDF">
            <FastWebView>false</FastWebView>
            <EmbedAllFonts>false</EmbedAllFonts>
            <SubsetFonts>false</SubsetFonts>
            <PostProcessFile>false</PostProcessFile>
            <ViewerPreferences>
              <CenterWindow>true</CenterWindow>
              <NavigationPane>Bookmarks</NavigationPane>
              <DisplayTitle>false</DisplayTitle>
              <FitWindow>false</FitWindow>
              <HideMenubar>false</HideMenubar>
              <HideToolbar>false</HideToolbar>
              <HideWindowUI>false</HideWindowUI>
              <PageLayout>SinglePage</PageLayout>
              <HideEmptyNavigationPane>false</HideEmptyNavigationPane>
              <PageScalingMode>Default</PageScalingMode>
              <FullScreen>false</FullScreen>
            </ViewerPreferences>
       </OutputFormatSpecificSettings>
    </ConversionSettings>
</Override>
```

Control Font Embedding, PDF Version, PDF Fast Web View

As of version 7.0 a number of facilities have been added to the PDF Converter Professional add-on license. These features allow fonts to be embedded / stripped, Fast Web View (Linearisation) to be enabled and the PDF Version to be changed (anything between PDF 1.1 and 1.7, including PDF/A1b, 2b and 3b). For details see this blog post.

An example that shows how to specify these new features using the XML based override syntax can be found below. Please make sure that *PostProcessFile* is set to *true* in order to pick up these settings.

```
<Override>
    <ConversionSettings>
       <OutputFormatSpecificSettings type="OutputFormatSpecificSettings_PDF">
            <FastWebView>true</FastWebView>
            <EmbedAllFonts>true</EmbedAllFonts>
            <SubsetFonts>false</SubsetFonts>
            <PostProcessFile>true</PostProcessFile>
       </OutputFormatSpecificSettings>
    </ConversionSettings>
</Override>
```

Specifying which InfoPath views to convert

As mentioned in the MS-Word revision tracking example above, it is possible to specify *Converter Specific Settings*. In this example we'll show how to use this facility to specify which InfoPath views to convert. The same can be achieved at design time or by setting workflow variables, but the example provided below allows view names to be specified at run-time and only requires a single workflow step.

Let's have a look at the code needed for doing the same via our Web Services based object model (*sample code is C#, other languages follow the same structure*).

```
ConverterSpecificSettings_InfoPath csc =
                                new ConverterSpecificSettings_InfoPath();
csc.ConversionViews = new InfoPathView[2];
csc.ConversionViews[0] = new InfoPathView();
csc.ConversionViews[0].Name = "NAME-OF-VIEW1";
csc.ConversionViews[1] = new InfoPathView();
csc.ConversionViews[1].Name = "NAME-OF-VIEW2";
// ** As we are overriding settings, we need to override ALL of them
csc.ConvertAttachments = true;
csc.AutoTrustForms = false;
csc.ProcessFullTrustForms = true;
csc.StripDataObjects = true;
csc.StripDotNETCode = true;
conversionSettings.ConverterSpecificSettings = csc;
```

What is important to realise is that when specifying *ConverterSpecificSettings*, it is essential that ALL values are specified as fields that are not will be initialised to their default value. The default value for a boolean field is 'false', which in this example would mean that attachments are not converted and InfoPath Data Objects are not stripped. Change these values in line with your needs, but unless you are 100% sure what each value means, keep them as specified in this example.

When serialising the code into XML we get the following:

```
<Override>
    <ConversionSettings>
        <ConverterSpecificSettings type="ConverterSpecificSettings_InfoPath">
            <ConversionViews>
                <InfoPathView>
                    <Name>NAME-OF-VIEW1</Name>
                </InfoPathView>
                <InfoPathView>
                    <Name>NAME_OF_VIEW2</Name>
                </InfoPathView>
            </ConversionViews>

