User Manual

Foxit® PDF Toolkit

Microsoft® Partner
Gold Independent Software Vendor (ISV)

©2015 Foxit Software Incorporated. All rights reserved.

Copyright (C) 2015 Foxit Software Incorporated. All Rights Reserved.

No part of this document can be reproduced, transferred, distributed or stored in any format without the prior written permission of Foxit.

Anti-Grain Geometry - Version 2.3, Copyright (C) 2002-2005 Maxim Shemanarev (http://www.antigrain.com).

FreeType2, Copyright 1996-2001, 2006 David Turner, Robert Wilhelm, and Werner emberg. (http://git.savannah.gnu.org/cgit/freetype/freetype2.git/tree/docs/FTL.TXT)

THE SOFTWARE IS PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EXPRESS, IMPLIED OR OTHERWISE, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.
IN NO EVENT SHALL SAM LEFFLER OR SILICON GRAPHICS BE LIABLE FOR ANY SPECIAL, INCIDENTAL, INDIRECT OR CONSEQUENTIAL DAMAGES OF ANY KIND, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER OR NOT ADVISED OF THE POSSIBILITY OF DAMAGE, AND ON ANY THEORY OF LIABILITY, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

LibJPEG copyright (C) 1991-1998 independent JPEG Group. (http://sourceforge.net/projects/libjpeg/)

ZLib (zlib 1.2.2), Copyright (C) 1995-2003 Jean-loup Gailly and Mark Adler. (http://www.zlib.net/zlib_license.html)
Copyright (C) 1995-2013 Jean-loup Gailly and Mark Adler This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software.
Jean-loup Gailly Mark Adler
jloup@gzip.org madler@alumni.caltech.edu

Little CMS, Copyright (c) 1998-2010 Marti Maria Saguer. (http://www.littlecms.com/)

Kakadu, Copyright (C) 2001, David Taubman, The University of New South Wales (UNSW).
(http://www.kakadusoftware.com/index.php?option=com_content&task=blogcategory&id=6&Itemid=12)

PNG, Copyright (C) 1998-2009 Glenn Randers-Pehrson.

LibTIFF, Copyright (C) 1988-1997 Sam Leffler and Copyright (C) 1991-1997 Silicon Graphics, Inc.

ZXing (Zebra Crossing), Copyright 2007 ZXing authors Apache 2.0

libjpeg-turbo, Copyright (C) 2011 D. R. Commander. All Rights Reserved.

Table of Contents

1 Introduction to Foxit PDF Toolkit ... 1

1.1 Why is Foxit PDF Toolkit your choice ... 2
1.2 Foxit PDF Toolkit Features ... 2
  1.2.1 Image2PDF Features ... 2
  1.2.2 Office2PDF Features ... 3
  1.2.3 PDFWatermark Features ... 3
  1.2.4 PDFHeaderFooter Features ... 4
  1.2.5 PDFOptimizer Features ... 4
1.3 Common Use Case Scenarios ... 5
  1.3.1 Image2PDF Use Case Scenarios ... 5
  1.3.2 Office2PDF Use Case Scenarios ... 5
  1.3.3 PDFWatermark Use Case Scenarios ... 5
  1.3.4 PDFHeaderFooter Use Case Scenarios ... 5
  1.3.5 PDFOptimizer Use Case Scenarios ... 6
1.4 System Requirements ... 6
1.5 About This Manual ... 6

2 Installing and Uninstalling Foxit PDF Toolkit ... 7

2.1 Foxit PDF Toolkit Installation ... 7
2.2 Foxit PDF Toolkit Evaluation ... 8
2.3 Foxit PDF Toolkit Registration ... 9
2.4 Foxit PDF Toolkit Uninstallation ... 10

3 Command Line Usage ... 11

3.1 Image2PDF ... 11
  3.1.1 Basic Syntax ... 11
  3.1.2 Command Line Summary ... 11
  3.1.3 Basic Usage ... 14
3.2 Office2PDF ... 23
  3.2.1 Basic Syntax ... 23
  3.2.2 Command Line Summary ... 23
  3.2.3 Basic Usage ... 26

3.3 PDFWatermark....33

3.3.1 Basic Syntax....33

3.3.2 Command Line Summary....33

3.3.3 Basic Usage....35

3.4 PDFHeaderFooter....41

3.4.1 Basic Syntax....41

3.4.2 Command Line Summary....41

3.4.3 Basic Usage....44

3.5 PDFOptimizer....50

3.5.1 Basic Syntax....50

3.5.2 Command Line Summary....50

3.5.3 Basic Usage....55

3.6 RMS....64

3.6.1 Basic Syntax....64

4 Foxit Configuration Tool....65

4.1 Watermark Configuration Tool....65

4.1.1 Watermark Settings....66

4.2 PDFHeaderFooter Configuration Tool....68

4.2.1 Header/Footer Settings....70

5 Working with API....72

5.1 Image2PDF....73

5.1.1 Working with Image2PDF API....73

5.1.2 Reporting Progress Messages and Errors....76

5.2 Office2PDF....77

5.2.1 Working with Office2PDF API....77

5.2.2 Reporting Progress Messages and Errors....79

5.3 PDFWatermark....80

5.3.1 Working with PDFWatermark API....80

5.3.2 Reporting Progress Messages and Errors....81

5.4 PDFHeaderFooter....83

5.4.1 Working with PDFHeaderFooter API....83

5.4.2 Reporting Progress Messages and Errors....85

5.5 PDFOptimizer 86
  5.5.1 Working with PDFOptimizer API 86
  5.5.2 Reporting Progress Messages and Errors 88

6 Support........ 90

6.1 Reporting Problem 90
6.2 Contact Information 90

1 Introduction to Foxit PDF Toolkit

Foxit PDF Toolkit consists of a series of command line tools to perform batch PDF generation and processing in server environments. It includes six tools: Image2PDF, Office2PDF, PDFWatermark, PDFHeaderFooter, PDFOptimizer and RMS. Each tool offers a simple-to-use API for users who want to perform PDF manipulation through calling API.

In this manual, we introduce tools of Image2PDF, Office2PDF, PDFWatermark, PDFHeaderFooter and PDFOptimizer. For the RMS tool, we just list the basic syntax in section 3.6 "RMS", please go to the "rms" folder in the installation package for the detailed introduction.

Image2PDF is an easy-to-use command line tool for you to batch convert image files into high-quality PDF files in server environments, without the need of installing any additional software. Image2PDF currently supports BMP, PNG, JPEG, JPX, GIF, TIFF (or TIF, including the image with a single page or multiple pages) image formats and provides a great many options to control the properties of the output PDF files.

Office2PDF is an easy-to-use command line tool for you to batch convert Microsoft Office files into professional-quality PDF files in server environments, with Microsoft Office installed (Office2PDF tool saves the converted PDF files with Microsoft Office Engine). Office2PDF currently supports the popular Microsoft Office document formats, including doc, docx, xls, xlsx, ppt, pptx, and the converted PDF files can be compliant with PDF/A Standard.

PDFWatermark is an easy-to-use command line tool for you to batch add watermark into PDF files in server environments, without the need of installing any additional software. PDFWatermark applies pre-made watermark, such as text or an image, to PDF files. The watermark is saved as a configuration file with the extension ".xml" which was generated by the build-in Foxit Configuration Tool (fpdfwmconf.exe or fpdfwmconf64.exe depending on your operating system, a GUI application).

PDFHeaderFooter is an easy-to-use command line tool for you to batch add header/footer into PDF files in server environments, without the need of installing any additional software. PDFHeaderFooter applies pre-made header/footer to PDF files. The header/footer is saved as a configuration file with the extension ".xml" which was generated by the build-in Foxit Configuration Tool (fpdfhfconf.exe or fpdfhfconf64.exe depending on your operating system, a GUI application).

PDFOptimizer is an easy-to-use command line tool for you to optimize PDF files in server environments, without the need of installing any additional software. With the PDFOptimizer, users can optimize PDF files to reduce disk space and make them easier to share or archive.

1.1 Why is Foxit PDF Toolkit your choice

Foxit is an Amazon-invested leading software provider of solutions for reading, editing, creating, organizing, and securing PDF documents. Foxit PDF Toolkit is a suite of modules to perform high volume PDF creation and processing on Windows servers. These modules help IT organizations develop workflows to process large amounts of PDF files. They also provide libraries for software development groups to incorporate PDF processing into their applications. Customers choose this product for the following reasons:

- **High performance** – Multi-threading support speeds up the PDF processing based on today's server architectures.
- **Professional quality** – Professional-quality on PDF manipulation and PDF conversion.
- **Lightweight footprint** – Lower memory usage and can be installed quickly.
- **Perfect message mechanism** – Perfect hints and output messages enable better user experience.
- **Robustness** – Ensures smooth running of the application and the enhanced ability of fault tolerant.
- **Easy to integrate** – Command line or application interfaces enable flexible and seamless integration with user's existing workflows.
- **Plug and Play** – Choose one or more of the specific modules that meet your needs.

In addition, Foxit products are fully supported by Foxit’s dedicated support engineers if support and maintenance are purchased. Updates are released on a regular basis. Foxit PDF Toolkit will be the right choice if you need batch PDF generation and processing with excellent features and low cost!

1.2 Foxit PDF Toolkit Features

1.2.1 Image2PDF Features

- Batch convert image files to PDF files in server environments.
- Support multi-thread processing.
- Support sub-folder recursion processing.
- Support single-file processing, single-folder processing and multi-file processing.
- Support various image formats like BMP, PNG, JPEG, JPX, GIF and TIFF (TIF).
- Convert multiple image files into one PDF file, or individual PDF files.
- Specify any resolution (DPI) for the PDF file to be converted.

- Set page width and height for the PDF file to be converted.
- Set open password for the PDF file to be converted.
- Add bookmarks to the PDF file to be converted.
- Set margin for each PDF page.
- Support wildcard characters in batch conversion, e.g., *.jpg, *.png.
- Offer simple-to-use API.

1.2.2 Office2PDF Features

- Batch convert Microsoft Office files to PDF files in server environments.
- Support multi-thread processing.
- Support sub-folder recursion processing.
- Support single PDF file processing or single-folder processing.
- Support the popular Microsoft Office document formats, including doc, docx, xls, xlsx, ppt and pptx.
- Add bookmarks to the PDF file converted only from Microsoft Word document.
- Convert every sheet of Microsoft Excel document into one PDF file.
- Retain hyperlinks from Microsoft Office file to PDF file.
- Support wildcard characters in batch conversion, e.g., *.docx, *.pptx.
- Support PDF/A Standard for the PDF file to be converted.
- Offer simple-to-use API.

1.2.3 PDFWatermark Features

- Batch add watermark into PDF files in server environments.
- Support multi-thread processing.
- Support sub-folder recursion processing.
- Support single PDF file processing or single-folder processing.
- Support stylized text watermark including font, font size, font color and underline settings.
- Support image watermark with various formats (BMP, DIB, JPG, JPEG, JPE, GIF, TIF, PNG, and TIFF).
- Rotate watermark in 360 degree.
- Set opacity for text and image watermark.
- Scale relative to target page for text and image watermark.
- Control the visible of watermark when printing
- Control the visible of watermark when displaying on screen.
- Keep watermark position and size for different page sizes.
- Appear on top or background of page.
- Set watermark position in PDF page.
- Specify the page range to add watermark.
- Support wildcard characters in batch conversion, e.g., *.pdf.

- Preview watermark in the build-in Foxit Configuration Tool.
- Offer simple-to-use API.

1.2.4 PDFHeaderFooter Features

- Batch add header/footer into PDF files in server environments.
- Support multi-thread processing.
- Support sub-folder recursion processing.
- Support single PDF file processing or single-folder processing.
- Support stylized text header/footer including font, font size, font color, underline and embedded font settings.
- Support margin settings.
- Shrink a document to avoid overwriting the document's text and graphics.
- Keep header/footer position and size for different page sizes.
- Set page number and date as header/footer.
- Support page number and date format settings.
- Specify the page range to add header/footer.
- Support wildcard characters in batch conversion, e.g., *.pdf.
- Preview header/footer in the build-in Foxit Configuration Tool.
- Offer simple-to-use API.

1.2.5 PDFOptimizer Features

- Batch optimize PDF files in server environments.
- Support multi-thread processing.
- Support sub-folder recursion processing.
- Support single PDF file processing or single-folder processing.
- Support downsampling and compression for color/grayscale and monochrome images.
- Unembed fonts of PDF files.
- Discard objects and user data of PDF files:
    - Discard all form submission, import and reset actions.
    - Flatten form fields.
    - Discard all JavaScript actions.
    - Discard embedded page thumbnails.
    - Discard embedded print settings.
    - Discard bookmarks.
    - Discard all comments, forms and multimedia.
    - Discard external cross references.
    - Discard document information and metadata.

  - Discard file attachments.
  - Discard private data of other applications.
- Clear up PDF files:
  - Discard all form submission, import and reset actions.
  - Use Flate to encode streams that are not encoded.
  - In streams that use LZW encoding, use Flate instead.
  - Remove invalid bookmarks.
  - Remove invalid links.
- Support wildcard characters in batch conversion, e.g., *.pdf.
- Offer simple-to-use API.

1.3 Common Use Case Scenarios

1.3.1 Image2PDF Use Case Scenarios

- Batch convert images to PDF files in server environments, for better image management.
- Make electronic books. Users can scan paper document to image files and then convert them into one PDF file for composing an electronic book.
- Make electronic albums. Users can collect their photos and then convert them into one PDF file for composing an electronic album.

1.3.2 Office2PDF Use Case Scenarios

- Batch convert Microsoft Office files to PDF files in server environments.
- Set the converted PDF files to be compliant with PDF/A standard for better archive management.

1.3.3 PDFWatermark Use Case Scenarios

- Batch add watermark into PDF files to protect users' copyright. Users can use company logos, author signatures, products or web addresses as watermark to protect their PDF files.
- Label PDF file status for better management. Users can easily label status of their PDF files, such as approved, draft, final or confidential. For example, label "Confidential" on PDF pages that include sensitive information or "Draft" on a preliminary PDF document to be distributed for review.

1.3.4 PDFHeaderFooter Use Case Scenarios

- Batch add header/footer into PDF files to present some useful information, such as date, page numbers or the title of the document.

1.3.5 PDFOptimizer Use Case Scenarios

- Batch optimize large PDF files for electronic document exchange or space-saving document archiving in server environments. Large PDF files are not suitable for electronic document exchange or space-saving document archiving. Users can optimize large PDF documents instead of converting them into other formats which may make the situation even worse.

1.4 System Requirements

Windows Platform：

- Windows Server 2012
- Windows Server 2008 R2
- Windows Server 2003
- Windows 8.1
- Windows 7
- Windows Vista
- Windows XP

Note *For the Office2PDF tool, please make sure that you have already installed Microsoft Office 2007 SP2 or later and virtual printers. If Microsoft Office 2007 is not SP2 version, please download the "Microsoft Save as PDF" plugin from http://www.microsoft.com/en-us/download/details.aspx?id=9943*.

1.5 About This Manual

This manual aims at introducing the command line usage of Foxit PDF Toolkit. It is intended for the audiences who want to batch generate or process PDF files in server environments.

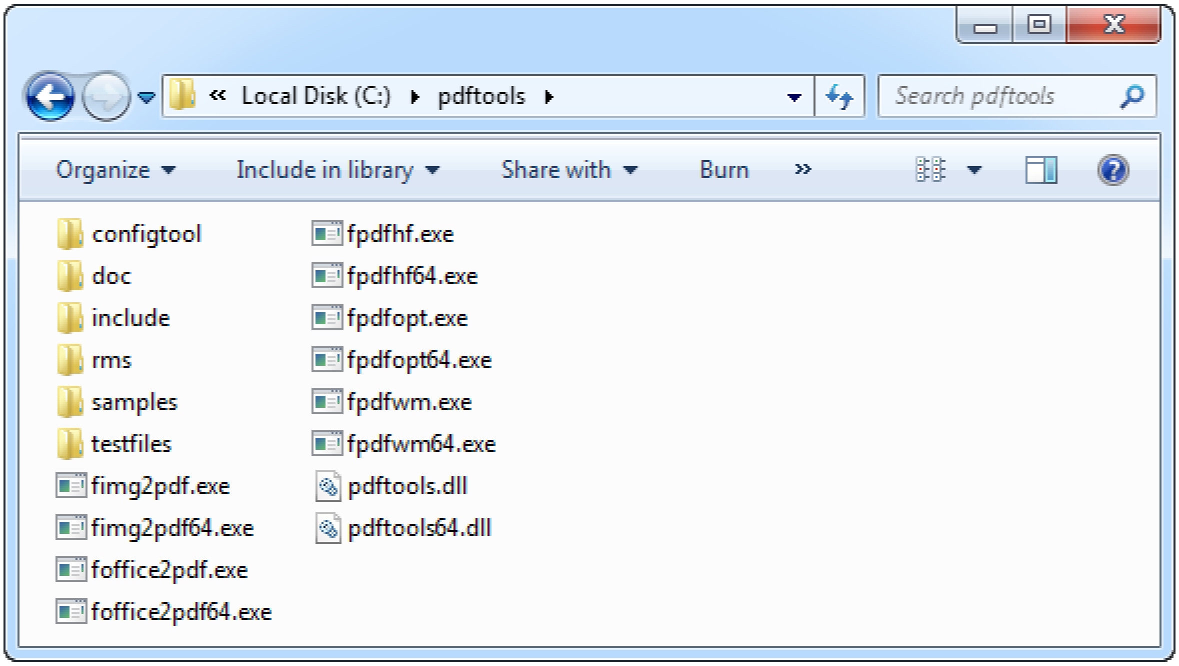
- Section 1: gives an introduction of Foxit PDF Toolkit.
- Section 2: illustrates how to install and uninstall Foxit PDF Toolkit.
- Section 3: describes basic usage of Foxit PDF Toolkit.
- Section 4: introduces Foxit Configuration Tool of PDFWatermark and PDFHeaderFooter tools.
- Section 5: presents how to work with API.
- Section 6: provides the support information.