           <!-- ** As we are overriding settings, we need to override ALL -->
            <ConvertAttachments>true</ConvertAttachments>
            <AutoTrustForms>true</AutoTrustForms>
            <ProcessFullTrustForms>true</ProcessFullTrustForms>
            <StripDataObjects>true</StripDataObjects>
            <StripDotNETCode>true</StripDotNETCode>
        </ConverterSpecificSettings>
    </ConversionSettings>
</Override>
```

Carrying out OCR processing during conversion

As of version 7.1 the Muhimbi PDF Converter supports the use of OCR (Optical Character Recognition) during the conversion process. This can be used to convert an image into a searchable PDF or make bitmap based PDFs (scanned PDFs) searchable and indexable. For details see chapter 15 *Carry out OCR (Optical Character Recognition).*

Starting with version 7.2 the PDF Converter includes dedicated OCR Workflow Actions. However, if you are running an older version of the software, or you have a good reason to carry out OCR using the *Convert Document* Workflow Activity, then specify OCR settings using the XML syntax described below.

Image files can be converted into searchable PDFs using the syntax below. When converting image based PDFs into searchable PDFs the syntax is the same, however the *SkipPDFFiles* setting will need to be disabled and the *PDF Pass-through converter* needs to be enabled in our Central Admin screen.

Please note that for use in a Production environment, the OCR facilities require a PDF Converter Professional add-on license.

```
<Override>
    <ConversionSettings>
       <OCRSettings>
          <Language>English</Language>
          <Performance>Slow</Performance>
          <Paginate>true</Paginate>
          <WhiteList></WhiteList>
          <BlackList></BlackList>
          <Regions>
             <OCRRegion>
               <X>100</X><Y>100</Y><Width>200</Width><Height>50</Height>
               <StartPage>0</StartPage><EndPage>0</EndPage>
               <PageInterval>1</PageInterval><PageRange></PageRange>
             </OCRRegion>
          </Regions>
       </OCRSettings>
    </ConversionSettings>
</Override>
```

The *Regions* element can be removed unless you wish to OCR only a certain part of the page or specific pages. When specifying *OCRRegions* the *StartPage*, *EndPage*, *PageInterval* and *PageRange* elements are optional.

The unit of measure for all coordinates is *pt* (Points, 1/72 inch).

Generate a table of contents

As of version 7.3 the Muhimbi PDF Converter comes with a brilliant new facility to generate tables of content for individual conversions as well as merge operations. Chapter 17 describes the basics and, although it is a little bit 'technical', explains how the underlying system works as well as a XSL based sample template.

Let's have a look at a typical example.

```
<Override>
  <ConversionSettings>
    <TOCSettings>
      <MinimumEntries>0</MinimumEntries>
      <Bookmark>Table Of Contents</Bookmark>
      <Location>Front</Location>
      <Properties>
        <NameValuePair>
          <Name>title</Name>
          <Value>Custom Title</Value>
        </NameValuePair>
      </Properties>
      <Template>http://YourServer/SomeSiteCollection/SomeDocLib/TOC.xsl</Template>
    </TOCSettings>
  </ConversionSettings>
</Override>
```

This XML is similar to the C# sample code in Chapter 17, which also provides details about the various properties such as *MinimumEntries* and *Location*. The recommended approach is to upload the XSL template to a SharePoint document library and specifying its location in the *Template* element.

If you wish to merge a number of documents AND add a table of contents for the entire - merged - PDF then this can be achieved as follows:

1. (Convert) and merge multiple files into a single PDF using the *Merge Documents into PDF* workflow activity (See the various examples in chapter 10).
2. Process the generated PDF by passing it into the *Convert Document* workflow activity and specifying *TOCSettings* using the XML override syntax.

The XML 'overriding' facility works in SharePoint Designer Workflows (See 9.1), K2 (See 5) as well as Nintex Workflows (See 9.2).

Appendix - Specifying path and file names

Regardless of the operation carried out by the Muhimbi PDF Converter, input and / or output file names, including paths, need to be specified. Although this has been made as intuitive as possible, we still get questions about it from time to time. We hope to clarify the situation in this Appendix. The latest version of this Appendix can be found on-line at https://www.muhimbi.com/blog/specifying-paths-and-file-names-when-using-the-pdf-converter-for-sharepoint/.

Target the same directory / use the same file name as the source file

If you wish to read from or write to the same location as the source file that is being processed, e.g. you are using a SharePoint Designer workflow to automatically convert files to PDF, then leave the path empty. This works for all situations with the exception of the HTML to PDF Conversion as that is not necessarily associated with a source file (it can convert URLs as well).

If you wish to use the same name for the target file as the source file then there is no need to specify a file name, the system will generate the name for you. When converting files to PDF the files' extension will automatically be updated to 'PDF', however when the source file is a PDF (e.g. when applying PDF Security) then omitting the file name and path will overwrite the source file. This may be desired, but keep it in mind.

Target a (sub) folder in the current Document Library

If you wish to specify a (sub) folder in the same Document Library as the source file then you must specify the Document Library name as well as the path to the folder. Just specifying the folder name will not work. For example, to convert a file to the '*Archives/PDFs*' folder in the *Shared Documents* library, specify:

```
Shared Documents/Archives/PDFs/
```

Make sure you use a trailing slash as that is used in some cases to resolve potentially conflicting file and folder names. Please do not start the path with a slash in this particular case.

Target a different Document Library

To specify a location in a different Document Library in the current Site, specify the name of the Document Library followed by any folders. E.g. if your workflow activity is operating on a file in the *Shared Documents* library, but you want to write the generated file to the *PDF* folder in the *Archive* Document Library then specify the following as the path:

```
Archive/PDF/
```

Make sure you use a trailing slash when specifying folders as that is used in some cases to resolve potentially conflicting file and folder names.

Target a Sub Site

SharePoint Site Collections can contain multiple Sub Sites. To write a file to a location in a Sub Site specify the name of that Sub Site followed by the name of the Document Library followed by any folder names. For example if the current workflow is acting on a file in the root web of a Site Collection and you wish to write it to the *PDFs* folder in the *Paid* Document Library in the *Invoices* Sub Site specify the following path:

```
Invoices/Paid/PDFs/
```

Make sure you use a trailing slash when specifying folders as that is used in some cases to resolve potentially conflicting file and folder names.

To target a Sub Site 'next' to the current site, use an absolute path. For details see *Targeting a different Site Collection* below. Please note that 'traditional' relative path notation such as '../../' is not supported.

Target a different Site Collection

In order to read from or write to a file in a different site collection you have to use an absolute path starting with '/'. For example if a PDF Conversion workflow is running in the *Accounting* Site collection and the generated PDF file should be written to the *Archiving* site collection in the *PDFs* folder in the *Accounting* Document Library, use the following path:

```
/Archiving/Accounting/PDFs/
```

If all your Site Collections live under '/sites/' then this will need to be reflected in the path, e.g.:

```
/sites/Archiving/Accounting/PDFs/
```

Please do not start absolute paths with *http://YourWebApplication/*, always start absolute paths with a slash.

Targeting historical files

As of version 6.0 it is possible to specify specific versions in a file's history for Merge and Watermark (elements) operations. The syntax is similar to what is described above, for example:

Specify a document in the same document library (named Automatic Merging):

```
_vti_history/1536/Automatic Merging/End Page.docx
```

Specify a document in a different document library:

```
_vti_history/1024/Shared Documents/Subfolder/End Page.docx
```

Specify a document in a sub-site of the current web:

```
SubSite/AnotherSubSite/_vti_history/1024/Shared Documents/Empty.pdf
```

Specify a document in a different site collection or in a sub web next to the current web:

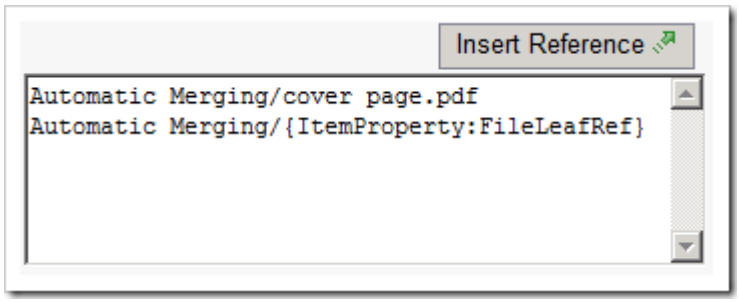
```
/sites/PDFTest/_vti_history/512/Shared Documents/introduction.pdf
```

Target a different Web Application

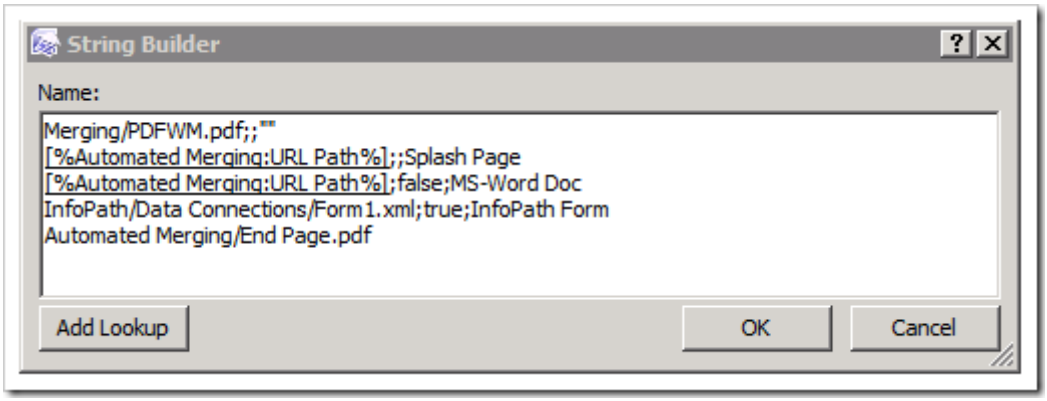
As there are clear security boundaries between SharePoint Web Applications it is not possible to exchange files between them. If this is required then you will need to use a third party Workflow Activity or use Nintex Workflow.

Creating dynamic paths / file names

All of Muhimbi's SharePoint Designer and Nintex Workflow activities allow lookup variables to be used in order to generate dynamic paths. For example, in order to make a workflow generic, the name of a source or target File / Library / Folder can be determined at run-time.



For Nintex Workflow use the 'Insert Reference' button



In SharePoint Designer use the 'Add Lookup' button

Some final, generic, hints and tips:

- Never start a path with *http://YourWebApplication/*. Start absolute paths with a '/'.
- Some activities support templates in the output file name, e.g. to automatically generate file names when splitting PDFs. For details see chapter 11 *Splitting PDFs into multiple documents*.
- Although you can use forward (/) as well as back (\) slashes, make it a habit to use forward slashes as SharePoint Designer 2010 (and later) workflows do not always deal well with back slashes.
- No matter where you read from or write to, your user must have access to the location that is being specified.

Appendix - Watermark field matrix

	Watermark	Text	Pdf	Rtf	Image	Ellipse	Rectangle	Line
hPosition	o	o	o	o	o	o	o	o
vPosition	o	o	o	o	o	o	o	o
x	o	o	o	o	o	o	o	m
y	o	o	o	o	o	o	o	m
width	m	m	o	m	o	m	m	-
height	m	m	o	m	o	m	m	-
zOrder	o	o	o	o	o	o	o	o
scaleMode	o	o	o	o	o	o	o	o
scaleX	o	o	o	o	o	o	o	o
scaleY	o	o	o	o	o	o	o	o
rotation	o	o	o	o	o	o	o	o
opacity	o	o	o	o	o	o	o	o
lineColor	-	o	o	o	o	o	o	o
lineWidth	-	o	o	o	o	o	o	o
fillColor	-	o	o	o	o	o	o	-
pageOrientation	o	-	-	-	-	-	-	-
startPage	o	-	-	-	-	-	-	-
endPage	o	-	-	-	-	-	-	-
pageInterval	o	-	-	-	-	-	-	-
pageRange	o	-	-	-	-	-	-	-
content	-	m	-	-	-	-	-	-
fontFamilyName	-	o	-	-	-	-	-	-
fontSize	-	o	-	-	-	-	-	-
fontStyle	-	o	-	-	-	-	-	-
wordWrap	-	o	-	-	-	-	-	-
hAlign	-	o	-	-	-	-	-	-
vAlign	-	o	-	-	-	-	-	-
pdfFilePath	-	-	m	-	-	-	-	-
rtfData	-	-	-	m	-	-	-	-
imageFilePath	-	-	-	-	m	-	-	-
endX	-	-	-	-	-	-	-	m
endY	-	-	-	-	-	-	-	m

default	available values / remarks				
absolute	absolute	random	left	center	right
absolute	absolute	random	top	middle	bottom
0	number as a string				
0	number as a string				
-	number as a string				
-	number as string				
0	int as string, <1 means background, >=1 means foreground				
absolute	absolute	exactFit	maintainaspectratio		
1	number as string, 1 means 100%				
1	number as string, 1 means 100%				
0	number as string, in degrees				
100	number as string, [0-100]				
#ff000000	either #aarrggbb or #rrggbb hex color				
0 or 1	1 for lines 0 for everything else				
#ff000000	either #aarrggbb or #rrggbb hex color				
both	portrait	landscape	both		
0	0 will default to the first page				
0	0 will default to last page				
0	0 will default to every page				
null	comma separated list of page numbers or intervals (1-5,7)				
-	the text to display				
Arial	name of a font				
12	positive number as string				
regular	regular	bold	italic	strikeout	underline
	Font styles can be combined, e.g. "bold|italic"				
wordonly	none	character	word	wordonly	
center	left	center	right	justify	
middle	top	middle	bottom		
-	path to the watermark pdf file				
-	the rtf source as string				
-	path to the watermark image file				
-	number as string				
-	number as string				

	o	optional		m	mandatory

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 User Guide.

- Using the PDF Converter from a SharePoint Designer workflow
- Convert and merge multiple PDF files using a SharePoint Designer workflow
- Using the SharePoint Workflow Manager to Convert documents to PDF
- Using the PDF Converter in combination with SharePoint Online / Office 365
- Installing The PDF Converter APP in SP2013 & SP2016 (on-premise)
- Converting multiple SharePoint files to PDF Format using Nintex Workflow
- Convert to non-PDF formats 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 Nintex Workflow
- Extract PDF Forms Data (FDF, XFDF, XML) using SharePoint or C#, Java, PHP
- Combining multiple Nintex Workflow Activities into a sequence