2 Installing and Uninstalling Foxit PDF Toolkit

2.1 Foxit PDF Toolkit Installation

Installation of Foxit PDF Toolkit is straightforward. You can download the trial release package which is a zip file from our website (www.foxitsoftware.com), and then extract the package to the desired location, which is shown in the following figure. In this manual, we rename the package "pdftools" and unzip it to "C:\pdftools". The package contains:

configtool: configuration tools for pdfwatermark and pdfheaderFooter
doc: user manual for Foxit PDF Toolkit
include: header file for Foxit PDF Toolkit
rms: RMS command line tool
samples: some batch samples for Foxit PDF Toolkit
testfiles: testfiles for Foxit PDF Toolkit

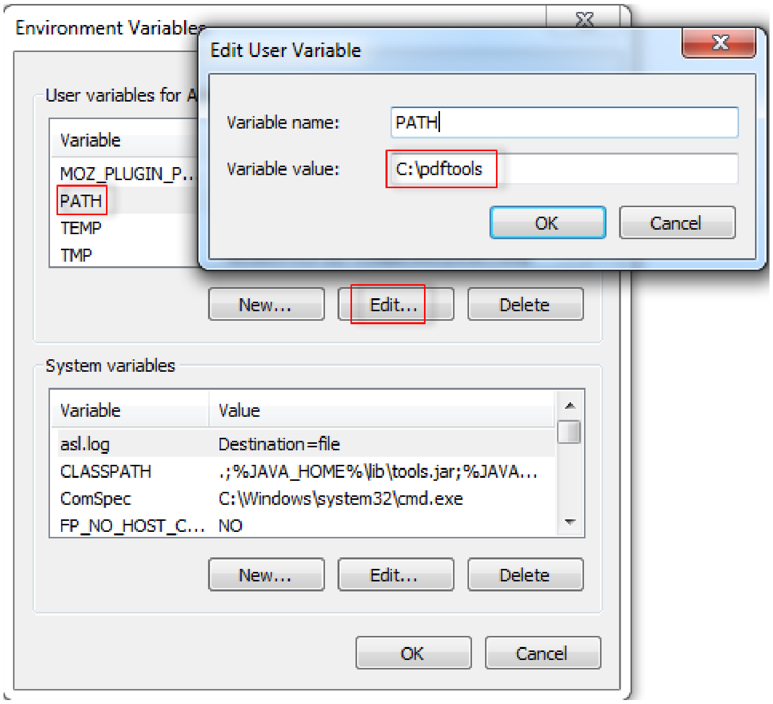


Foxit PDF Toolkit package

After that, open a terminal session and change the directory to the installation location to run your application except for PDFWatermark and PDFHeaderFooter tools which will be used only after you use Foxit

Configuration Tool to generate configuration files that contain the setting information of watermark or header/footer. For the Office2PDF tool, please make sure that you have already installed Microsoft Office 2007 SP2 or later and virtual printers. If Microsoft Office 2007 is not SP2 version, please download the "Microsoft Save as PDF" plugin from http://www.microsoft.com/en-us/download/details.aspx?id=9943.

Tips: You can set the installation path to Environment Variables, so that you could use the commands in terminal window directly, without the need of changing the directory to the installation location. Go to **Start**-> **Control Panel**-> **System**-> **Advanced system settings** -> **Advanced** -> **Environment Variables**, in "User variables for Administrator" box, select **PATH** and **Edit**, and then add the installation path as shown in the following figure. If the environment variable "path" does not exist, create it by clicking **New**.



Variable name: PATH

Variable value: C:\pdftools

Variable	Value
asl.log	Destination=file
CLASSPATH	.;%JAVA_HOME%\lib\tools.jar;%JAVA...
ComSpec	C:\Windows\system32\cmd.exe
FP_NO_HOST_C...	NO

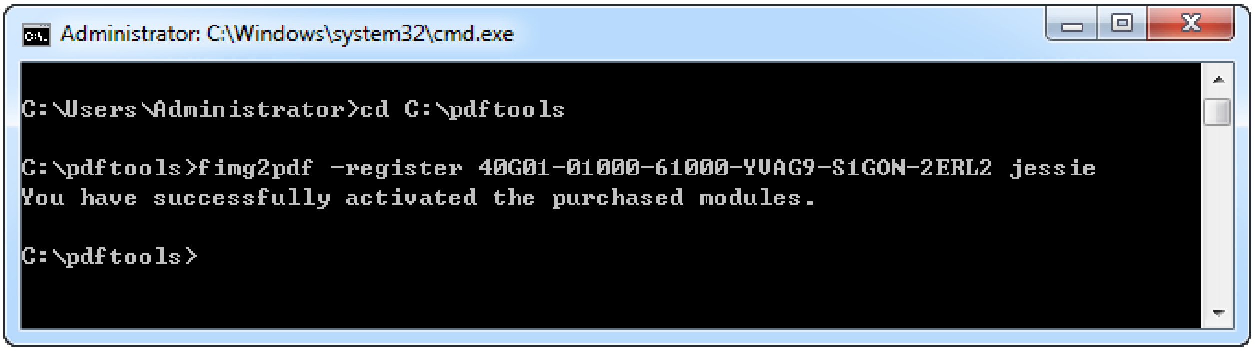
Set installation path to Environment Variables

2.2 Foxit PDF Toolkit Evaluation

The Foxit PDF Toolkit is distributed on a "try-before-you -buy" basis. Foxit allows users to download trial version to evaluate the features. You have 30 days free trial, during which the pages of all generated PDF files will contain our watermark. After evaluation period, customers should contact Foxit sales team to purchase the corresponding activation code of the desired tools if they want to continue using it.

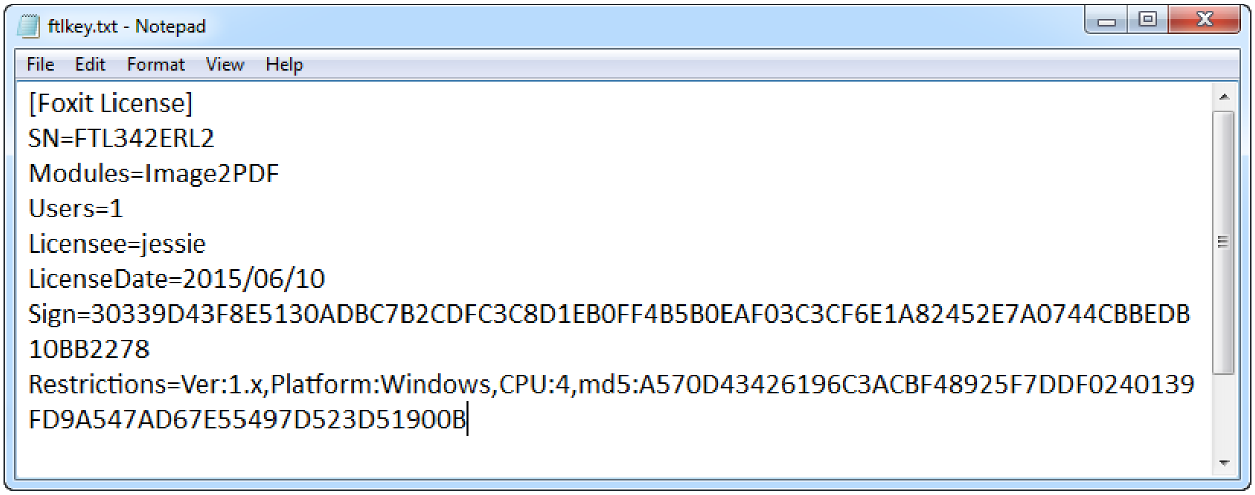
2.3 Foxit PDF Toolkit Registration

When you got the activation code of the tool you want to purchase, please use the argument "**-register <code> <licensee>**" to active it in command line window. For example, assume you get an activation code (40G01-01000-61000-YVAG9-S1GON-2ERL2) of Image2PDF tool, you can use "fimg2pdf -register 40G01-01000-61000-YVAG9-S1GON-2ERL2 jessie" in command line window as follows.



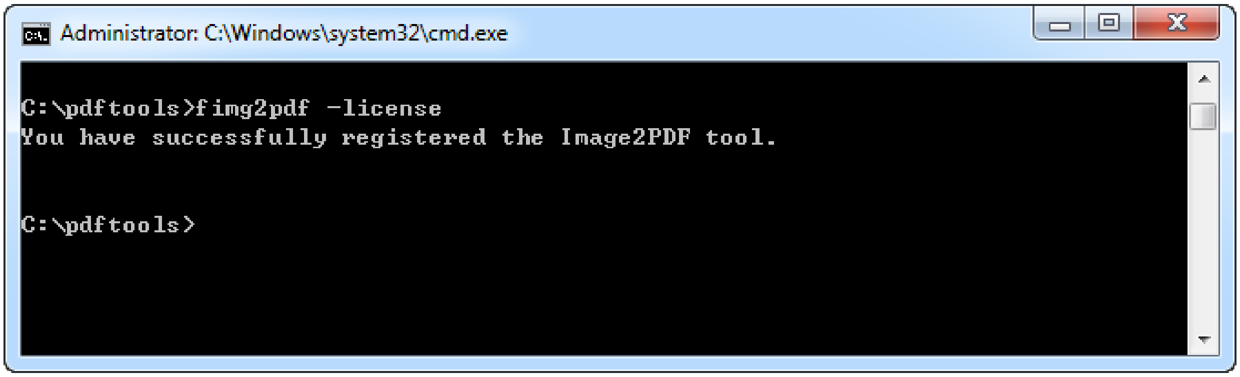
```
C:\Users\Administrator>cd C:\pdftools

C:\pdftools>fimg2pdf -register 40G01-01000-61000-YVAG9-S1GON-2ERL2 jessie
You have successfully activated the purchased modules.

C:\pdftools>
```

Here, "jessie" is the name of the licensee you designated. After activation, a key file named "**ftlkey.txt**" will be generated in the installed path and the contents are shown in the following figure.



```
[Foxit License]
SN=FTL342ERL2
Modules=Image2PDF
Users=1
Licensee=jessie
LicenseDate=2015/06/10
Sign=30339D43F8E5130ADBC7B2CDFC3C8D1EB0FF4B5B0EAF03C3CF6E1A82452E7A0744CBBEDB
10BB2278
Restrictions=Ver:1.x,Platform:Windows,CPU:4,md5:A570D43426196C3ACBF48925F7DDF0240139
FD9A547AD67E55497D523D51900B
```

Then you can run "-fimg2pdf -license" in command line window to check the license information as follows.



```
C:\pdftools>fimg2pdf -license
You have successfully registered the Image2PDF tool.


C:\pdftools>
```

2.4 Foxit PDF Toolkit Uninstallation

If you want to uninstall the Foxit PDF Toolkit, all you need to do is to delete the installed folder.

3 Command Line Usage

3.1 Image2PDF

3.1.1 Basic Syntax

fimg2pdf *<-i <srcfile/srcfolder>> <-o <destfile/destfolder>> [-t <thread>] [-r [recursion]] [[-width <PDF width>] [-height <PDF height>]] [-sp <password>] [-b] [-dpi <resolution>] [-margin <left [top right bottom]>] [-log <logfile>] [-l <level>]*

fimg2pdf *-register <code> <licensee>*

fimg2pdf *-license*

fimg2pdf *-version/-v*

fimg2pdf *-help/-h*

Note:

- <> required
- [ ] optional
- A space is needed between the command line argument and the value

Only the *<-i <srcfile/srcfolder>>* and *<-o <output>>* arguments are actually required. All others are optional which are available for controlling the output PDF files as desired. The arguments could be given in any order. Full details on each are explained in the following section.

3.1.2 Command Line Summary

Option	Parameter	Description
-i	*<-i <string>>* *e.g.* *-i c:\input\1.jpg* *-i "c:\input\1.jpg" "c:\input\2.tif"* *-i c:\input* *-i "c:\input\*.jpg"*	Specifies the input file to be converted. - The input can be a single image file (.bmp, .png, .jpg, .jpx, .gif, .tif, .tiff), multiple image files, or a single folder. - The file name can contain the wildcard characters (*). For example, use *.tiff to include all TIFF image files in a given folder. **Note** Wildcard characters (*.*) is not supported currently.

Option	Parameter	Description
-o	*<-o <string>>* *e.g.* *-o d:\output\one.pdf* *-o d:\output*	Specifies the path of the output PDF file or folder. - If specify a path of a PDF file, (e.g. *-o d:\output\one.pdf*), all input image files will be combined into one PDF file. - If specify a path of a folder, (e.g. *-o d:\output*), every input image file will be converted into individual PDF files. **Note** The specified output path must already exist.
-t	*[-t <integer>]* *e.g.* *-t 1* *-t 2*	Specifies the number of CPU threads to use. The default value is 1.
-r	*[-r [integer]]* *e.g.* *-r* *-r 0* *-r 1* *-r 2* …	Specifies the number of layers to recurse when the input is a folder. The default value is 1 (search only the current folder, no sub-folders). - **-r 0 <-r>**: searches the full folders. - **-r 1**: searches only the current folder. - **-r 2**: searches the current folder and its sub-folders … **Note** The input folder will be skipped if it is secured and the messages will be output.
-sp	*[-sp <string>]*	Sets the document open password of the output PDF as "string". By default, there is no password.
-width	*[-width <points>]*	Sets the page width of the output PDF file in points. Default: Width of input image. **Note** The **–width** and **–height** options must be used together and the setting value must be greater than 0.
-height	*[-height <points>]*	Sets the page height of the output PDF file in points. Default: Height of input image. **Note** The **–width** and **–height** options must be used together and the setting value must be greater than 0.
-b		Uses the filename of the image(s) to create bookmark(s) for the output PDF.

Option	Parameter	Description
-dpi	*[-dpi <integer>]* *e.g.* *-dpi 0* *-dpi 1* *-dpi 300*	Specifies DPI (Dots Per Inch) resolution of the output PDF file. The default value is 72. - **-dpi 0**: uses the default image width and height information. - **-dpi 1**: uses the DPI value of the input image. - **-dpi <integer>**: sets the DPI resolution as the given integer. **Note** - If the program failed to get the DPI value of the input image when the (**-dpi**) is set to 1, the DPI value will be 96. - The -weight, -height and -dpi options should not be used together. ♦ If you only use -width and -dpi, or -height and -dpi, the width or height setting will be omitted. ♦ If you use -width, -height and -dpi, the dpi setting will be omitted.
-margin	*[-margin <points [points points points]>]* *-margin <left [top right bottom]>* *e.g.* *-margin 20* *-margin 10 20* *-margin 10 20 0* *-margin 10 20 0 40*	Sets size of margin for each PDF page in points. Default value for each margin: 0. Allowable values: 0-size of page in points; in addition, the sum of the left and right values must be less than the width of the page, and the sum of the top and bottom values must be less than the height of the page. ***-margin left top right bottom*** **-margin 20**: set the left margin to 20 points. **-margin 10 20**: set the left margin to 10 points and the top margin to 20 points. **-margin 10 20 0**: set the left margin to 10 points, the top margin to 20 points, and the right margin to 0 points. **-margin 10 20 0 40**: set the left margin to 10 points, the top margin to 20 points, the right margin to 0 points, and the bottom margin to 40 points.

Option	Parameter	Description
-log	*[-log <string>]* *e.g. -log d:\a.log*	Writes log information into a logfile at the specified existing path.
-l	*[-l <integer>]* *e.g.* *-l 1* *-l 2* *-l 3* *-l 4*	Sets the log level. The default is 4. • **-l 1**: logs only messages concerning program crash. • **-l 2**: logs failure messages concerning the errors caused during execution or returned from underlying libraries, as well as those for level 1. • **-l 3**: logs warning messages concerning that PDF files are overwritten, as well as those for level 2. • **-l 4**: logs informational messages, as well as those for level 3. **Note** The (**-l**) is valid only when (**-log**) is used.
-register	*[-register <String> <String>]* *-register <code> <licensee>*	Registers the command line tool. ▪ **<code>**: the activation code from Foxit. ▪ **<licensee>**: the Licensee name designated by users.
-help/-h		Prints the usage information.
-version/-v		Prints the version information.
-license		Prints the license information.

3.1.3 Basic Usage

3.1.3.1 Input and Output

a) Input (-i)

- The input files should be a single image file, multiple image files, or a single folder. It doesn't allow users to input multiple folders or a mixture of folders and image files. For example:

-i c:\input\1.jpg	(a single image file)
-i “c:\input\1.jpg” “c:\input\2.tif”	(multiple image files)
-i c:\input	(a single folder)

Note *If the input files are multiple image files, please better enclose each input path in double quotation marks (" "). In the manual, we all add (" ") when the input files are multiple image files.*

- It supports relative paths, that is, if the input files are in the current working folder, you can input only the name of the image files or folder, instead of an absolute path. For example:

-i test\3.bmp	("test\3.bmp" is in the current working folder)
-i test	("test" folder is in the current working folder)

- It also supports wildcard characters, which are used to process multiple image files with specified formats. For example:

-i "c:\input\*.jpg"	(Only convert images with JPG format)
-i "c:\input\*.jpg" "c:\input\*.tif"	(Only convert images with JPG and TIF formats)
-i "test*.png"	(Only convert images with PNG format)

Note *When using wildcard characters in the input files, please better enclose the input files in double quotation marks (" "). Also, in this manual, we all add (" ") when the input files use wildcard characters.*

b) Output (-o)

- There are two output formats:
  - Combine multiple image files into one PDF file
  - Convert each image file into individual PDF files
- If you want to combine multiple image files into one PDF file, you should specify the output path of a PDF file, or if you want to convert each image file into individual PDF files, you should specify the output path of a folder. For example:

-o d:\output\one.pdf	(Combine multiple image files into one PDF file)
-o d:\output	(Convert each image file into individual PDF files)

Note *The specified output path must already exist.*

- The output also supports relative paths. If the specified output location is in the current working folder, you can input only the name of the PDF file or output folder, instead of an absolute path. For example:

```
-o output\one.pdf                  (“output” folder is in the current working folder)
-o output                          (“output” folder is in the current working folder)
```

- When the input is one or more image files, or including wildcard characters, the output should be a single PDF file. For example, the first following line combines the “3.bmp” and “4.gif” image files into “out.pdf” file, in which the “test” and “output” folders are both in the current working folder. And the second line combines all the “.jpg” and “.tif” image files in “c:\input” folder into “out.pdf” in “d:\output” folder.

```
fimg2pdf -i “test\3.bmp” “test\4.gif” -o output\out.pdf
fimg2pdf -i “c:\input\*.jpg” “c:\input\*.tif” –o d:\output\out.pdf
```

Usage Examples

1) Convert each image file into individual PDF files:

```
fimg2pdf -i test -o output          (“test” and “output” folders are both in the current working folder)
fimg2pdf -i c:\input -o d:\output
```

2) Combine multiple image files into one PDF file:

```
fimg2pdf -i “test\3.bmp” “test\4.gif” -o output\one.pdf
fimg2pdf -i c:\input\1.jpg -o d:\output\one.pdf
fimg2pdf -i “c:\input\1.jpg” “c:\input\2.tif” -o d:\output\one.pdf
fimg2pdf -i c:\input -o d:\output\one.pdf
fimg2pdf -i “c:\input\*.jpg” -o d:\output\one.pdf
```

3.1.3.2 Recursion Depth of Sub-folders

- The optional argument (**-r**) indicates the recursion depth of sub-folders, which is valid only when the input is a folder or a path including wildcard characters like “c:\input\*.jpg”. By default, the recursion depth is 1, that is, the sub-folders are not processed. For more details about this argument, please refer to section 3.1.2 “Command Line Summary”.

Usage Examples

1) Search the full sub-folders (-r or -r 0)

```
fimg2pdf -i test -o output -r 0
fimg2pdf -i c:\input -o d:\output -r 0
```

```
fimg2pdf -i “c:\input\*.jpg” -o d:\output\one.pdf -r 0
fimg2pdf -i test -o output -r
fimg2pdf -i c:\input -o d:\output -r
fimg2pdf -i “c:\input\*.jpg” -o d:\output\one.pdf -r
```

2) Search only the current folder (-r 1)

```
fimg2pdf -i test -o output -r 1
fimg2pdf -i c:\input -o d:\output -r 1
fimg2pdf -i “c:\input\*.jpg” -o d:\output\one.pdf -r 1
```

Note *If you don’t use this argument, it searches the current folder by default. For example:*

```
fimg2pdf -i test -o output
fimg2pdf -i c:\input -o d:\output
fimg2pdf -i “c:\input\*.jpg” -o d:\output\one.pdf
```

3) Search the current folder and its sub-folders (-r 2)

```
fimg2pdf -i test -o output -r 2
fimg2pdf -i c:\input -o d:\output -r 2
fimg2pdf -i “c:\input\*.jpg” -o d:\output\one.pdf -r 2
```

3.1.3.3 Multi-thread Support

- The optional argument (**-t**) indicates the number of threads, which are used to speed up the batch conversion by making full use of the CPU. By default, the number of the threads is 1.

  Note *It is recommended that you should set the number according to the CPU configuration of your computer.*

Usage Example

1) Set the number of threads to 3 (-t 3)

```
fimg2pdf -i test -o output -t 3
fimg2pdf -i c:\input -o d:\output -t 3
fimg2pdf -i “c:\input\*.jpg” -o d:\output\one.pdf -t 3
```

3.1.3.4 PDF/Page Settings

a) Password setting (-sp)

- The optional argument (**-sp**) is used to set the open password to the output PDF file. By default, the output PDF file has no open password.

Usage Example

1) Set the open password to “welcome” for the output PDF files (-sp welcome)

```
fimg2pdf -i “c:\input\1.jpg” “c:\input\2.tif” -o d:\output\one.pdf -sp welcome
fimg2pdf -i c:\input –o d:\output -sp welcome
fimg2pdf -i “c:\input\*.jpg” –o d:\output\one.pdf -sp welcome
```

b) Page size setting (-width, -height)

- The optional arguments (**-width**) and (**-height**) are used to set the page width and height for the output PDF file, in points.

Note *The –width and –height options must be used together and the setting value must be greater than 0.*

Usage Example

1) Set the page width and height to 400 and 300 for the output PDF file (-width 400 -height 300)

```
fimg2pdf -i “c:\input\1.jpg” “c:\input\2.tif” -o d:\output\one.pdf -width 400 -height 300
fimg2pdf -i c:\input -o d:\output -width 400 -height 300
fimg2pdf -i “c:\input\*.jpg” -o d:\output\one.pdf -width 400 -height 300
```

c) Resolution (DPI) setting (-dpi)

- The optional argument (**-DPI**) is used to set the resolution for the output PDF file. By default, the DPI (Dots Per Inch) is 72, which is the typical resolution for web images. For more details about this argument, please refer to section 3.1.2 “Command Line Summary”.

Note

- *If the program failed to get the DPI value of the input image when the (**-dpi**) is set to 1, the DPI value will be 96.*

- *The options -weight, -height and -dpi should not be used together.*
  - If you only use -width and -dpi, or -height and -dpi, the width or height setting will be omitted.
  - If you use -width, -height and -dpi, the dpi setting will be omitted.

Usage Examples

1) Use the default image width and height information (-dpi 0)

```
fimg2pdf -i “c:\input\1.jpg” “c:\input\2.tif” -o d:\output\one.pdf -dpi 0
fimg2pdf -i c:\input –o d:\output -dpi 0
fimg2pdf -i “c:\input\*.jpg” –o d:\output\one.pdf -dpi 0
```

2) Use the DPI information of the original image (-dpi 1)

```
fimg2pdf -i “c:\input\1.jpg” “c:\input\2.tif” -o d:\output\one.pdf -dpi 1
fimg2pdf -i c:\input –o d:\output -dpi 1
fimg2pdf -i “c:\input\*.jpg” –o d:\output\one.pdf -dpi 1
```

3) Set the DPI to 300 for the output PDF file (-dpi 300)

```
fimg2pdf -i “c:\input\1.jpg” “c:\input\2.tif” -o d:\output\one.pdf -dpi 300
fimg2pdf -i c:\input –o d:\output -dpi 300
fimg2pdf -i “c:\input\*.jpg” –o d:\output\one.pdf -dpi 300
```

d) Bookmark (-b)

- The optional argument (-b) is used to create bookmarks for the output PDF file. The name of the bookmarks is the name of the images.

Usage Example

1) Create bookmarks for the output PDF file (-b)

```
fimg2pdf -i “c:\input\1.jpg” “c:\input\2.tif” -o d:\output\one.pdf -b
fimg2pdf -i c:\input -o d:\output\one.pdf -b
fimg2pdf -i “c:\input\*.jpg” -o d:\output\one.pdf -b
```

e) Margin setting (-margin)

- The optional argument (-margin) is used to set size of margin for each PDF page in points. By default, the output PDF page has no margin. For more details about this argument, please refer to section 3.1.2 "Command Line Summary".

Note *The sum of the left and right values must be less than the width of the page, and the sum of the top and bottom values must be less than the height of the page.*

Usage Example

1) Set the left margin to 20 points (-margin 20)

```
fimg2pdf -i "c:\input\1.jpg" "c:\input\2.tif" -o d:\output\one.pdf -margin 20
fimg2pdf -i c:\input -o d:\output\one.pdf -margin 20
fimg2pdf -i "c:\input\*.jpg" -o d:\output\one.pdf -margin 20
```

2) Set the left margin to 20 points, and the top margin to 10 points (-margin 20 10)

```
fimg2pdf -i "c:\input\1.jpg" "c:\input\2.tif" -o d:\output\one.pdf -margin 20 10
fimg2pdf -i c:\input -o d:\output\one.pdf -margin 20 10
fimg2pdf -i "c:\input\*.jpg" -o d:\output\one.pdf -margin 20 10
```

3) Set the left margin to 10 points, the top margin to 10 points and the right margin to 30 points (-margin 10 10 30)

```
fimg2pdf -i "c:\input\1.jpg" "c:\input\2.tif" -o d:\output\one.pdf -margin 10 10 30
fimg2pdf -i c:\input -o d:\output\one.pdf -margin 10 10 30
fimg2pdf -i "c:\input\*.jpg" -o d:\output\one.pdf -margin 10 10 30
```

4) Set the left margin to 10 points, the top margin to 10 points, the right margin to 30 points and the bottom margin to 20 points (-margin 10 10 30 20)

```
fimg2pdf -i "c:\input\1.jpg" "c:\input\2.tif" -o d:\output\one.pdf -margin 10 10 30 20
fimg2pdf -i c:\input -o d:\output\one.pdf -margin 10 10 30 20
fimg2pdf -i "c:\input\*.jpg" -o d:\output\one.pdf -margin 10 10 30 20
```

3.1.3.5 Other Settings

a) Log file (-log<logfile> -l<log level>)

- The optional argument (**-log**) indicates the path of logfile, where the log message is placed. The (**-l**) indicates the log level. It is valid only when (**-log**) is used. By default, the log level is 4. For more details about this argument, please refer to section 3.1.2 "Command Line Summary".

Usage Example

1) Save the log file to "d:\output\image2pdf.log" and set the log level to 3 (-log d:\output\image2pdf.log -l 3)

```
fimg2pdf -i c:\input -o d:\output -log d:\output\image2pdf.log -l 3
```

b) Register information (-register <code> <licensee>)

- The optional argument (**-register**) is used to register the command line tool. The **<code>** is the activation code from Foxit and **<licensee>** is the licensee name designated by users.

Usage Example

1) Register the image2pdf tool with the code "40G01-01000-50000-2JCTF-3DV20-RZOF4" and the licensee "jessie" (-register 40G01-01000-50000-2JCTF-3DV20-RZOF4 jessie)

```
fimg2pdf -register 40G01-01000-50000-2JCTF-3DV20-RZOF4 jessie
```

c) License information (-license)

- The optional argument (**-license**) is used to print the license information.

Usage Example

1) Print the license information (-license)

```
fimg2pdf -license
```

d) Version information (-version/-v)

- The optional argument (-version/-v) is used to print the version information.

Usage Example

1) Print the version information (-version/-v)

```
fimg2pdf -version
fimg2pdf -v
```

e) Help information (-help/-h)

- The optional argument (**-help/-h**) is used to print the usage information.

Usage Example

1) Print the usage information (-help/-h)

```
fimg2pdf -help
fimg2pdf -h
```

3.2 Office2PDF

3.2.1 Basic Syntax

foffice2pdf *<-i <srcfile/srcfolder>> <-o <destfile/destfolder>> [-t <threads>] [-r [recursion]] [-op <password>] [-b <bookmark level>] [-pdfa] [-scale] [-log <logfile>] [-l <level>]*

foffice2pdf *-register <code> <licensee>*

foffice2pdf *-license*

foffice2pdf *-version/-v*

foffice2pdf *-help/-h*

Note:

- <> required
- [ ] optional
- A space is needed between the command line argument and the value

Only the <*-i <srcfile/srcfolder*>> and <*-o <output*>> arguments are actually required. All others are optional which are available for controlling the output PDF files as desired. The arguments could be given in any order. Full details on each are explained in the following section.

3.2.2 Command Line Summary

Option	Parameter	Description
-i	*<-i <string>>* *e.g.* *-i c:\input\1.doc* *-i c:\input* *-i "c:\input\*.ppt"*	Specifies the input file to be converted. ▪ The input can be a single Microsoft Office file (.doc, .docx, .xls, .xlsx, .ppt, .pptx) or a single folder. ▪ The file name can contain the wildcard characters (*). For example, use *.doc to include all Doc files in a given folder. **Note** The wildcard characters (*.*) is not supported currently.
-o	*<-o <string>>* *e.g.* *-o D:\output\1.pdf* *-o D:\output*	Specifies the path of the output PDF file or folder. ▪ If the input is a single Microsoft Office file, the output should be a single PDF file, (e.g. *-o D:\output\1.pdf*). ▪ If the input is a folder, the output should be a

Option	Parameter	Description
		folder, (e.g. *-o D:\output*). **Note** The specified output path must already exist.
-t	*[-t <integer>]* *e.g.* *-t 1* *-t 2*	Specifies the number of CPU threads to use. The default value is 1.
-r	*[-r [integer]]* *e.g.* *-r* *-r 0* *-r 1* *-r 2* …	Specifies the number of layers to recurse when the input is a folder. The default value is 1 (search only the current folder, no sub-folders). • **-r 0 <-r>**: searches the full folders. • **-r 1**: searches only the current folder. • **-r 2**: searches the current folder and its sub-folders … **Note** The input Microsoft Office file or folder will be skipped if it is secured and the messages will be output.
-op	*[-op<string>]*	Specifies the open password for the input file. Not required if the input file is not secured. **Note** The output PDF file will not retain the open password from the input documents.
-pdfa		Specifies that the output PDF file(s) should be compliant with the PDF/A standard.
-b	*[-b <integer>]* *e.g.* *-b 0* *-b 1* *-b 2*	Creates bookmarks for the output PDF file. If not set, there are no PDF bookmarks. • **-b 0**: Not to create PDF bookmarks. • **-b 1**: Create PDF bookmarks using headings of a Microsoft Word file. • **-b 2**: Create PDF bookmarks using bookmarks of a Microsoft Word file. **Note** This argument is valid only when the input is Microsoft Word file.

Option	Parameter	Description
-scale	*[-scale<integer>]* *e.g.* *-scale 0* *-scale 1* *-scale 2* *-scale 3*	Specifies the conversion mode for Microsoft Excel file. The default is 1. • **-scale 0**: No scaling. Convert sheets at their actual size. • **-scale 1**: Fit all columns on one page. Scale every sheet so that it is one page wide. • **-scale 2**: Fit all rows on one page. Scale every sheet so that it is one page high. • **-scale 3**: Fit sheet on one page. Scale every sheet so that it fits on one page. **Note** This argument is supported on the version higher than Microsoft Office 2007 and it is valid only for Microsoft Excel file.
-log	*[-log <string>]* *e.g. -log d:\a.log*	Writes log information into a log file at the specified existing path.
-l	*[-l <integer>]* *e.g.* *-l 1* *-l 2* *-l 3* *-l 4*	Sets the log level. The default is 4. • -l 1: logs only messages concerning program crash. • -l 2: logs failure messages concerning the errors caused during execution or returned from underlying libraries, as well as those for level 1. • -l 3: logs warning messages concerning that PDF files are overwritten, as well as those for level 2. • -l 4: logs informational messages, as well as those for level 3. **Note** The (**-l**) is valid only when (**-log**) is used.
-register	*[-register <String> <String>]* *-register <code> <licensee>*	Registers the command line tool. ▪ **<code>**: the activation code from Foxit. ▪ **<licensee>**: the Licensee name designated by users.
-help/-h		Prints the usage information.

Option	Parameter	Description
-version/-v		Prints the version information.
-license		Prints the license information.

3.2.3 Basic Usage

3.2.3.1 Input and Output

a) Input (-i)

- The input file should be a single Microsoft Office file or a single folder. It doesn't allow users to input multiple Office files or folders or a mixture of folders and Office files. For example:

-i c:\input\1.doc	(a single Microsoft Office file)
-i c:\input	(a single folder)

- It supports relative paths, that is, if the input file is in the current working folder, you can input only the name of the Microsoft Office file or folder, instead of an absolute path. For example:

-i test\2.xls	("test\2.xls" is in the current working folder)
-i test	("test" folder is in the current working folder)

- It also supports wildcard characters, which are used to process multiple office files with specified formats. For example:

-i "c:\input\*.doc"	(Only convert Office files with DOC format)
-i "test*.xls"	(Only convert Office files with XLS format)

Note *When using wildcard characters in the input files, please better enclose the input files in double quotation marks (" "). In this manual, we all add (" ") when the input files use wildcard characters.*

b) Output (-o)

- If the input file is a single Microsoft Office file, you should specify the output path of a PDF file. Or if the input file is a single folder, you should specify the output path of a folder. For example:

```
-o d:\output\ 1.pdf
-o d:\output
```

Note *The specified output path must already exist.*

- The output also supports relative paths. If the specified output location is in the current working folder, you can input only the name of the PDF file or output folder, instead of an absolute path. For example:

-o output\2.pdf	("output\2.pdf" is in the current working folder)
-o output	("output" folder is in the current working folder)

Usage Examples

1) Convert Microsoft Office documents to PDF files:

```
foffice2pdf -i c:\input -o d:\output
foffice2pdf -i test\2.xls -o output\2.pdf
foffice2pdf -i test -o output
```

3.2.3.2 Recursion Depth of Sub-folders

- The optional argument (**-r**) indicates the recursion depth of sub-folders, which is valid only when the input is a folder or a path including wildcard characters like "c:\input\*.doc". By default, the recursion depth is 1, that is, the sub-folders are not processed. For more details about this argument, please refer to section 3.2.2 "Command Line Summary".

Usage Examples

1) Search the full sub-folders (-r or -r 0)

```
foffice2pdf -i c:\input -o d:\output -r
foffice2pdf -i test -o output -r
foffice2pdf -i "c:\input\*.doc" -o d:\output -r
foffice2pdf -i c:\input -o d:\output -r 0
foffice2pdf -i test -o output -r 0
foffice2pdf -i "c:\input\*.doc" -o d:\output -r 0
```

2) Search only the current folder (-r 1)

```
foffice2pdf -i c:\input -o d:\output -r 1
foffice2pdf -i test -o output -r 1
foffice2pdf -i "c:\input\*.doc" -o d:\output -r 1
```

Note *If you don't use this argument, it searches the current folder by default. For example:*

```
foffice2pdf -i c:\input -o d:\output
foffice2pdf -i test -o output
foffice2pdf -i “c:\input\*.doc” -o d:\output
```

3) Search the current folder and its sub-folders (-r 2)

```
foffice2pdf -i c:\input -o d:\output -r 2
foffice2pdf -i test -o output -r 2
foffice2pdf -i “c:\input\*.doc” -o d:\output -r 2
```

3.2.3.3 Multi-thread Support

- The optional argument (**-t**) indicates the number of threads, which are used to speed up the batch conversion by making full use of the CPU. By default, the number of the threads is 1.

  Note *It is recommended that you should set the number according to the CPU configuration of your computer.*

Usage Example

1) Set the number of threads to 3 (-t 3)

```
foffice2pdf -i c:\input -o d:\output -t 3
foffice2pdf -i test -o output -t 3
foffice2pdf -i “c:\input\*.doc” -o d:\output -t 3
```

3.2.3.4 Open Password

- The optional argument (**-op**) indicates the open password for the input Microsoft Office file which is secured. It is not required if the input file is not secured.

  Note

  - *The argument (-op) is only valid for the password-protected files. If the input file is not secured, the typed open password will be ignored.*
  - *The output PDF file will not retain the open password from the input documents.*

Usage Example

1) Specify the open password for an input Office file (-op 123)

```
fofffice2pdf -i c:\input\1.doc -o d:\output\1.pdf -op 123
fofffice2pdf -i test\2.xls -o output\2.pdf -op 123
```

2) Specify the open password for all input Office files which have the same open password (-op 123)

```
fofffice2pdf -i c:\input -o d:\output -op 123
fofffice2pdf -i test -o output -op 123
fofffice2pdf -i "c:\input\*.doc" -o d:\output -op 123
```

Note *Version 1.0 only supports typing one value for the argument (**-op**). That is, it will only convert the Office files matched the typed open password at a time. If the input Office files have different open passwords, you should do the conversion in multiple times.*

3.2.3.5 PDF/A Support

- The optional argument (**-pdfa**) is used to specify the output PDF file to be compliant with PDF/A Standard. PDF/A is an ISO standardized version of the PDF specialized for the digital preservation of electronic documents, which defines provisions for long-term archiving of the electronic documents.

  Note *Version 1.0 only supports PDF/A-1b Standard. For more details about PDF/A-1b, please refer to PDF Reference.*

Usage Example

1) Specify the output PDF file to be compliant with PDF/A Standard (-pdfa)

```
fofffice2pdf -i c:\input\1.doc -o d:\output\1.pdf -pdfa
fofffice2pdf -i c:\input -o d:\output -pdfa
fofffice2pdf -i test -o output -pdfa
fofffice2pdf -i "c:\input\*.doc" -o d:\output -pdfa
```

3.2.3.6 Bookmark

- The optional argument (**-b**) is used to create bookmarks for the output PDF file. If not set, there are no PDF bookmarks by default. For more details about this argument, please refer to the section 3.2.2 "Command Line Summary".

  Note *This argument is valid only when the input is Microsoft Word files.*

Usage Example

1) Create PDF bookmarks using headings of a Microsoft Word file (-b 1)

```
foffice2pdf -i c:\input\1.doc -o d:\output\1.pdf -b 1
foffice2pdf -i c:\input -o d:\output -b 1
foffice2pdf -i test -o output -b 1
foffice2pdf -i “c:\input\*.doc” -o d:\output -b 1
```

2) Create PDF bookmarks using bookmarks of a Microsoft Word file (-b 2)

```
foffice2pdf -i c:\input\1.doc -o d:\output\1.pdf -b 2
foffice2pdf -i c:\input -o d:\output -b 2
foffice2pdf -i test -o output -b 2
foffice2pdf -i “c:\input\*.doc” -o d:\output -b 2
```

3.2.3.7 Scale

- The optional argument (**-scale**) is used to specify the conversion mode for Microsoft Excel file. For more details about this argument, please refer to the section 3.2.2 “Command Line Summary”.