- Copy Meta-Data and set content types using a SharePoint Designer Workflow
- Convert SharePoint documents to PDF using K2 workflows
- Watermarking features of the Muhimbi PDF Converter for SharePoint
- Applying user specific watermarks when a PDF document is opened
- Apply User Specific PDF Security when a document is opened in SharePoint
- Convert and merge files to PDF using the SharePoint User Interface
- PDF/A Support in the Muhimbi PDF Converter Services & SharePoint
- OCR Facilities provided by Muhimbi's server based PDF Conversion products
- OCR Scans and Images using SharePoint Designer Workflows
- OCR Scans and Images using Nintex Workflow
- Extract text from scanned content using OCR and SharePoint Designer Workflows
- Extract text from scanned content using OCR and Nintex Workflow
- OCR Scans and Images using a Web Services based Interface (WSImport)
- OCR Scans and Images using a Web Services based Interface (.NET)
- Merging dynamic data into watermarks using the PDF Converter for SharePoint
- Specifying filtering criteria when automatically applying watermarks in SharePoint
- Inserting SharePoint List data into a PDF document using a workflow
- Configure PDF Security from a SharePoint Designer Workflow
- Splitting PDF Files using a SharePoint workflow
- Splitting PDF Files using the PDF Converter Web Service (.NET)
- Converting and merging multiple files using Web Services (.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
- Set PDF Version, enable Fast Web Views, embed / strip fonts
- Specifying PDF Viewer Preferences
- Automatically convert files to PDF using an e-mail enabled Document Library
- Batch print InfoPath Forms using the PDF Converter for SharePoint
- Convert InfoPath to MS-Word, Excel, XPS and PDF
- Cross-Convert document types (xls to xslx, doc to docx)
- Converting InfoPath forms including all attachments to a single PDF file
- Controlling which views to export to PDF format in InfoPath
- Using SharePoint Forms Services to convert InfoPath forms to PDF format
- Dealing with hyperlinks when converting InfoPath files to PDF format
- Programmatically Convert SharePoint HTML pages 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
- 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
- Converting files to PDF Format using a Web Services based interface (.NET)
- Converting files to PDF Format using a Web Services based interface (Java)
- Invoking the PDF Converter Web Service from Visual Studio 2005 using vb.net
- Converting PDF files to PDF/A1b using a Web Services based interface
- Adding custom Converters to Muhimbi's range of PDF Conversion products
- Embedding SharePoint Document IDs in PDF files and generating Short URLs
- Specifying paths and file names when using the PDF Converter for SharePoint
- Muhimbi PDF Converter Deployment scenarios
- Send rich emails with attachments from a SharePoint Designer Workflow
- Create Shortened ('TinyURL') links from your SharePoint Workflow
- SharePoint Designer workflows trigger before properties are set
- Troubleshooting steps for the PDF Converter for SharePoint
- Troubleshooting InfoPath to PDF Conversion / Document Converter Architecture
- Performance metrics for the Muhimbi PDF Converter
- Adding the PDF Converter to non standard Document and Form Libraries
- Tuning SharePoint's workflow engine

Appendix - Licensing

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

Muhimbi's software is licensed on a per-server basis. This means that a license is required for every Production and Disaster Recovery server that runs our software. This means all Web Front End servers, all Application Servers participating in workflows (how to determine this) and all servers running the conversion service. Any K2 Workflow Servers that have our K2 integration facilities installed require a license as well.

For details see:

1. License 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 an evaluation message will be displayed on most screens, in the workflow history and in any generated 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 SharePoint farms consisting of a single combined WFE / App server.
3. **Small Farm License:** Covers a single SharePoint farm consisting of up to 3 servers with support for load balancing and redundancy.
4. **Enterprise License:** Covers an unlimited number of SharePoint Farms, Servers 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, which is a separate product.

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.