  Note *This argument is supported on the version higher than Microsoft Office 2007 and it is valid only for Microsoft Excel file.*

Usage Example

1) Convert sheets at their actual size (-scale 0)

```
foffice2pdf -i c:\input\1.xlsx -o d:\output\1.pdf -scale 0
foffice2pdf -i c:\input -o d:\output -scale 0
foffice2pdf -i test -o output -scale 0
foffice2pdf -i “c:\input\*.xls” -o d:\output -scale 0
```

2) Fit all columns on one page (-scale 1)

```
foffice2pdf -i c:\input\1.xlsx -o d:\output\1.pdf -scale 1
foffice2pdf -i c:\input -o d:\output -scale 1
foffice2pdf -i test -o output -scale 1
foffice2pdf -i “c:\input\*.xls” -o d:\output -scale 1
```

3) Fit all rows on one page (-scale 2)

```
foffice2pdf -i c:\input\1.xlsx -o d:\output\1.pdf -scale 2
```

```
foffice2pdf -i c:\input -o d:\output -scale 2
foffice2pdf -i test -o output -scale 2
foffice2pdf -i “c:\input\*.xls” -o d:\output -scale 2
```

4) Fit sheet on one page (-scale 3)

```
foffice2pdf -i c:\input\1.xlsx -o d:\output\1.pdf -scale 3
foffice2pdf -i c:\input -o d:\output -scale 3
foffice2pdf -i test -o output -scale 3
foffice2pdf -i “c:\input\*.xls” -o d:\output -scale 3
```

3.2.3.8 Other Optional Arguments

a) Log file (-log<logfile> -l<log level>)

- The optional argument (**-log**) indicates the path of logfile, where the log message is placed. The (**-l**) indicates the log level. It is valid only when (**-log**) is used. By default, the log level is 4. For more details about this argument, please refer to section 3.2.2 “Command Line Summary”.

Usage Example

1) Save the log file to “d:\output\office2pdf.log” and set the log level to 3 (-log d:\output\office2pdf.log -l 3)

```
foffice2pdf -i c:\input\1.doc -o d:\output\1.pdf -log d:\output\office2pdf.log -l 3
```

b) Register information (-register <code> <licensee>)

- The optional argument (**-register**) is used to register the command line tool. The **<code>** is the activation code from Foxit and **<licensee>** is the licensee name designated by users.

Usage Example

1) Register the office2pdf tool with the code “40G01-01000-50000-2JCTF-3DV20-RZOF4” and the licensee “jessie” (-register 40G01-01000-50000-2JCTF-3DV20-RZOF4 jessie)

```
foffice2pdf -register 40G01-01000-50000-2JCTF-3DV20-RZOF4 jessie
```

c) License information (-license)

- The optional argument (**-license**) is used to print the license information.

Usage Example

1) Print the license information (-license)

```
foffice2pdf -license
```

d) Version information (-version/-v)

- The optional argument (-version/-v) is used to print the version information.

Usage Example

1) Print the version information (-version/-v)

```
foffice2pdf -version
foffice2pdf -v
```

e) Help information (-help/-h)

- The optional argument (**-help/-h**) is used to print the usage information.

Usage Example

1) Print the usage information (-help/-h)

```
foffice2pdf -version
foffice2pdf -v
```

3.3 PDFWatermark

3.3.1 Basic Syntax

fpdfwm *<-i <srcfile/srcfolder>> <-o <destfile/destfolder>> <-conf <xmlfile>> [-t <threads>] [-r [recursion]] [-op <password>] [-log <logfile>] [-l <level>]*
fpdfwm *-register <code> <licensee>*
fpdfwm *-license*
fpdfwm *-version/-v*
fpdfwm *-help/-h*

Note:

- <> required
- [ ] optional
- A space is needed between the command line argument and the value

Only the <*-i <srcfile/srcfolder>*>, <*-o <output>*> and <*-conf <xmlfile>*> arguments are actually required. All others are optional which are available for controlling the process as desired. The arguments could be given in any order. The XML file is a configuration file which was generated by the build-in Foxit Configuration Tool. Full details on each are explained in the following section.

3.3.2 Command Line Summary

Option	Parameter	Description
-i	*<-i <string>>* *e.g.* *-i c:\input\1.pdf* *-i c:\input*	Specifies the input file to be processed. ▪ The input can be a single PDF file or a single folder. ▪ The file name can contain the wildcard characters (*). For example, use *.pdf to include all PDF files in a given folder. **Note** Wildcard characters (*.*) is not supported currently.

Option	Parameter	Description
-o	*<-o <string>>* *e.g.* *-o D:\output\1_wm.pdf* *-o D:\output*	Specifies the path of the output PDF file or folder. ▪ If the input is a PDF file, the output should be a single PDF file, (e.g. *-o D:\output\1_wm.pdf*). ▪ If the input is a folder, the output should be a folder, (e.g. *-o D:\output*). **Note** The specified output path must already exist.
-conf	*<-conf <xmlfile>>*	Specifies the configuration file on the PDFWatermark tool. This file should be generated by the build-in Foxit Configuration Tool.
-t	*[-t <integer>]* *e.g.* *-t 1* *-t 2*	Specifies the number of CPU threads to use. The default value is 1.
-r	*[-r [integer]]* *e.g.* *-r* *-r 0* *-r 1* *-r 2* …	Specifies the number of layers to recurse when the input is a folder. The default value is 1 (search only the current folder, no sub-folders). • -r 0 <-r>: searches the full folders. • -r 1: searches only the current folder. • -r 2: searches the current folder and its sub-folders … **Note** The PDF file or folder will be skipped if it is secured and the message will be output.
-op	*[-op<string>]*	Specifies the open password for the input file. Not required if the input file is not secured. **Note** The output PDF file will retain the open password from the input documents.
-log	*[-log <string>]* *e.g. -log d:\a.log*	Writes log information into a logfile at the specified existing path.

Option	Parameter	Description
-l	*[-l <integer>]* *e.g.* *-l 1* *-l 2* *-l 3* *-l 4*	Sets the log level. The default is 4. • -l 1: logs only messages concerning program crash. • -l 2: logs failure messages concerning the errors caused during execution or returned from underlying libraries, as well as those for level 1. • -l 3: logs warning messages concerning that PDF files are overwritten, as well as those for level 2. • -l 4: logs informational messages, as well as those for level 3. **Note** The (**-l**) is valid only when (**-log**) is used.
-register	*[-register <String> <String>]* *-register <code> <licensee>*	Registers the command line tool. ▪ **<code>**: the activation code from Foxit. ▪ **<licensee>**: the Licensee name designated by users.
-help/-h		Prints the usage information.
-version/-v		Prints the version information.
-license		Prints the license information.

3.3.3 Basic Usage

3.3.3.1 Required Arguments

a) Input (-i)

- The input file should be a single PDF file or a single folder. It doesn’t allow users to input multiple PDF files or folders or a mixture of folders and PDF files. For example:

```
-i c:\input\1.pdf                  (a single PDF file)
-i c:\input                        (a single folder)
```

- It supports relative paths, that is, if the input file is in the current working folder, you can input only the name of the PDF file or folder, instead of an absolute path. For example:

-i test\2.pdf	("test\2.pdf" is in the current working folder)
-i test	("test" folder is in the current working folder)

- It also supports wildcard characters, which are used to process multiple PDF files. For example:

-i "c:\input\*.pdf"	(Only convert PDF files under "c:\input" folder)
-i "test*.pdf"	(Only convert PDF files under "test" folder)

Note *When using wildcard characters in the input files, please better enclose the input files in double quotation marks (" "). In this manual, we all add (" ") when the input files use wildcard characters.*

b) Output (-o)

- If the input file is a single PDF file, you should specify the output path of a PDF file. Or if the input file is a single folder, you should specify the output path of a folder. For example:

-o d:\output\ 1_wm.pdf	(a single PDF file)
-o d:\output	(a single folder)

Note *The specified output path must already exist.*

- The output supports relative path. If the specified output location is in the current working folder, you can input only the name of the PDF file or output folder, instead of an absolute path. For example:

-o output\2_wm.pdf	("output" folder is in the current working folder)
-o output	("output" folder is in the current working folder)

c) XML Configuration File (-conf)

- The XML configuration file argument (**-conf**) is required in the command line. The configuration file contains the setting information of watermark, which is generated by the build-in Foxit Configuration Tool (watermarkconfig.exe or watermarkconfig64.exe). For more details about watermark settings, please refer to section 4.1.1 "Watermark Settings". You should input the path of an XML file. For example:

```
-conf c:\conf_wm.xml
```

- It also supports relative paths. If the specified XML configuration file is in the current working folder, you can input only the name of the XML file, instead of an absolute path. For example:

-conf conf_wm.xml (conf_wm.xml is in the current working folder)

Usage Examples

1) Add watermark into PDF files:

```
fpdfwm -i test -o output -conf conf_wm.xml
fpdfwm -i c:\input\1.pdf -o d:\output\1_wm.pdf -conf c:\conf_wm.xml
```

3.3.3.2 Recursion Depth of Sub-folders

- The optional argument (**-r**) indicates the recursion depth of sub-folders, which is valid only when the input is a folder. By default, the recursion depth is 1, that is, the sub-folders are not processed. For more details about this argument, please refer to section 3.3.2 "Command Line Summary".

Usage Examples

1) Search the full sub-folders (-r or -r 0)

```
fpdfwm -i test -o output -conf conf_wm.xml -r
fpdfwm -i c:\input -o d:\output -conf c:\conf_wm.xml -r
fpdfwm -i test -o output -conf conf_wm.xml -r 0
fpdfwm -i c:\input -o d:\output -conf c:\conf_wm.xml -r 0
```

2) Search only the current folder (-r 1)

```
fpdfwm -i test -o output -conf conf_wm.xml -r 1
fpdfwm -i c:\input -o d:\output -conf c:\conf_wm.xml -r 1
```

Note *If you don't use this argument, it searches the current folder by default. For example:*

```
fpdfwm -i test -o output -conf conf_wm.xml
fpdfwm -i c:\input -o d:\output -conf c:\conf_wm.xml
```

3) Search the current folder and its sub-folders (-r 2)

```
fpdfwm -i test -o output -conf conf_wm.xml -r 2
fpdfwm -i c:\input -o d:\output -conf c:\conf_wm.xml -r 2
```

3.3.3.3 Multi-thread Support

- The optional argument (**-t**) indicates the number of threads, which are used to speed up the batch program by making full use of the CPU. By default, the number of the threads is 1.

  Note *It is recommended that you should set the number according to the CPU configuration of your computer.*

Usage Example

1) Set the number of threads to 3 (-t 3)

```
fpdfwm -i test -o output -conf conf_wm.xml -t 3
fpdfwm -i c:\input -o d:\output -conf c:\conf_wm.xml -t 3
```

3.3.3.4 Open Password

- The optional argument (**-op**) indicates the open password for the input PDF file which is secured. It is not required if the input file is not secured.

  Note *The output PDF file will retain the open password from the input documents.*

Usage Example

1) Specify the open password for an input PDF file (-op 123)

```
fpdfwm -i test\2.pdf -o output\2_wm.pdf -conf c:\conf_wm.xml -op 123
fpdfwm -i c:\input\1.pdf -o d:\output\1_wm.pdf -conf c:\conf_wm.xml -op 123
```

2) Specify the open password for all input PDF files which have the same open password (-op 123)

```
fpdfwm -i test -o output -conf conf_wm.xlm -op 123
fpdfwm -i c:\input -o d:\output -conf c:\conf_wm.xml -op 123
```

Note Version 1.0 only supports typing one value for the argument (**-op**). That is, it will only process the PDF files matched the typed open password at a time. If the input PDF files have different open passwords, you should do the work in multiple times.

3.3.3.5 Other Optional Arguments

a) Log file (-log<logfile> -l<log level>)

- The optional argument (**-log**) indicates the path of logfile, where the log message is placed. The (**-l**) indicates the log level. It is valid only when (**-log**) is used. By default, the log level is 4. For more details about this argument, please refer to section 3.3.2 "Command Line Summary".

Usage Example

1) Save the log file to "d:\output\watermark.log" and set the log level to 3 (-log d:\output\watermark.log -l 3)

```
fpdfwm -i c:\input -o d:\output -conf c:\conf_wm.xml -log d:\output\watermark.log -l 3
```

b) Register information (-register <code> <licensee>)

- The optional argument (**-register**) is used to register the command line tool. The **<code>** is the activation code from Foxit and **<licensee>** is the licensee name designated by users.

Usage Example

1) Register the pdfwatermark tool with the code "40G01-01000-50000-2JCTF-3DV20-RZOF4" and the licensee "jessie" (-register 40G01-01000-50000-2JCTF-3DV20-RZOF4 jessie)

```
fpdfwm -register 40G01-01000-50000-2JCTF-3DV20-RZOF4 jessie
```

c) License information (-license)

- The optional argument (**-license**) is used to print the license information.

Usage Example

1) Print the license information (-license)

```
fpdfwm -license
```

d) Version information (-version/-v)

- The optional argument (-version/-v) is used to print the version information.

Usage Example

1) Print the version information (-version/-v)

```
fpdfwm -version
fpdfwm -v
```

e) Help information (-help/-h)

- The optional argument (**-help/-h**) is used to print the usage information.

Usage Example

1) Print the usage information (-help/-h)

```
fpdfwm -help
fpdfwm -h
```

3.4 PDFHeaderFooter

3.4.1 Basic Syntax

fpdfhf *<-i <srcfile/srcfolder>> <-o <destfile/destfolder>> <-mode <operation mode>> [-conf <xmlfile>] [-t <threads>] [-r [recursion]] [-op <password>] [-overlay] [-log <logfile>] [-l <level>]*
fpdfhf *-register <code> <licensee>*
fpdfhf *-license*
fpdfhf *-version/-v*
fpdfhf *-help/-h*

Note:

- <> required
- [ ] optional
- A space is needed between the command line argument and the value

Only the <*-i <srcfile/srcfolder>*>, <*-o <output>*> and <*-mode <operation mode>*> arguments are actually required. The argument *[-conf <xmlfile>]* is required only when "*-mode*" is set to 1 or 2. All others are optional which are available for controlling the process as desired. The arguments could be given in any order. The XML file is a configuration file which was generated by the build-in Foxit Configuration Tool. Full details on each are explained in the following section.

3.4.2 Command Line Summary

Option	Parameter	Description
-i	*<-i <string>>* *e.g.* *-i c:\input\1.pdf* *-i c:\input*	Specifies the input file to be processed. ▪ The input can be a single PDF file or a single folder. ▪ The file name can contain the wildcard characters (*). For example, use *.pdf to include all PDF files in a given folder. **Note** Wildcard characters (*.*) is not supported currently.

Option	Parameter	Description
-o	*<-o <string>>* *e.g.* *-o D:\ output\1_hf.pdf* *-o D:\output*	Specifies the path of the output PDF file or folder. ▪ If the input is a PDF file, the output should be a single PDF file, (e.g. *-o D:\output\1_hf.pdf*). ▪ If the input is a folder, the output should be a folder, (e.g. *-o D:\output*). **Note** The specified output path must already exist.
-mode	*<-mode <integer>>* *e.g.* *-mode 1* *-mode 2* *-mode 3*	Specifies the mode to be used. • **-mode 1**: adds a new header/footer. An existing header/footer will be overlaid if the (**-overlay**) argument is set; otherwise, the document will not be modified if a header/footer already exists. • **-mode 2**: replaces an existing header/footer. If none exists in the document, a new header/footer will be added. • **-mode 3**: removes an existing header/footer. If none exists in the document, the document will not be modified.
-conf	*<-conf <xmlfile>>*	Specifies the configuration file on the PDFHeaderFooter tool. This file should be generated by the build-in Foxit Configuration Tool. **Note** This argument will only be used if (**-mode**) is set to 1 or 2.
-overlay		Overlays an existing header/footer. **Note** This argument (**-overlay**) is valid only when (**-mode**) is set to 1.
-op	*[-op <string>]*	Specifies the open password for the input file. Not required if the input file is not secured. **Note** The output PDF file will retain the open password from the input documents.

Option	Parameter	Description
-t	*[-t <integer>]* *e.g.* *-t 1* *-t 2*	Specifies the number of CPU threads to use. The default value is 1.
-r	*[-r [integer]]* *e.g.* *-r* *-r 0* *-r 1* *-r 2* …	Specifies the number of layers to recurse when the input is a folder. The default value is 1 (search only the current folder, no sub-folders). • -r 0 <-r>: searches the full folders. • -r 1: searches only the current folder. • -r 2: searches the current folder and its sub-folders … **Note** The PDF file or folder will be skipped if it is secured and the message will be output.
-log	*[-log <string>]* *e.g. -log d:\a.log*	Writes log information into a logfile at the specified existing path.
-l	*[-l <integer>]* *e.g.* *-l 1* *-l 2* *-l 3* *-l 4*	Sets the log level. The default is 4. • **-l 1**: logs only messages concerning program crash. • **-l 2**: logs failure messages concerning the errors caused during execution or returned from underlying libraries, as well as those for level 1. • **-l 3**: logs warning messages concerning that PDF files are overwritten, as well as those for level 2. • **-l 4**: logs informational messages, as well as those for level 3. **Note** The (**-l**) is valid only when (**-log**) is used.
-register	*[-register <String> <String>]* *-register <code> <licensee>*	Registers the command line tool. ▪ **<code>**: the activation code from Foxit. ▪ **<licensee>**: the Licensee name designated by users.
-help/-h		Prints the usage information.

Option	Parameter	Description
-version/-v		Prints the version information.
-license		Prints the license information.

3.4.3 Basic Usage

3.4.3.1 Required Arguments

a) Input (-i)

- The input file should be a single PDF file or a single folder. It doesn't allow users to input multiple PDF files or folders or a mixture of folders and PDF files. For example:

-i c:\input\1.pdf	(a single PDF file)
-i c:\input	(a single folder)

- It supports relative paths, that is, if the input file is in the current working folder, you can input only the name of the PDF file or folder, instead of an absolute path. For example:

-i test\2.pdf	("test\2.pdf" is in the current working folder)
-i test	("test" folder is in the current working folder)

- It also supports wildcard characters, which are used to process multiple PDF files. For example:

-i "c:\input\*.pdf"	(Only convert PDF files under "c:\input" folder)
-i "test*.pdf"	(Only convert PDF files under "test" folder)

Note *When using wildcard characters in the input files, please better enclose the input files in double quotation marks (" "). In this manual, we all add (" ") when the input files use wildcard characters.*

b) Output (-o)

- If the input file is a single PDF file, you should specify the output path of a PDF file. Or if the input file is a single folder, you should specify the output path of a folder. For example:

-o d:\output\1_hf.pdf	(a single PDF file)
-o d:\output	(a single folder)

Note *The specified output path must already exist.*

- The output supports relative paths. If the specified output location is in the current working folder, you can input only the name of the PDF file or output folder, instead of an absolute path. For example:

-o output\1_hf.pdf	("output" folder is in the current working folder)
-o output	("output" folder is in the current working folder)

c) Operation Mode (-mode)

- The operation mode argument (-mode) is required in the command line, which is used to specify the mode to process header/footer. For more details about this argument, please refer to section 3.4.2 "Command Line Summary".

d) XML Configuration File (-conf)

- The XML configuration file argument (**-conf**) is required in the command line when the argument (**-mode**) is set to 1 or 2. The configuration file contains the setting information of header/footer, which is generated by the build-in Foxit Configuration Tool (headerfooterconfig.exe or headerfooterconfig64.exe). For more details about header/footer settings, please refer to section 4.2.1 "Header/Footer Settings". You should input a path of an XML file. For example:

```
-conf c:\conf_hf.xml
```

- It also supports relative paths. If the specified XML configuration file is in the current working folder, you can input only the name of the XML file, instead of an absolute path. For example:

-conf conf_hf.xml	(conf_hf.xml is in the current working folder)

Usage Examples

1) Add a new header/footer into PDF files (-mode 1 -conf conf_hf.xml)

```
fpdfhf -i c:\input\1.pdf -o d:\output\1_hf.pdf -mode 1 -conf c:\conf_hf.xml
fpdfhf -i test -o output -mode 1 -conf conf_hf.xml
```

2) Replace the header/footer in PDF files (-mode 2 -conf conf_hf.xml)

```
fpdfhf -i c:\input\1.pdf -o d:\output\1_hf.pdf -mode 2 -conf c:\conf_hf.xml
fpdfhf -i test -o output -mode 2 -conf conf_hf.xml
```

3) Remove the header/footer in PDF files (-mode 3)

```
fpdfhf -i c:\input\1.pdf -o d:\output\1_hf.pdf -mode 3
fpdfhf -i test -o output -mode 3
```

3.4.3.2 Recursion Depth of Sub-folders

- The optional argument (**-r**) indicates the recursion depth of sub-folders, which is valid only when the input is a folder. By default, the recursion depth is 1, that is, the sub-folders are not processed. For more details about this argument, please refer to section 3.4.2 "Command Line Summary".

Usage Examples

1) Search the full sub-folders (-r or -r 0)

```
fpdfhf -i c:\input -o d:\output -mode 1 -conf c:\conf_hf.xml -r
fpdfhf -i test -o output -mode 2 -conf conf_hf.xml -r
fpdfhf -i test -o output -mode 3 -r
fpdfhf -i c:\input -o d:\output -mode 1 -conf c:\conf_hf.xml -r 0
fpdfhf -i test -o output -mode 2 -conf conf_hf.xml -r 0
fpdfhf -i test -o output -mode 3 -r 0
```

2) Search only the current folder (-r 1)

```
fpdfhf -i c:\input -o d:\output -mode 1 -conf c:\conf_hf.xml -r 1
fpdfhf -i test -o output -mode 2 -conf conf_hf.xml -r 1
fpdfhf -i test -o output -mode 3 -r 1
```

Note *If you don't use this argument, it searches the current folder by default. For example:*

```
fpdfhf -i c:\input -o d:\output -mode 1 -conf c:\conf_hf.xml
fpdfhf -i test -o output -mode 2 -conf conf_hf.xml
fpdfhf -i test -o output -mode 3
```

3) Search the current folder and its sub-folders (-r 2)

```
fpdfhf -i c:\input -o d:\output -mode 1 -conf c:\conf_hf.xml -r 2
fpdfhf -i test -o output -mode 2 -conf conf_hf.xml -r 2
fpdfhf -i test -o output -mode 3 -r 2
```

3.4.3.3 Multi-thread Support

- The optional argument (**-t**) indicates the number of threads, which are used to speed up the batch program by making full use of the CPU. By default, the number of the threads is 1.

  Note *It is recommended that you should set the number according to the CPU configuration of your computer.*

Usage Example

1) Set the number of threads to 3 (-t 3)

```
fpdfhf -i c:\input -o d:\output -mode 1 -conf c:\conf_hf.xml -t 3
fpdfhf -i test -o output -mode 2 -conf conf_hf.xml -t 3
fpdfhf -i test -o output -mode 3 -t 3
```

3.4.3.4 Overlay

- The optional argument (-overlay) is used to overlay an existing header/footer in PDF file. It is valid only when the argument (**-mode**) is set to 1 and the PDF file already contains a header/footer.

Usage Example

1) Overlay an existing header/footer in PDF file. (-overlay)

```
fpdfhf -i c:\input\1.pdf -o d:\output\1_hf.pdf -mode 1 -conf c:\conf_hf.xml -overlay
fpdfhf -i c:\input -o d:\output -mode 1 -conf c:\conf_hf.xml -overlay
fpdfhf -i test -o output -mode 1 -conf conf_hf.xml -overlay
```

3.4.3.5 Open Password

- The optional argument (**-op**) indicates the open password for the input PDF file which is secured. It is not required if the input file is not secured.

  Note *The output PDF file will retain the open password from the input documents.*

Usage Example

1) Specify the open password for an input PDF file (-op 123)

```
fpdfhf -i c:\input\1.pdf -o d:\output\1_hf.pdf -mode 1 -conf c:\conf_hf.xml -op 123
fpdfhf -i test\2.pdf -o output\2_hf.pdf -mode 2 -conf conf_hf.xml -op 123
fpdfhf -i test\3.pdf -o output\3_hf.pdf -mode 3 -op 123
```

2) Specify the open password for all input PDF files which have the same open password (-op 123)

```
fpdfhf -i c:\input -o d:\output -mode 1 -conf c:\conf_hf.xml -op 123
fpdfhf -i test -o output -mode 2 -conf conf_hf.xml -op 123
fpdfhf -i test -o output -mode 3 -op 123
```

Note *Version 1.0 only supports typing one value for the argument (**-op**). That is, it will only process the PDF files matched the typed open password at a time. If the input PDF files have different open passwords, you should do the work in multiple times.*

3.4.3.6 Other Optional Arguments

a) Log file (-log<logfile> -l<log level>)

- The optional argument (**-log**) indicates the path of logfile, where the log message is placed. The (**-l**) indicates the log level. It is valid only when (**-log**) is used. By default, the log level is 4. For more details about this argument, please refer to section 3.4.2 "Command Line Summary".

Usage Example

1) Save the log file to "d:\output\hf.log" and set the log level to 3 (-log d:\output\hf.log -l 3)

```
fpdfhf -i c:\input -o d:\output -mode 1 -conf c:\conf_hf.xml -log d:\output\hf.log -l 3
fpdfhf -i test -o output -mode 2 -conf conf_hf.xml -log d:\output\hf.log -l 3
fpdfhf -i test -o output -mode 3 -log d:\output\hf.log -l 3
```

b) Register information (-register <code> <licensee>)

- The optional argument (**-register**) is used to register the command line tool. The **<code>** is the activation code from Foxit and **<licensee>** is the licensee name designated by users.

Usage Example

1) Register the pdfheaderfooter tool with the code "40G01-01000-50000-2JCTF-3DV20-RZOF4" and the licensee "jessie" (-register 40G01-01000-50000-2JCTF-3DV20-RZOF4 jessie)

```
fpdfhf -register 40G01-01000-50000-2JCTF-3DV20-RZOF4 jessie
```

c) License information (-license)

- The optional argument (**-license**) is used to print the license information.

Usage Example

1) Print the license information (-license)

```
fpdfhf -license
```

d) Version information (-version/-v)

- The optional argument (-version/-v) is used to print the version information.

Usage Example

1) Print the version information (-version/-v)

```
fpdfhf -version
fpdfhf -v
```

e) Help information (-help/-h)

- The optional argument (**-help/-h**) is used to print the usage information.

Usage Example

1) Print the usage information (-help/-h)

```
fpdfhf -help
fpdfhf -h
```

3.5 PDFOptimizer

3.5.1 Basic Syntax

fpdfopt *<-i <srcfile/srcfolder>> <-o <destfile/destfolder>> [-t <thread>] [-r [recursion]] [-dc <algorithm> [DPIAbove] [DPISet]] [-cc <algorithm> <level> [blocksize]] [-dm <algorithm> [DPIAbove] [DPISet]] [-cm <algorithm> [level]] [-rd] [-u] [-d <intlist>] [-cl <intlist>] [-op <password>] [-log <logfile>] [-l <level>]*

fpdfopt *-register <code> <licensee>*

fpdfopt *-license*

fpdfopt *-version/-v*

fpdfopt *-help/-h*

Note:

- <> required
- [ ] optional
- A space is needed between the command line argument and the value

Only the < *srcfile/srcfolder* > and <*output*> arguments are actually required. All others are optional which are available for controlling the output PDF files as desired. The arguments could be given in any order. Full details on each are explained in the following section.

3.5.2 Command Line Summary

Option	Parameter	Description
-i	*<-i <string>>* *e.g.* *-i C:\input\1.pdf* *-i c:\input*	Specifies the input file to be optimized. ▪ The input can be a single PDF file or a single folder. ▪ The file name can contain the wildcard characters (*). For example, use *.pdf to include all PDF files in a given folder. **Note** Wildcard characters (*.*) is not supported currently.

Option	Parameter	Description
-o	*<-o <string>>* *e.g.* *-o D:\output\1_opt.pdf* *-o D:\output*	Specifies the path of the output PDF file or folder. ▪ If the input is a PDF file, the output should be a single PDF file, (e.g. *-o D:\output\1_opt.pdf*). ▪ If the input is a folder, the output should be a folder, (e.g. *-o D:\output*). **Note** The specified output path must already exist.
-r	*[-r [integer]]* *e.g.* *-r* *-r 0* *-r 1* *-r 2* …	Specifies the number of layers to recurse when the input is a folder. The default value is 1 (search only the current folder, no sub-folders). ● -r 0 <-r>: searches the full folders. ● -r 1: searches only the current folder. ● -r 2: searches the current folder and its sub-folders … **Note** The PDF file or folder will be skipped if it is secured and the message will be output.
-t	*[-t <integer>]* *e.g.* *-t 1* *-t 2*	Specifies the number of CPU threads to use. The default value is 1.

Option	Parameter	Description
-dc	*[-dc <string> [integer] [integer]]* *-dc <algorithm> [DPIAbove] [DPISet]* *e.g.* *-dc a 225 150* *-dc s 225 150* *-dc b 225 150* *-dc a* *-dc s* *-dc b*	Downsamples color/grayscale images. ▪ **<algorithm> :** Chooses downsampling algorithms **a**: Average Downsampling **s**: SubSampling **b**: Bicubic Downsampling ▪ [**DPIAbove**]: DPI threshold value. Images with a DPI higher than this value will be downsampled. Default value: 225. Allowable range: DPISet-DPISet*10 ▪ [**DPISet**]: Target DPI value images will be downsampled to if their DPI is above the threshold. Default value: 150. Allowable range: 9-2400. **Note** ▪ The **DPIAbove** and **DPISet** must be set at the same time. ▪ If the argument (**-dc**) is not set, images will not be downsampled.
-cc	*[-cc <string> <string> [integer]]* *-cc <algorithm> <level> [blocksize]* *e.g.* *-cc j min* *-cc j low* *-cc j medium* *-cc j high* *-cc j max* *-cc j2 min 256* *-cc j2 low 256* *-cc j2 medium 256* *-cc j2 high 256* *-cc j2 max 256*	Compresses color/grayscale images. ▪ <**algorithm**>: Chooses compression algorithm. **j**: JPEG **j2**: JPEG2000 ▪ <**level**>: Chooses quality level. Allowable levels are min, low, medium, high and max. ▪ [**blocksize**]: Chooses block size for JPEG2000 algorithm only. Default value: 256. Allowable values: 128-2048. **Note** If the argument (**-cc**) is not set, images will not be compressed.

Option	Parameter	Description
-dm	*[-dm <string> [integer] [integer]]* *-dm <algorithm> [DPIAbove] [DPISet]* *e.g.* *-dm a 450 300* *-dm s 450 300* *-dm b 450 300* *-dm a* *-dm s* *-dm b*	Downsamples monochrome images. ▪ **<algorithm> :** Chooses downsampling algorithms **a**: Average Downsampling **s**: SubSampling **b**: Bicubic Downsampling ▪ [**DPIAbove**]: DPI threshold value. Images with a DPI higher than this value will be downsampled. Default value: 450. Allowable range: DPISet-DPISet*10 ▪ [**DPISet**]: Target DPI value images will be downsampled to if their DPI is above the threshold. Default value: 300. Allowable range: 9-2400. **Note** ▪ The **DPIAbove** and **DPISet** must be set at the same time. ▪ If the argument (**-dm**) is not set, images will not be downsampled.
-cm	*[-cm <string> [string]]* *-cm <algorithm> [level]* *e.g.* *-cm jbig2 lossless* *-cm jbig2 lossy* *-cm ccitt* *-cm runlength*	Compresses monochrome images. ▪ <**algorithm**>: Chooses compression algorithm. **jbig2**: JBIG2 **ccitt**: CCITT Group 4 **runlength**: RUN LENGTH ▪ [**level**]: Chooses quality level for JBIG2 algorithm only. Allowable levels are lossless and lossy. **Note** If the argument (**-cm**) is not set, images will not be compressed.
-rd	*[-rd]*	If this flag is set, images will be optimized only if there is a reduction in size.
-u	*[-u]*	Unembeds all fonts in selected PDF document(s).

Option	Parameter	Description
-d	*[-d <string>]* *-d <intlist>* *e.g.* *-d "1"* *-d "1,3,11"* *-d "2-11"*	Discards objects and user data: • 1: discards all form submission, import and reset actions. • 2: flattens form fields. • 3: discards all JavaScript actions. • 4: discards embedded page thumbnails. • 5: discards embedded print settings. • 6: discards bookmarks. • 7: discards all comments, forms and multimedia. • 8: discards external cross references. • 9: discards document information and metadata. • 10: discards file attachments. • 11: discards private data of other applications. **Note** The <intlist> should be entered in the format "1,3,11" or "2-11" without any spaces.
-cl	*[-cl <string>]* *-cl <intlist>* *e.g.* *-cl "1"* *-cl "1,3,4"* *-cl "1-4"*	Clears up the streams, bookmarks or links. • 1: Use Flate to encode streams that are not encoded. • 2: In streams that use LZW encoding, use Flate instead. • 3: Remove invalid bookmarks. • 4: Remove invalid links. **Note** The <intlist> should be entered in the format "1,2,3,4" or "1-4" without any spaces.
-op	*[-op <string>]* *e.g.* *-op 123* *-op welcome*	Specifies the open password for the input file. Not required if the input file is not secured. **Note** The output PDF file will retain the open password from the input documents.
-log	*[-log <string>]* *e.g.* *-log d:\a.log*	Writes log information into a logfile at the specified existing path.

Option	Parameter	Description
-l	*[-l <integer>]* *e.g.* *-l 1* *-l 2* *-l 3* *-l 4*	Sets the log level. The default is 4. • **-l 1**: logs only messages concerning program crash. • **-l 2**: logs failure messages concerning the errors caused during execution or returned from underlying libraries, as well as those for level 1. • **-l 3**: logs warning messages concerning that PDF files are overwritten, as well as those for level 2. • **-l 4**: logs informational messages, as well as those for level 3. **Note** The (**-l**) is valid only when (**-log**) is used.
-register	*[-register <String> <String>]* *-register <code> <licensee>*	Registers the command line tool. ▪ **<code>**: the activation code from Foxit. ▪ **<licensee>**: the Licensee name designated by users.
-help/-h		Prints the usage information.
-version/-v		Prints the version information.
-license		Prints the license information.

3.5.3 Basic Usage

3.5.3.1 Input and Output

a) Input (-i)

- The input file should be a single PDF file or a single folder. It doesn't allow users to input multiple PDF files or folders or a mixture of folders and PDF files. For example:

-i c:\input\1.pdf	(a single PDF file)
-i c:\input	(a single folder)

- It supports relative paths, that is, if the input file is in the current working folder, you can input only the name of the PDF file or folder, instead of an absolute path. For example:

-i test\2.pdf	("test\2.pdf" is in the current working folder)
-i test	("test" folder is in the current working folder)

- It also supports wildcard characters, which are used to process multiple PDF files. For example:

-i "c:\input\*.pdf"	(Only convert PDF files under "c:\input" folder)
-i "test*.pdf"	(Only convert PDF files under "test" folder)

Note *When using wildcard characters in the input files, please better enclose the input files in double quotation marks (" "). In this manual, we all add (" ") when the input files use wildcard characters.*

b) Output (-o)

- If the input file is a single PDF file, you should specify the output path of a PDF file. Or if the input file is a single folder, you should specify the output path of a folder. For example:

-o d:\output\1_opt.pdf	(a single PDF file)
-o d:\output	(a single folder)

Note *The specified output path must already exist.*

- The output also supports relative paths. If the specified output location is in the current working folder, you can input only the name of the PDF file or output folder, instead of an absolute path. For example:

-o output\2_opt.pdf	("output" folder is in the current working folder)
-o output	("output" folder is in the current working folder)

Usage Examples

1) Optimize one single PDF file:

```
fpdfopt -i c:\input\1.pdf -o d:\output\1_opt.pdf
```

2) Optimize PDF files in one folder:

```
fpdfopt -i test -o output  ("test" and "output" folders are in the current working folder)
```

3.5.3.2 Recursion Depth of Sub-folders

- The optional argument (**-r**) indicates the recursion depth of sub-folders, which is valid only when the input is a folder. By default, the recursion depth is 1, that is, the sub-folders are not processed. For more details about this argument, please refer to section 3.5.2 "Command Line Summary".

Usage Examples

1) Search the full sub-folders (-r or -r 0)

```
fpdfopt -i test -o output -r
fpdfopt -i c:\input -o d:\output -r
fpdfopt -i test -o output -r 0
fpdfopt -i c:\input -o d:\output -r 0
```

2) Search only the current folder (-r 1)

```
fpdfopt -i test -o output -r 1
fpdfopt -i c:\input -o d:\output -r 1
```

Note *If you don't use this argument, it searches the current folder by default. For example:*

```
fpdfopt -i test -o output
fpdfopt -i c:\input -o d:\output
```

3) Search the current folder and its sub-folders (-r 2)

```
fpdfopt -i test -o output -r 2
fpdfopt -i c:\input -o d:\output -r 2
```

3.5.3.3 Multi-thread Support

- The optional argument (**-t**) indicates the number of threads, which are used to speed up the batch optimization by making full use of the CPU. By default, the number of the threads is 1.

Note *It is recommended that you should set the number according to the CPU configuration of your computer.*

Usage Example

1) Set the number of threads to 3 (-t 3)

```
fpdfopt -i test -o output -t 3
fpdfopt -i c:\input -o d:\output -t 3
```

3.5.3.4 PDF/Page Settings

a) Color/grayscale images Downsampling (-dc)

- The optional argument (**-dc**) is used to downsample color/grayscale images in the input PDF file. By default, images will not be downsampled. For more details about this argument, please refer to section 3.5.2 "Command Line Summary".

Usage Example

1) Downsample all color/grayscale images with an original resolution higher than 300 dpi to a new resolution of 100 dpi using Average Downsampling algorithm (-dc a 300 100)

```
fpdfopt -i c:\input\1.pdf -o d:\output\1_opt.pdf -dc a 300 100
fpdfopt -i test -o output -dc a 300 100
fpdfopt -i c:\input -o d:\output -dc a 300 100
```

2) Downsample all color/grayscale images with an original resolution higher than 225 dpi to a new resolution of 150 dpi using Bicubic Downsampling algorithm (-dc b, 225 and 150 are the default values)

```
fpdfopt -i c:\input\1.pdf -o d:\output\1_opt.pdf -dc b
fpdfopt -i test -o output -dc b
fpdfopt -i c:\input -o d:\output -dc b
```

b) Color/grayscale images Compression (-cc)

- The optional argument (**-cc**) is used to compress color/grayscale images in the input PDF files with JPEG or JPEG2000 algorithm. For more details about this argument, please refer to section 3.5.2 "Command Line Summary".

Usage Example

1) Compress color/grayscale images with JEPG, medium quality (-cc j medium)

```
fpdfopt -i c:\input\1.pdf -o d:\output\1_opt.pdf -cc j medium
fpdfopt -i test -o output -cc j medium
fpdfopt -i c:\input -o d:\output -cc j medium
```

2) Compress color/grayscale images with JEPG2000, max quality (-cc j2 max)

```
fpdfopt -i c:\input\1.pdf -o d:\output\1_opt.pdf -cc j2 max
fpdfopt -i test -o output -cc j2 max
fpdfopt -i c:\input -o d:\output -cc j2 max
```

c) Monochrome images Downsampling (-dm)

- The optional argument (**-dm**) is used to downsample monochrome images in the input PDF file. By default, images will not be downsampled. For more details about this argument, please refer to section 3.5.2 "Command Line Summary".

Usage Example

1) Downsample all monochrome images with an original resolution higher than 300 dpi to a new resolution of 100 dpi using Average Downsampling algorithm (-dm a 300 100)

```
fpdfopt -i c:\input\1.pdf -o d:\output\1_opt.pdf -dm a 300 100
fpdfopt -i test -o output -dm a 300 100
fpdfopt -i c:\input -o d:\output -dm a 300 100
```

2) Downsample all monochrome images with an original resolution higher than 450 dpi to a new resolution of 300 dpi using Bicubic Downsampling algorithm (-dm b, 450 and 300 are the default values)

```
fpdfopt -i c:\input\1.pdf -o d:\output\1_opt.pdf -dm b
fpdfopt -i test -o output -dm b
fpdfopt -i c:\input -o d:\output -dm b
```

d) Monochrome images Compression (-cm)

- The optional argument (**-cm**) is used to compress monochrome images in the input PDF file with JBIG2, CCITT Group 4, or Run Length algorithm. For more details about this argument, please refer to section 3.5.2 "Command Line Summary".

Usage Example

1) Compress monochrome images with JBIG2, lossless quality (-cm jbig2 lossless)

```
fpdfopt -i c:\input\1.pdf -o d:\output\1_opt.pdf -cm jbig2 lossless
fpdfopt -i test -o output -cm jbig2 lossless
fpdfopt -i c:\input -o d:\output -cm jbig2 lossless
```

2) Compress monochrome images with Run Length (-cm runlength)

```
fpdfopt -i c:\input\1.pdf -o d:\output\1_opt.pdf -cm runlength
fpdfopt -i test -o output -cm runlength
fpdfopt -i c:\input -o d:\output -cm runlength
```

e) Reduction (-rd)

- The optional argument (**-rd**) is used to optimize images only if there is a reduction in size.

Usage Example

1) Optimize images only if there is a reduction in size (-rd)

```
fpdfopt -i c:\input\1.pdf -o d:\output\1_opt.pdf -rd
fpdfopt -i test -o output -rd
fpdfopt -i c:\input -o d:\output -rd
```

f) Unembedded fonts (-u)

- The optional arguments (**-u**) is used to unembed all fonts in selected PDF documents.

Usage Example

1) Unembed all fonts in selected PDF documents (-u)

```
fpdfopt -i c:\input\1.pdf -o d:\output\1_opt.pdf -u
fpdfopt -i test -o output -u
fpdfopt -i c:\input -o d:\output -u
```

g) Objects and user data Discard (-d)

- The optional argument (**-d**) is used to discard objects and user data in PDF files. For more details about this argument, please refer to section 3.5.2 “Command Line Summary”.

Usage Example

1) Discard all form submission, import and reset actions (-d “1”)

```
fpdfopt -i c:\input\1.pdf -o d:\output\1_opt.pdf -d “1”
fpdfopt -i test -o output -d “1”
fpdfopt -i c:\input -o d:\output -d “1”
```

2) Flatten form fields, discard all JavaScript actions and discard bookmarks (-d “2,3,6”)

```
fpdfopt -i c:\input\1.pdf -o d:\output\1_opt.pdf -d “2,3,6”
fpdfopt -i test -o output -d “2,3,6”
fpdfopt -i c:\input -o d:\output -d “2,3,6”
```

h) Clear up (-cl)

- The optional argument (**-cl**) is used to clear up stream, bookmarks, or links. For more details about this argument, please refer to the section 3.5.2 “Command Line Summary”.

Usage Example

1) Use Flate to encode streams that are not encoded, and remove invalid bookmarks (-cl “1,3”)

```
fpdfopt -i c:\input\1.pdf -o d:\output\1_opt.pdf -cl “1,3”
fpdfopt -i test -o output -cl “1,3”
fpdfopt -i c:\input -o d:\output -cl “1,3”
```

i) Open Password (-op)

- The optional argument (**-op**) indicates the open password for the input PDF file which is secured. It is not required if the input file is not secured.

  Note *The output PDF file will retain the open password from the input documents.*

Usage Example

1) Specify the open password for an input PDF file (-op 123)

```
fpdfopt -i c:\input\1.pdf -o d:\output\1_opt.pdf -op 123
fpdfopt -i test\2.pdf -o output\2_opt.pdf -op 123
```

2) Specify the open password for all input PDF files which have the same open password (-op 123)

```
fpdfopt -i c:\input -o d:\output -op 123
fpdfopt -i test -o output -op 123
```

Note *Version 1.0 only supports typing one value for the argument (**-op**). That is, it will only process the PDF files matched the typed open password at a time. If the input PDF files have different open passwords, you should do the work in multiple times.*

3.5.3.5 Other Settings

a) Log file (-log<logfile> -l<log level>)

- The optional argument (**-log**) indicates the path of logfile, where the log message is placed. The (**-l**) indicates the log level. It is valid only when (**-log**) is used. By default, the log level is 4. For more details about this argument, please refer to section 3.5.2 “Command Line Summary”.

Usage Example

1) Save the log file to “d:\output\pdfoptimizer.log” and set the log level to 3 (-log d:\output\PDFOptimizer.log -l 3)

```
fpdfopt -i c:\input -o d:\output -log d:\output\PDFOptimizer.log -l 3
```

b) Register information (-register <code> <licensee>)

- The optional argument (**-register**) is used to register the command line tool. The **<code>** is the activation code from Foxit and **<licensee>** is the licensee name designated by users.

Usage Example

1) Register the pdfoptimizer tool with the code “40G01-01000-50000-2JCTF-3DV20-RZOF4” and the licensee “jessie” (-register 40G01-01000-50000-2JCTF-3DV20-RZOF4 jessie)

```
fpdfopt -register 40G01-01000-50000-2JCTF-3DV20-RZOF4 jessie
```

c) License information (-license)

- The optional argument (**-license**) is used to print the license information.

Usage Example

1) Print the license information (-license)

```
fpdfopt -license
```

d) Version information (-version/-v)

- The optional argument (-version/-v) is used to print the version information.

Usage Example

1) Print the version information (-version/-v)

```
fpdfopt -version
fpdfopt -v
```

e) Help information (-help/-h)

- The optional argument (**-help/-h**) is used to print the usage information.

Usage Example

1) Print the usage information (-help/-h)

```
fpdfopt -help
fpdfopt -h
```

3.6 RMS

3.6.1 Basic Syntax

```
RMSProtector [/decrypt <location>]
              [/encrypt <location> </template <name> [issuer]> [/highstrength] [/revoke]
                       [/MicrosoftIRMV1]]
              [/encrypt <location> </user <name> /rights <rights> [issuer]> [/highstrength] [/revoke]
                       [/MicrosoftIRMV1]]
              [/showtemplates [/sync]] [/preserveattributes]
              [/showencryption <location>]
              [/log <log_file> [/append] [/simple]] [/silent]
RMSProtector /register <code> <licensee>
RMSProtector /license
```

Note:

- <> required
- [ ] optional
- A space is needed between the command line argument and the value

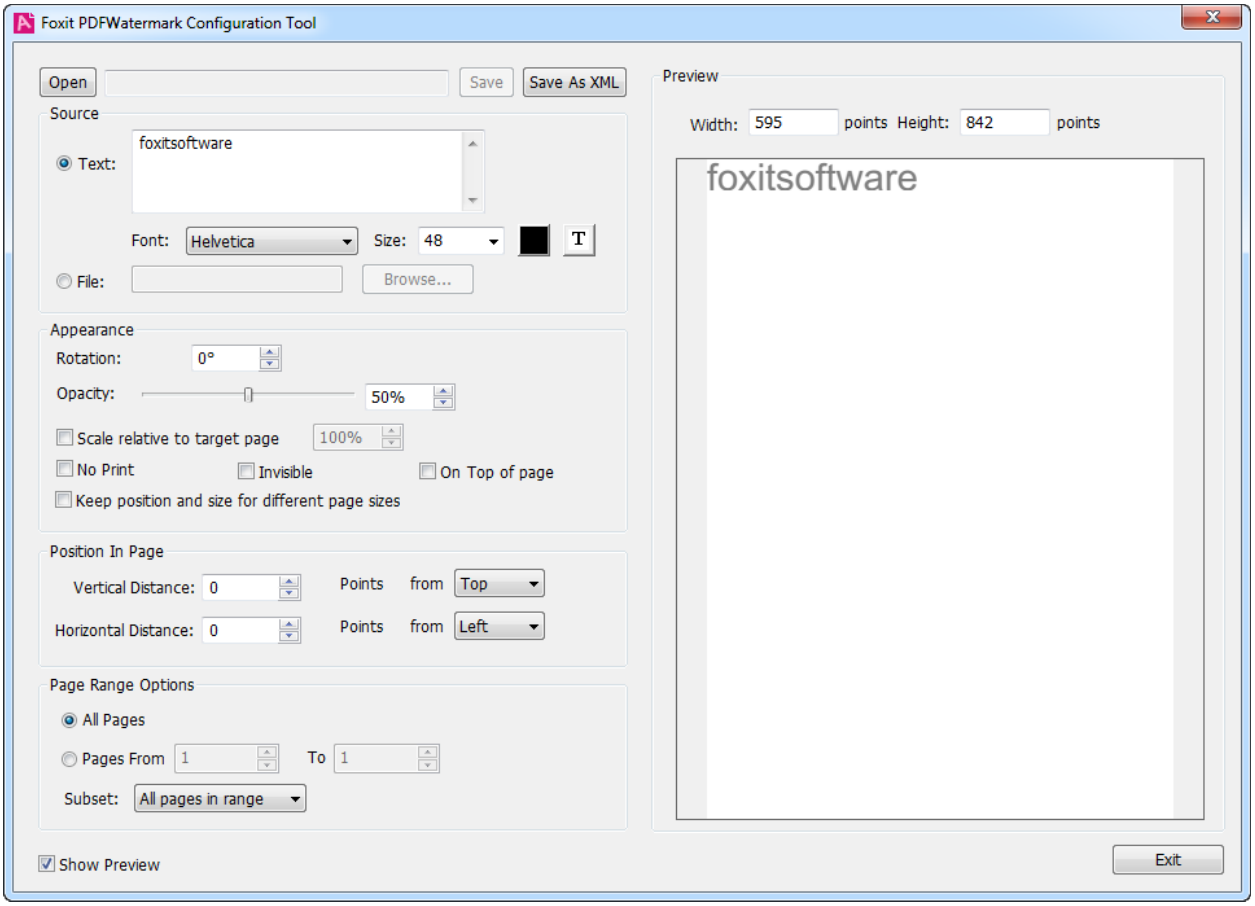
For the RMS tool, please go to the "rms" folder in the installation package for more detailed introduction.

4 Foxit Configuration Tool

4.1 Watermark Configuration Tool

Foxit PDFWatermark Configuration Tool is used to set the properties of watermark and then save them as a configuration file with extension ".xml" which is required in the command line. The watermark settings contain four main parts: source, appearance, positon in page and page range options. We will give a detailed introduction on each part in the following sections.

First, go to "configtool" folder, double-click the fpdfwmconf.exe or fpdfwmconf64.exe to launch Foxit PDFWatermark Configuration Tool, and then you can see the configuration screen as follows.



Source

● Text: foxitsoftware

Font: Helvetica Size: 48

○ File:

Appearance

Rotation: 0°

Opacity: 50%

☐ Scale relative to target page 100%

☐ No Print ☐ Invisible ☐ On Top of page

☐ Keep position and size for different page sizes

Position In Page

Vertical Distance: 0 Points from Top

Horizontal Distance: 0 Points from Left

Page Range Options

● All Pages

○ Pages From 1 To 1

Subset: All pages in range

☑ Show Preview

Preview

Width: 595 points Height: 842 points

PDFWatermark Configuration Screen

Users can click the “**Open**” button to load an existing watermark configuration file (.xml), and then edit its settings according to their desired requirement. After editing, users can click “**Save**” button to save the changes or click “**Save As XML**” button to save it as a new watermark configuration file.

Check “**Show Preview**” to preview how the text/image watermark will be displayed in the PDF file. By default, it is checked.

4.1.1 Watermark Settings

4.1.1.1 Source

Foxit PDFWatermark supports two types of watermark: text and image.

Text Watermark

- **Text**

  Users can enter text in the content box, which will be appeared as text watermark in preview window.

- **Font**

  This option is activated only when Text Watermark is selected. Users can choose the font name, font size and font color from the drop-down menus.

- **Underline**

  Users can click **T** icon to set underline for text when Text Watermark is selected.

Image Watermark

- **File**

  Users can browse their computer and choose an image that will be appeared as image watermark in preview window. The following image formats are currently supported: BMP, DIB, JPG, JPEG, JPE, GIF, PNG, TIFF and TIF.

4.1.1.2 Appearance

This section introduces how to set appearance for text/image watermark.

- **Rotation**

  Users can enter an angle to rotate the text/image watermark. The rotation angle can be set between 0 and 360.

- **Opacity**

  This option is used to set the transparence for text/image watermark. The value is from 0 to 100. 0 means invisible, and 100 means visible solid.

- **Scale relative to target page**

  Check "**Scale relative to target page**" and enter a number in the percentage box to resize the text/image watermark in relation to the PDF page dimensions.

  Note *The drop-down box of font size will be empty if you check "**Scale relative to target page**". And if you reset the font size, the "**Scale relative to target page**" will be unchecked.*

- **No Print**

  This option is used to control the visibility for text/image watermark when you print the PDF. If you check "**No Print**", the added watermark will not be shown when printing. By default, it is not checked.

- **Invisible**

  This option is used to control the visibility for text/image watermark on screen. If you check "**Invisible**", the added watermark will not be shown on screen. By default, it is not checked.

- **On Top of page**

  If you Check "**On Top of page**", the text/image watermark will be placed on top of the page content, that is, the watermark may cover some content. Otherwise, the watermark will be placed under the content, and the page content may obstruct your view of some part of the watermark. By default, it is not checked.

- **Keep position and size for different page size**

  This option is used to control watermark variations in a PDF with pages of varying sizes. If you check "**Keep position and size for different page size**", the added watermark will be the same in different pages.

Note *Please do not check "**Keep position and size for different page size**" and "**Scale relative to target page**" simultaneously, or the "**Keep position and size for different page size**" will be omitted, that is, it will not have any effect.*

4.1.1.3 Position in page

Watermark position in page is controlled by **Vertical Distance** and **Horizontal Distance**. The distance unit is points. Users can set the relative benchmark for the distance, such as Top, Bottom, Left, Right and Center.

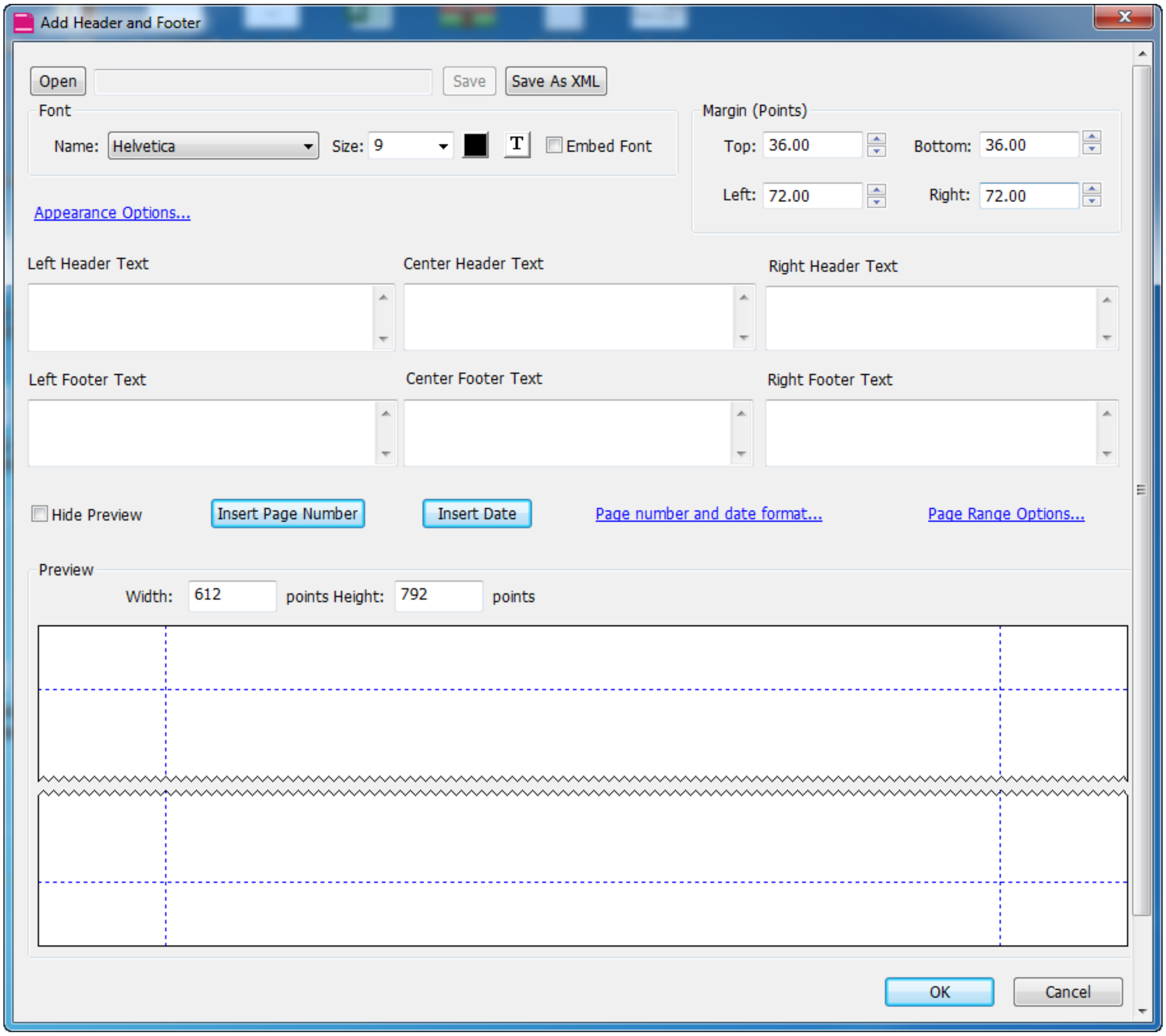
4.1.1.4 Page Range Options

Page range options are used to choose the page range to add watermark. Users can select all pages or specify the page range, or choose even pages or odd pages via clicking the right items in the subset list.

4.2 PDFHeaderFooter Configuration Tool

Foxit PDFHeaderFooter Configuration Tool is used to set the properties of header/footer and then save them as a configuration file with extension ".xml" which is required in the command line. The header/footer settings contain six main parts: font, margin, appearance options, text, page number and date format, and page range options. We will give a detailed introduction on each part on the following sections.

First, go to "configtool" folder, double-click the fpdfhfconf.exe or fpdfhfconf64.exe to launch Foxit PDFHeaderFooter Configuration Tool, and then you can see the configuration screen as follows.



Name: Helvetica Size: 9 ☐ Embed Font

Top: 36.00 Bottom: 36.00

Left: 72.00 Right: 72.00

☐ Hide Preview

Width: 612 points Height: 792 points

PDFHeaderFooter Configuration Screen

Users can click the “**Open**” button to load an existing header/footer configuration file (.xml), and then edit its settings according to their desired requirement. After editing, users can click “**Save**” button to save the changes or click “**Save As XML**” button to save it as a new header/footer configuration file.

Check “**Hide Preview**” to hide the preview window about how the header/footer will be displayed in the PDF file. By default, it is unchecked.

4.2.1 Header/Footer Settings

4.2.1.1 Font

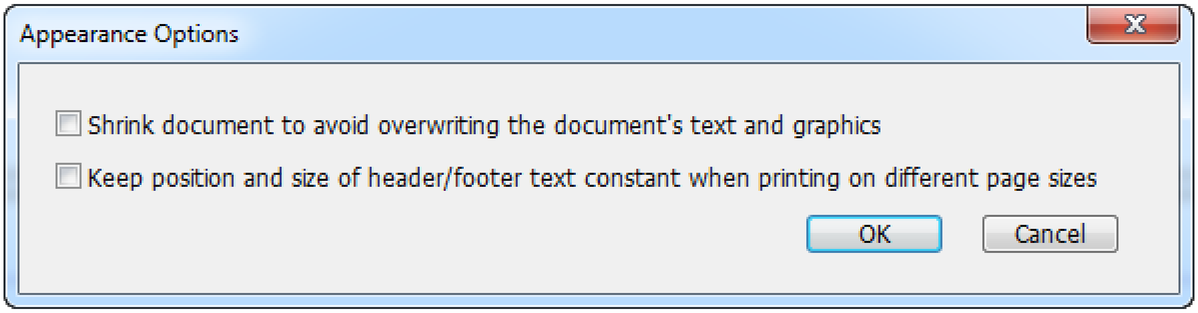
The font option allows users to choose the font name, font size and font color from the drop-down menus, to click the T icon to set underline for text, and to check the “**Embed Font”** to set the text font as embedded font.

4.2.1.2 Margin

The margin option is used to control the header/footer position in PDF page. The unit is “points”. Users can set margins in four directions for the added header/footer, including top, bottom, left and right.

4.2.1.3 Appearance Options

There are two options you can choose when you click on “Appearance Options” as shown in the following figure. One is “**Shrink document to avoid overwriting the document’s text and graphics**”, which can be used to shrink the document when the added header/footer may overwrite the text or graphic in the document. The other is “**Keep position and size of header/footer text constant when printing on different page sizes**”, which can keep the added header/footer at the same position in different pages.



☐ Shrink document to avoid overwriting the document's text and graphics
☐ Keep position and size of header/footer text constant when printing on different page sizes

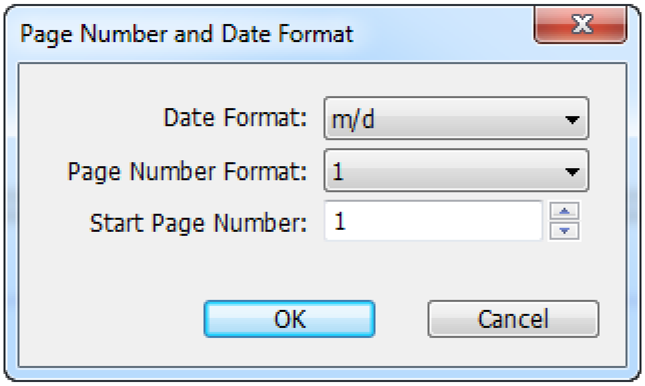
Appearance Options

4.2.1.4 Text

Six text-input boxes are provided to input the texts of header/footer you want to add. It includes left header text, center header text, right header text, left footer text, center footer text and right footer text. Users can input text to the desired box to add header/footer.

4.2.1.5 Page Number and Date Format

Click “**Page Number and Date Format**” to open the setting window as shown in the following figure. Users can set the Date Format, Page Number Format and Start Page Number. After setting, click the “**Insert Page Number**” button to insert page number to the desired text-input box, and click the “**Insert Date**” button to insert date to the desired text-input box.

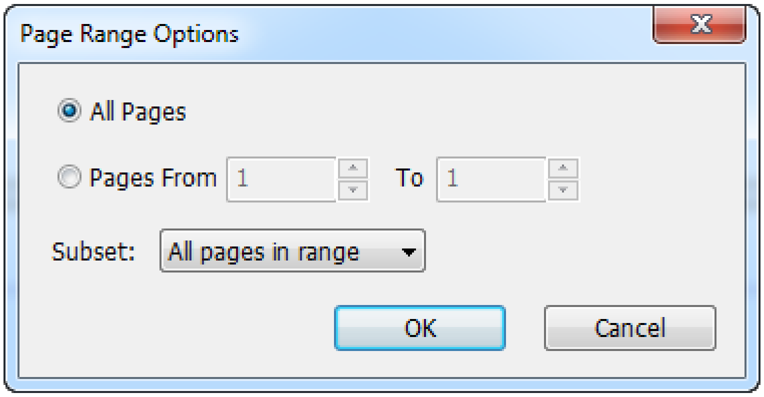


Date Format: m/d
Page Number Format: 1
Start Page Number: 1

Page Number and Date Format

4.2.1.6 Page Range Options

Click the “**Page Range Options**” to open the setting window as shown in the following figure. Page range options are used to choose the page range to add header/footer. Users can select all pages or specify the page range, or choose even pages or odd pages via clicking the right items in the subset list.



● All Pages
○ Pages From 1 To 1
Subset: All pages in range

Page Range Options

5 Working with API

Foxit PDF Toolkit provides another way for users who want to perform PDF manipulation through calling API. Each tool offers a simple-to-use API and users should call four functions to run it.

First, initialize Foxit PDF Toolkit library and check the license.

```
int FXT_InitLibrary(const wchar* key, int screenFlag);
```

The parameter "**key**" is the path of the license file ("**ftlkey.txt**", generated in the installed path after registering the tool using the activation code purchased from Foxit.).The parameter "**screenFlag**" controls whether to print the output information to screen. If the value is 1, print the output information to screen, not vice versa.

Second, call the function of the desired tool.

```
int FXT_Image2PDFRun(const char* commandline, FXT_CallbackFun callback, void* userData = 0);
int FXT_Office2PDFRun(const char* commandline, FXT_CallbackFun callback, void* userData = 0);
int FXT_WatermarkRun(const char* commandline, FXT_CallbackFun callback, void* userData = 0);
int FXT_HeaderFooterRun(const char* commandline, FXT_CallbackFun callback, void* userData = 0);
int FXT_OptimizerRun(const char* commandline, FXT_CallbackFun callback, void* userData = 0);
```

Select the corresponding function of the tool. Where, the parameter "**commandline**" is a command string which is exactly the same as the general syntax used for the command line application (e.g. "-i c:\input -o d:\output"). The parameter "**callback**" is the callback function provided for users to do some special processing. The parameter "**userData**" is a pointer used to transfer user data.

Third, declare the callback function.

```
typedef char*(*FXT_CallbackFun)(void* userData, int mode, char* msg, bool* isStop)
```

The parameter "**useData**" is a pointer of user data. The parameter "**mode**" specifies the output mode of information including CALLBACK_PDFTOOL_RUN_ERROR, CALLBACK_PDFTOOL_PARAM_ERROR, CALLBACK_PDFTOOL_MSG and CALLBACK_PDFTOOL_PASSWORD. The parameter "**msg**" is the output message when calling the callback function. The parameter "**isStop**" controls whether to stop the current application. If the value is true, stop the current application, otherwise, continue running the application.

Last, release and destroy Foxit PDF Toolkit library.

```
void FXT_DestroyLibrary();
```

For more details about the API, please refer to the header file “fxpdftools.h” in the “include” folder in the installation path.

The following are some examples on how to work with API for each tool.

5.1 Image2PDF

5.1.1 Working with Image2PDF API

The following is the simplest application that can be built using Image2PDF API:

```
// Using C/C++
void main(int argc, char* argv[])
{
	int ret = FXT_InitLibrary(L"license_key", 0);
	FXT_Image2PDFRun("-i d:\\input -o d:\\output\\output.pdf", myCallBack, NULL);

	FXT_DestroyLibrary();
}
```

This application converts all the image files in the “input” folder into one PDF file (output.pdf). To convert each image file into individual PDF files, just delete “output.pdf” from the command string.

The command string (`"-i d:\\input -o d:\\output\\output.pdf"`) of the above application is set by users directly in advance. Users also can get the command string through command line. Building a command line application using Image2PDF API is as simple as the following:

```
// Using C/C++
void main(int argc, char* argv[])
{
	int ret = FXT_InitLibrary(L"license_key", 0);

	std::string strCommandline = "";
	for(int i = 1; i < argc; i++)
		strCommandline.append(argv[i]).append(" ");
	FXT_Image2PDFRun(strCommandline.c_str(), myCallBack, NULL);

	FXT_DestroyLibrary();
}
```

It is also possible to build the command string dynamically (e.g. based on the user or a dynamic input) as illustrated in the following code snippet:

```
void main(int argc, char* argv[])
{
	int ret = FXT_InitLibrary(L"license_key", 0);

	std::string input_folder = "d:\\samples\\input";//or image file: input.jpg...
	std::string output_folder = "d:\\samples\\output";//or PDF file: output.pdf...
	std::string set_password = "secret";
	std::string log_file = "d:\\samples\\output\\image2pdf.log";
	//set thread number
	int thread_number = 4;

	// log level
	int log_level = 4;

	// the resolution (DPI:= Dots Pet Inch) for the output PDF file. Valid only when not
	// set width and height.
	int dpi = 96;

	// recursion depth of search sub-folders. 0: search the full folders.
	int depth = 0;

	// page size of the output PDF file.
	int width = 500, height = 500;

	// page margin of the output PDF file. No margin by default.
	int margin_top = 20, margin_bottom = 20, margin_left = 20, margin_right = 20;

	// create bookmark for the output PDF file using image name.
	bool bookmark = true;


	// --------------------------------------------------
	// Given the above settings build a command string.
	std::string strCommandline = "";

	if(!input_folder.empty())
		strCommandline.append("-i ").append(input_folder).append(" ");

	if(!output_folder.empty())
		strCommandline.append("-o " ).append(output_folder).append(" ");

	if(!set_password.empty())
		strCommandline.append("-sp ").append(set_password).append(" ");

	if(!log_file.empty())
		strCommandline.append("-log ").append(log_file).append(" ");

	const int MAX_LEGHT = 128;
	const int DATA_RADIX = 10;
	char temp[MAX_LEGHT] = {0};

	if (log_level > 0)
	{
		itoa(log_level,temp,DATA_RADIX);
		strCommandline.append("-l ").append(temp).append(" ");
	}

	if (width > 0 && height > 0)
	{
		memset(temp, 0 , MAX_LEGHT);
		itoa(width, temp ,DATA_RADIX);
		strCommandline.append("-width ").append(temp).append(" ");

		memset(temp, 0 , MAX_LEGHT);
		itoa(height, temp ,DATA_RADIX);
		strCommandline.append("-height ").append(temp).append(" ");
```

```
        }
        else
        {       //use dpi to set page size.
                if (dpi >=0 )
                {
                        memset(temp, 0 ,MAX_LEGHT);
                        itoa(dpi, temp ,DATA_RADIX);
                        strCommandline.append("-dpi ").append(temp).append(" ");
                }
        }

        if (depth >= 0)
        {
                memset(temp, 0 , MAX_LEGHT);
                itoa(depth, temp ,DATA_RADIX);
                strCommandline.append("-depth ").append(temp).append(" ");
        }

        if (thread_number>= 0)
        {
                memset(temp, 0, MAX_LEGHT);
                itoa(thread_number, temp ,DATA_RADIX);
                strCommandline.append("-t ").append(temp).append(" ");
        }

        if (margin_top >= 0 || margin_right >= 0 || margin_bottom >= 0 || margin_left >= 0)
        {
                bool flag = false;
                if (margin_left >= 0)
                {
                        memset(temp, 0 , MAX_LEGHT);
                        itoa(margin_left, temp ,DATA_RADIX);
                        strCommandline.append("-margin ").append(temp).append(" ");
                        if(margin_top >= 0)
                        {
                                memset(temp, 0 , MAX_LEGHT);
                                itoa(margin_top, temp ,DATA_RADIX);
                                strCommandline.append(temp).append(" ");
                                if(margin_right >= 0)
                                {
                                        memset(temp, 0 , MAX_LEGHT);
                                        itoa(margin_right, temp ,DATA_RADIX);
                                        strCommandline.append(temp).append(" ");
                                        if(margin_bottom >= 0)
                                        {
                                                memset(temp, 0 , MAX_LEGHT);
                                                itoa(margin_bottom, temp ,DATA_RADIX);
                                                strCommandline.append(temp).append(" ");
                                        }
                                }
                        }
                }
        }

        if(bookmark)  strCommandline.append("-b").append(" ");

        FXT_Image2PDFRun(strCommandline.c_str(), myCallBack, NULL);

        FXT_DestroyLibrary() ;
}
```

5.1.2 Reporting Progress Messages and Errors

To find out if Image2PDF processing was successful, the application can query the status code returned by FXT_Image2PDFRun().

For example,

```
int ret = FXT_Image2PDFRun(...);
if (ret == FXT_ERROR_SUCCESS) {
       // No errors...
}
else if (ret == FXT_ERROR_LICENSE) {
       // Invalid license...
}
else if (ret == FXT_ERROR_PARAM) {
       // Invalid param
}
else if (ret == FXT_ERROR_LIBRARY) {
       // Failed to initialize GSDK Library
}
else if (ret == FXT_ERROR_IMAGEFORMAT) {
       // Unsupported image format
}
else {
       // Other error
}
```

A non-zero value indicates that an error was encountered. You can find the list for all error code in the "include/fxpdftools.h" header.

For more detailed error and message reporting, you can pass a pointer to the custom callback function in the second parameter of FXT_Image2PDFRun(). The last parameter in FXT_Image2PDFRun() is a pointer to custom data that you may want to pass to the callback function.

The following is a sample callback function:

```
// Using C/C++
char* MyCallback(void* userData, int mode, char* msg, bool* isStop) {
       if (mode == CALLBACK_PDFTOOL_RUN_ERROR) {
              cout << "Error: " << msg << endl;
       }
       else if (mode == CALLBACK_PDFTOOL_PARAM_ERROR) {
              cout << msg;
       }
       else if (mode == CALLBACK_PDFTOOL_MSG) {
              cout << msg;
       }
       else if (mode == CALLBACK_PDFTOOL_PASSWORD) {
              static string password;
              cin >> password;
              return (char*)password.c_str();
       }
```

```
        return 0;
}
```

5.2 Office2PDF

5.2.1 Working with Office2PDF API

The following is the simplest application that can be built using Office2PDF API:

```
// Using C/C++
void main(int argc, char* argv[])
{
        int ret = FXT_InitLibrary(L"license_key", 0);
        FXT_ Office2PDFRun("-i d:\\input -o d:\\output\\output", myCallBack, NULL);

        FXT_DestroyLibrary();
}
```

This application converts all the Microsoft Office files in the “input” folder into PDF files except for the secured files.

The command string (`"-i d:\\input -o d:\\output"`) of the above application is set by users directly in advance. Users also can get the command string through command line. Building a command line application using Office2PDF API is as simple as the following:

```
// Using C/C++
void main(int argc, char* argv[])
{
        int ret = FXT_InitLibrary(L"license_key", 0);

        std::string strCommandline = "";
        for(int i = 1; i < argc; i++)
                strCommandline.append(argv[i]).append(" ");
        FXT_ Office2PDFRun(strCommandline.c_str(), myCallBack, NULL);

        FXT_DestroyLibrary();
}
```

It is also possible to build the command string dynamically (e.g. based on the user or a dynamic input) as illustrated in the following code snippet:

```
void main(int argc, char* argv[])
{
        int ret = FXT_InitLibrary(L"license_key", 1);

        std::string input_folder = "d:\\samples\\input";//or office file: input.doc...
        std::string output_folder = "d:\\samples\\output";//or   PDF file: output.pdf...
        std::string set_password = "secret";
        std::string log_file = "d:\\samples\\output\\office2pdf.log";

        //set thread number
        int thread_number = 4;

        // log level
        int log_level = 4;


        // create bookmark for the output PDF file using headings of a Microsoft Word file.
        int bookmark = 1;

        // specifies that the output PDF file(s) should be compliant with the PDF/A standard.
        bool pdfa = true;

        // specifies the conversion mode for Microsoft Excel file.
        int scale = 3;

        // --------------------------------------------------
        // Given the above settings build a command string.
        std::string strCommandline = "";

        if(!input_folder.empty())
                strCommandline.append("-i ").append(input_folder).append(" ");

        if(!output_folder.empty())
                strCommandline.append("-o " ).append(output_folder).append(" ");

        if(!set_password.empty())
                strCommandline.append("-op ").append(set_password).append(" ");

        if(!log_file.empty())
                strCommandline.append("-log ").append(log_file).append(" ");

        const int MAX_LEGHT = 128;
        const int DATA_RADIX = 10;
        char temp[MAX_LEGHT] = {0};

        if(log_level > 0)
        {
                itoa(log_level,temp,DATA_RADIX);
                strCommandline.append("-l ").append(temp).append(" ");
        }

        if(thread_number>= 0)
        {
                memset(temp, 0 , MAX_LEGHT);
                itoa(thread_number, temp ,DATA_RADIX);
                strCommandline.append("-t ").append(temp).append(" ");
        }

        if(bookmark >= 0)
        {
                memset(temp, 0 , MAX_LEGHT);
                itoa(bookmark, temp ,DATA_RADIX);
                strCommandline.append("-b ").append(temp).append(" ");
        }

        if(pdfa)      strCommandline.append("-pdfa").append(" ");

        if(scale >= 0)
        {
                memset(temp, 0 , MAX_LEGHT);
```

```
            itoa(scale, temp ,DATA_RADIX);
            strCommandline.append("-scale ").append(temp).append(" ");

        }

        FXT_Office2PDFRun(strCommandline.c_str(), myCallBack, NULL);

        FXT_DestroyLibrary();
}
```

5.2.2 Reporting Progress Messages and Errors

To find out if Office2PDF processing was successful, the application can query the status code returned by FXT_Office2PDFRun().

For example,

```
int ret = FXT_Office2PDFRun(...);
if (ret == FXT_ERROR_SUCCESS) {
        // No errors...
}
else if (ret == FXT_ERROR_LICENSE) {
        // Invalid license...
}
else if (ret == FXT_ERROR_PARAM) {
        // Invalid param
}
else if (ret == FXT_ERROR_LIBRARY) {
        // Failed to initialize GSDK Library
}
else if (ret == FXT_ERROR_FILE) {
        // File error: cannot be opened...
}
else {
        // Other error
}
```

A non-zero value indicates that an error was encountered. You can find the list for all error code in the “include/fxpdftools.h” header.

For more detailed error and message reporting, you can pass a pointer to the custom callback function in the second parameter of FXT_Office2PDFRun(). The last parameter in FXT_Office2PDFRun() is a pointer to custom data that you may want to pass to the callback function.

The following is a sample callback function:

```
// Using C/C++
char* MyCallback(void* userData, int mode, char* msg, bool* isStop) {
        if (mode == CALLBACK_PDFTOOL_RUN_ERROR) {
                cout << "Error: " << msg << endl;
        }
        else if (mode == CALLBACK_PDFTOOL_PARAM_ERROR) {
                cout << msg;
        }
        else if (mode == CALLBACK_PDFTOOL_MSG) {
                cout << msg;
        }
        else if (mode == CALLBACK_PDFTOOL_PASSWORD) {
                static string password;
                cin >> password;
                return (char*)password.c_str();
        }
        return 0;
}
```

5.3 PDFWatermark

5.3.1 Working with PDFWatermark API

The following is the simplest application that can be built using PDFWatermark API:

```
// Using C/C++
void main(int argc, char* argv[])
{
        int ret = FXT_InitLibrary(L"license_key", 0);
        int FXT_WatermarkRun("-i d:\\input -o d:\\output -conf d:\\conf_wm.xml", myCallBack,
NULL);

        FXT_DestroyLibrary();
}
```

This application adds watermark into PDF files. All the PDF files in the "input" folder will be added a watermark and output to "d:\output" folder except for the secured files.

The command string (`"-i d:\\input -o d:\\output -conf d:\\conf_wm.xml "`) of the above application is set by users directly in advance. Users also can get the command string through command line. Building a command line application using PDFWatermark API is as simple as the following:

```
// Using C/C++
void main(int argc, char* argv[])
{
        int ret = FXT_InitLibrary(L"license_key", 0);

        std::string strCommandline = "";
        for(int i = 1; i < argc; i++)
                strCommandline.append(argv[i]).append(" ");
        FXT_WatermarkRun(strCommandline.c_str(), myCallBack, NULL);

        FXT_DestroyLibrary();
}
```

It is also possible to build the command string dynamically (e.g. based on the user or a dynamic input) as illustrated in the following code snippet:

```
void main(int argc, char* argv[])
{
        int ret = FXT_InitLibrary(L"license_key", 0);

        std::string input_folder = "d:\\samples\\input";//or PDF file: input.pdf...
        std::string output_folder = "d:\\samples\\output";//or PDF file: output.pdf...
        std::string set_password = "secret";
        std::string log_file = "d:\\samples\\output\\watermark.log";
        std::string conf_file = "d:\\samples\\conf\\conf_wm.xml";//the configuration file

        //set thread number
        int thread_number = 4;

        // log level
        int log_level = 4;

        // --------------------------------------------------
        // Given the above settings build a command string.
        std::string strCommandline = "";

        if(!input_folder.empty())
                strCommandline.append("-i ").append(input_folder).append(" ");

        if(!output_folder.empty())
                strCommandline.append("-o " ).append(output_folder).append(" ");

        if(!conf_file.empty())
                strCommandline.append("-conf " ).append(conf_file).append(" ");

        if(!set_password.empty())
                strCommandline.append("-op ").append(set_password).append(" ");

        if(!log_file.empty())
                strCommandline.append("-log ").append(log_file).append(" ");

        if (log_level > 0)
        {
                itoa(log_level,temp,DATA_RADIX);
                strCommandline.append("-l ").append(temp).append(" ");
        }

        if (thread_number >= 0)
        {
                memset(temp, 0 , MAX_LEGHT);
                itoa(thread_number, temp ,DATA_RADIX);
                strCommandline.append("-t ").append(temp).append(" ");
        }

        FXT_WatermarkRun (strCommandline.c_str(), myCallBack, NULL);

        FXT_DestroyLibrary ();
```

5.3.2 Reporting Progress Messages and Errors

To find out if PDFWatermark processing was successful, the application can query the status code returned by FXT_WatermarkRun().

For example,

```
int ret = FXT_ WatermarkRun(...);
if (ret == FXT_ERROR_SUCCESS) {
	// No errors...
}
else if (ret == FXT_ERROR_LICENSE) {
	// Invalid license...
}
else if (ret == FXT_ERROR_PARAM) {
	// Invalid param
}
else if (ret == FXT_ERROR_LIBRARY) {
	// Failed to initialize GSDK Library
}
else if (ret == FXT_ERROR_FORMAT) {
	// Format error: not a PDF or a corrupted PDF
}
else {
	// Other error
}
```

A non-zero value indicates that an error was encountered. You can find the list for all error code in the "include/fxpdftools.h" header.

For more detailed error and message reporting, you can pass a pointer to the custom callback function in the second parameter of FXT_WatermarkRun(). The last parameter in FXT_WatermarkRun() is a pointer to custom data that you may want to pass to the callback function.

The following is a sample callback function:

```
// Using C/C++
char* MyCallback(void* userData, int mode, char* msg, bool* isStop) {
	if (mode == CALLBACK_PDFTOOL_RUN_ERROR) {
		cout << "Error: " << msg << endl;
	}
	else if (mode == CALLBACK_PDFTOOL_PARAM_ERROR) {
		cout << msg;
	}
	else if (mode == CALLBACK_PDFTOOL_MSG) {
		cout << msg;
	}
	else if (mode == CALLBACK_PDFTOOL_PASSWORD) {
		static string password;
		cin >> password;
		return (char*)password.c_str();
	}
	return 0;
}
```

5.4 PDFHeaderFooter

5.4.1 Working with PDFHeaderFooter API

The following is the simplest application that can be built using PDFHeaderFooter API:

```
// Using C/C++
void main(int argc, char* argv[])
{
        int ret = FXT_InitLibrary(L"license_key", 0);
        FXT_HeaderFooterRun("-i d:\\input -o d:\\output\\output.pdf -mode 1 -overlay -conf
d:\\conf_hf.xml", myCallBack, NULL);

        FXT_DestroyLibrary();
}
```

This application adds header/footer into PDF files. All the PDF files in the "input" folder will be added a header/footer and output to "d:\output" folder except for the secured files.

The command string (`"-i d:\\input -o d:\\output\\output.pdf -mode 1 -overlay -conf d:\\conf_hf.xml"`) of the above application is set by users directly in advance. Users also can get the command string through command line. Building a command line application using PDFHeaderFooter API is as simple as the following:

```
// Using C/C++
void main(int argc, char* argv[])
{
        int ret = FXT_InitLibrary(L"license_key", 0);

        std::string strCommandline = "";
        for(int i = 1; i < argc; i++)
                strCommandline.append(argv[i]).append(" ");
        FXT_HeaderFooterRun(strCommandline.c_str(), myCallBack, NULL);

        FXT_DestroyLibrary();
}
```

It is also possible to build the command string dynamically (e.g. based on the user or a dynamic input) as illustrated in the following code snippet:

```
void main(int argc, char* argv[])
{
        int ret = FXT_InitLibrary(L"license_key", 1);

        std::string input_folder = "d:\\samples\\input";//or PDF file: input.pdf...
        std::string output_folder = "d:\\samples\\output";//or   PDF file: output.pdf...
        std::string open_password = "secret";
        std::string log_file = "d:\\samples\\output\\headerfooter.log";
        std::string conf_file= "d:\\samples\\conf\\conf_hf.xml";// the configuration tool
        //set thread number
        int thread_number = 4;

        // log level
        int log_level = 4;

        // Specifies the mode to be used. Adds a new header/footer.
        int mode = 1;

        // --------------------------------------------------
        // Given the above settings build a command string.
        std::string strCommandline = "";

        if(!input_folder.empty())
                strCommandline.append("-i ").append(input_folder).append(" ");

        if(!output_folder.empty())
                strCommandline.append("-o ").append(output_folder).append(" ");

        if(!open_password.empty())
                strCommandline.append("-op ").append(open_password).append(" ");

        if(!log_file.empty())
                strCommandline.append("-log ").append(log_file).append(" ");

        const int MAX_LEGHT = 128;
        const int DATA_RADIX = 10;
        char temp[MAX_LEGHT] = {0};

        if (log_level > 0)
        {
                itoa(log_level,temp,DATA_RADIX);
                strCommandline.append("-l ").append(temp).append(" ");
        }

        if (thread_number>= 0)
        {
                memset(temp, 0 , MAX_LEGHT);
                itoa(thread_number, temp ,DATA_RADIX);
                strCommandline.append("-t ").append(temp).append(" ");
        }

        if (mode >= 0)
        {
                memset(temp, 0 , MAX_LEGHT);
                itoa(mode, temp ,DATA_RADIX);
                strCommandline.append("-mode ").append(temp).append(" ");
        }

        if(!config.empty())
                strCommandline.append("-conf ").append(conf_file).append(" ");

        FXT_HeaderFooterRun(strCommandline.c_str(), myCallBack, NULL);

        FXT_DestroyLibrary();
}
```

5.4.2 Reporting Progress Messages and Errors

To find out if PDFHeaderFooter processing was successful, the application can query the status code returned by FXT_HeaderFooterRun().

For example,

```
int ret = FXT_HeaderFooterRun(...);
if (ret == FXT_ERROR_SUCCESS) {
       // No errors...
}
else if (ret == FXT_ERROR_LICENSE) {
       // Invalid license...
}
else if (ret == FXT_ERROR_PARAM) {
       // Invalid param
}
else if (ret == FXT_ERROR_LIBRARY) {
       // Failed to initialize GSDK Library
}
else if (ret == FXT_ERROR_FORMAT) {
       // Format error: not a PDF or a corrupted PDF
}
else {
       // Other error
}
```

A non-zero value indicates that an error was encountered. You can find the list for all error code in the “include/fxpdftools.h” header.

For more detailed error and message reporting, you can pass a pointer to the custom callback function in the second parameter of FXT_Header0FooterRun(). The last parameter in FXT_HeaderFooterRun() is a pointer to custom data that you may want to pass to the callback function.

The following is a sample callback function:

```
// Using C/C++
char* MyCallback(void* userData, int mode, char* msg, bool* isStop) {
       if (mode == CALLBACK_PDFTOOL_RUN_ERROR) {
              cout << "Error: " << msg << endl;
       }
       else if (mode == CALLBACK_PDFTOOL_PARAM_ERROR) {
              cout << msg;
       }
       else if (mode == CALLBACK_PDFTOOL_MSG) {
              cout << msg;
       }
       else if (mode == CALLBACK_PDFTOOL_PASSWORD) {
              static string password;
              cin >> password;
              return (char*)password.c_str();
       }
```

```
        return 0;
}
```

5.5 PDFOptimizer

5.5.1 Working with PDFOptimizer API

The following is the simplest application that can be built using PDFOptimizer API:

```
// Using C/C++
void main(int argc, char* argv[])
{
        int ret = FXT_InitLibrary(L"license_key", 0);
        FXT_OptimizerRun("-i d:\\input -o d:\\output", myCallBack, NULL);

        FXT_DestroyLibrary();
}
```

This application optimizes PDF files. All the PDF files in the "input" folder will be optimized and output to "d:\output" folder except for the secured files.

The command string (`"-i d:\\input -o d:\\output\\output.pdf"`) of the above application is set by users directly in advance. Users also can get the command string through command line. Building a command line application using Image2PDF API is as simple as the following:

```
// Using C/C++
void main(int argc, char* argv[])
{
        int ret = FXT_InitLibrary(L"license_key", 0);

        std::string strCommandline = "";
        for(int i = 1; i < argc; i++)
                strCommandline.append(argv[i]).append(" ");
        FXT_OptimizerRun(strCommandline.c_str(), myCallBack, NULL);

        FXT_DestroyLibrary();
}
```

It is also possible to build the command string dynamically (e.g. based on the user or a dynamic input) as illustrated in the following code snippet:

```
int main(int argc, char* argv[])
{
        int ret =  FXT_InitLibrary(L"license_key", 0);

        std::string input_folder = "d:\\samples\\input";   //or PDF file: input.pdf
        std::string output_folder = "d:\\samples\\output"; //or PDF file: output.pdf
        std::string set_password = "secret";
        std::string log_file = "d:\\samples\\Output\\pdfoptimizer.log";

        //set thread number
        int thread_number = 4;
```

```
// log level
int log_level = 4;

// recursion depth of search sub-folders. 0: search the full folders.
int depth = 0;

// --------------------------------------------------
// Given the above settings build a command string.
std::string strCommandline = "";

if(!input_folder.empty())
       strCommandline.append("-i ").append(input_folder).append(" ");

if(!output_folder.empty())
       strCommandline.append("-o " ).append(output_folder).append(" ");

if(!set_password.empty())
       strCommandline.append("-op ").append(set_password).append(" ");

if(!log_file.empty())
       strCommandline.append("-log ").append(log_file).append(" ");

const int MAX_LEGHT = 128;
const int DATA_RADIX  = 10;
char temp[MAX_LEGHT] = {0};

if (thread_number >= 0)       //set thread number
{
       memset(temp, 0 , MAX_LEGHT);
       itoa(thread_number, temp ,DATA_RADIX);
       strCommandline.append("-t ").append(temp).append(" ");
}

if (log_level > 0)
{
       itoa(log_level,temp,DATA_RADIX);
       strCommandline.append("-l ").append(temp).append(" ");
}

if (depth >= 0)
{
       memset(temp, 0 , MAX_LEGHT);
       itoa(depth, temp ,DATA_RADIX);
       strCommandline.append("-r ").append(temp).append(" ");
}

//set color/gray images DownSample
std::string dcAlgorithm = "b";  //or "a","s"
std::string dcDpiAbove  = "225";
std::string dcDpiSet    = "150";
strCommandline.append("-dc ").append(dcAlgorithm).append(" ");
strCommandline.append(dcDpiAbove).append(" ");
strCommandline.append(dcDpiSet).append(" ");

//Set color/gray images compress
std::string ccAlgorithm = "j";  //or "j2"
std::string ccLevel = "medium";  //or "min","low","high","max"
strCommandline.append("-cc ").append(ccAlgorithm).append(" ");
strCommandline.append(ccLevel).append(" ");
```

```
        //set monochrome images DownSample
        std::string dmAlgorithm = "b";    //or "a","s"
        std::string dmDpiAbove  = "450";
        std::string dmDpiSet    = "300";
        strCommandline.append("-dm ").append(dmAlgorithm).append(" ");
        strCommandline.append(dmDpiAbove).append(" ");
        strCommandline.append(dmDpiSet).append(" ");

        //set monochrome images compress
        std::string cmAlgorithm = "jbig2";  //or "ccitt","runlength"
        std::string cmLevel = "lossless";   //or "lossy"
        strCommandline.append("-cm ").append(cmAlgorithm).append(" ");
        strCommandline.append(cmLevel).append(" ");

        //Discard objects or User Data.
        std::string discardList = "\"1-11\"";
        strCommandline.append("-d ").append(discardList).append(" ");

        //Cleans up streams, bookmarks, or links.
        std::string cleanupList = "\"1-4\"";
        strCommandline.append("-cl ").append(cleanupList).append(" ");

        strCommandline.append("-rd ");

        //Unembeds all fonts in selected PDF document(s).
        strCommandline.append("-u ");

        FXT_OptimizerRun(strCommandline.c_str(), myCallBack, NULL);
        FXT_DestroyLibrary();
}
```

5.5.2 Reporting Progress Messages and Errors

To find out if PDFOptimizer processing was successful, the application can query the status code returned by FXT_OptimizerRun().

For example,

```
int ret = FXT_OptimizerRun(...);
if (ret == FXT_ERROR_SUCCESS) {
        // No errors...
}
else if (ret == FXT_ERROR_LICENSE) {
        // Invalid license...
}
else if (ret == FXT_ERROR_PARAM) {
        // Invalid parameter
}
else if (ret == FXT_ERROR_LIBRARY) {
        // Failed to initialize GSDK Library
}
else if (ret == FXT_ERROR_FORMAT) {
        // Format error: not a PDF or a corrupted PDF
}
else {
        // Other error
}
```

A non-zero value indicates that an error was encountered. You can find the list for all error code in the "include/fxpdftools.h" header.

For more detailed error and message reporting, you can pass a pointer to the custom callback function in the second parameter of FXT_OptimizerRun(). The last parameter in FXT_OptimizerRun() is a pointer to custom data that you may want to pass to the callback function.

The following is a sample callback function:

```
// Using C/C++
char* MyCallback(void* userData, int mode, char* msg, bool* isStop) {
	if (mode == CALLBACK_PDFTOOL_RUN_ERROR) {
		cout << "Error: " << msg << endl;
	}
	else if (mode == CALLBACK_PDFTOOL_PARAM_ERROR) {
		cout << msg;
	}
	else if (mode == CALLBACK_PDFTOOL_MSG) {
		cout << msg;
	}
	else if (mode == CALLBACK_PDFTOOL_PASSWORD) {
		static string password;
		cin >> password;
		return (char*)password.c_str();
	}
	return 0;
}
```

6 Support

6.1 Reporting Problem

Should you encounter any technical questions or bug issues when using Foxit PDF Toolkit command line tools, please submit the problem report to the support group of Foxit at http://www.foxitsoftware.com/support/.
In order to better help you solve the problem, please provide the following information:

- Contact details
- Product name and its version
- Detailed description of the problem
- Any other related information, such as error screenshot

6.2 Contact Information

You can contact Foxit directly, please use the contact information as follows:

Foxit Support:

- http://www.foxitsoftware.com/support/

Sales Contact:

- Phone: 1-866-680-3668
- Email: sales@foxitsoftware.com

Support & General Contact:

- Phone: 1-866-MYFOXIT or 1-866-693-6948
- Email: support@foxitsoftware.com