Foxit

User Manual

Foxit® PDF Toolkit

Microsoft® Partner
Gold Independent Software Vendor (ISV)

©2016 Foxit Software Incorporated. All rights reserved.

TABLE OF CONTENTS

1 Introduction to Foxit PDF Toolkit

Foxit PDF Toolkit consists of a series of command line tools to perform batch PDF generation and processing. It includes 15 modules: Image2PDF, PDF2Image, Office2PDF, PDFWatermark, PDFHeaderFooter, PDFOptimizer, PDFRedactor, PDFMetadata, PDF2Text, Text2PDF, Html2PDF, PDF Page Organizer, PDFSecure, PDF2Word, and RMS. Each module offers a simple-to-use API for users who want to perform PDF manipulation through API.

In this manual, we will introduce the modules: Image2PDF, PDF2Image, Office2PDF, PDFWatermark, PDFHeaderFooter, PDFOptimizer, PDFRedactor, PDFMetadata, PDF2Text, Text2PDF, Html2PDF, PDF Page Organizer, PDFSecure, and PDF2Word. For the RMS PDF Protection module, the user manual will cover in detail the module's basic syntax in section 3.15 "RMS". Browse to the "rms" folder in the installation package for a more detailed introduction.

Image2PDF is an easy-to-use command line tool used to batch convert large volumes of image files into high-quality PDF files, without requiring additional software installation. Image2PDF currently supports the following image formats: BMP, PNG, JPEG, JPX, GIF, TIFF (or TIF, including the image with a single page or multiple pages). This module provides many features to customize the output properties of the generated PDF files.

PDF2Image batch converts large volumes of PDF files into image files, without requiring additional software installation. PDF2Image currently supports the following image formats: TIFF, GIF, BMP, PNG, JPEG, and Extended JPEG2000. It also provides many features to customize the output properties of the generated image files, such as color space and depth, image size and resolution, image rotation and quality, and more.

Office2PDF batch converts large volumes of Microsoft Office documents into professional-quality PDF files. Microsoft Office should be installed, because the Office2PDF tool saves the converted PDF files using the Microsoft Office Engine. Office2PDF currently supports the more popular document formats, which includes doc, docx, xls, xlsx, ppt and pptx. Additionally, the tool can convert and create professional-quality PDF files that support PDF/A Standard compliances.

PDFWatermark allows users to batch add a watermark into PDF files, without requiring additional software installation. The PDFWatermark tool applies a pre-made watermark to PDF files, which supports watermarks in text, image-based or PDF formats. The watermark is saved as a configuration file with the extension ".xml",

which is generated by the built-in Foxit Configuration Tool (Either *fpdfwmconf.exe* or *fpdfwmconf64.exe*, a GUI application, depending on your operating system).

PDFHeaderFooter batch adds headers and footers into large volumes of PDF files, without requiring additional software installation. The PDFHeaderFooter tool applies pre-made header/footer to PDF files. The headers and footers are saved as a configuration file with the extension ".xml", which is generated by the built-in Foxit Configuration Tool (Either *fpdfhfconf.exe* or *fpdfhfconf64.exe, a GUI application,* depending on your operating system).

PDFOptimizer batch reduces the size of PDF files to save disk space and make files easier to send and store, without requiring additional software installation. This tool optimizes PDF files by downsampling or compressing images, unembedding fonts, and discarding objects or user data. PDFOptimizer offers some options to clean up PDF files, such as using Flate to encode streams that are not encoded, converting LZW encoding to Flate encoding, and removing invalid bookmarks and invalid links.

PDFRedactor batch redacts PDF documents by completely removing sensitive content or private information from your PDF documents prior to making them available to others, without requiring additional software installation. With PDFRedactor, users can use redaction marks to redact or remove the sensitive contents that are visible in a PDF document and can specify custom text to appear over the redaction marks.

PDFMetadata batch sets document metadata information of PDF files, without requiring additional software installation. With PDFMetadata, users can set the title, author, subject, keywords and creator of PDF files, and view the metadata information. Setting metadata greatly helps in searching and organizing PDF files.

PDF2Text batch converts large volumes of PDF files into plain text files, without requiring additional software installation. With PDF2Text, users can extract text content from a textual PDF file and save the text as a plain text file, which is useful for text indexing or content retrieval.

Text2PDF batch converts large volumes of plain text files into high-quality PDF files with rich text formatting, without requiring additional software installation. This tool converts plain text to PDF files including settings of the page size, page margin, font style, font size, font color, password, and metadata.

Html2PDF converts a batch of local HTML files or a webpage (URL) into professional-quality PDF file(s), without requiring additional software installation. With Html2PDF tool, users can convert both the online webpage and local HTML files like invoices or reports into PDF file(s) with an industrial-strength HTML rendering engine.

PDF Page Organizer batch manages large volumes of PDF files including merging and splitting PDF files, and deleting pages from PDF files, without requiring additional software installation. It provides a flexible solution allowing users to process the whole PDF files, or just process some specific pages of the PDF files.

PDFSecure batch encrypts PDF files to protect sensitive information against misuse and unintentional alteration. With PDFSecure, users can set a user password for their PDF files to protect the PDF files from unauthorized opening and reading, and an owner password to only allow the people with this password to open the PDF files and make additional operations, such as changing the document's passwords and access permissions. It also supports certificate encryption to protect PDF files including public and private key encryption, and the Windows installed certificates in Windows certificate store. Therefore, only the people who have the password or certificate can access your PDF files.

PDF2Word batch converts large volumes of PDF files into high-quality Microsoft Word files, without requiring additional software installation. It greatly helps the users who want to edit or reuse the contents of the PDF document. With PDF2Word, users can export all the text, images and other contents from PDF files into Word files while preserving the original layout of the PDF.

1.1 Why Foxit PDF Toolkit is 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 generation and processing. 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 faster installation.
- **Perfect message mechanism** – Gives more perfect message hints if users encounter problems when using the tools to offer better user experience.
- **Robust and stable** – Ensures smooth running of the application and enhancement 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.

Foxit offers 24/7 support for its products and are fully supported by the PDF industry's largest development team of support engineers. Updates are released on a regular basis to improve user experience by adding new features and enhancements. Foxit PDF Toolkit is the best solution for PDF batch processing and integration of high performance features at a low cost!

1.2 Foxit PDF Toolkit Features

1.2.1 Image2PDF Features

- Batch convert image files into PDF files.
- Finish faster with multi-threaded processing.
- Make the complex simple with 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 output PDF file.
- Set page width and height for the output PDF file.
- Set margins for each PDF page.
- Support password encryption to secure PDF files.
- Add bookmarks to the output PDF file.
- Set document metadata information, including title, subject, keywords, author, and creator.
- Support wildcard characters in batch conversion, e.g., *.jpg, *.png.
- Offer simple-to-use API.

1.2.2 PDF2Image Features

- Batch convert PDF files into image files.
- Finish faster with multi-threaded processing.
- Make the complex simple with sub-folder recursion processing.
- Support single-file processing and single-folder processing.
- Support password-protected PDF files.
- Support various image formats, such as TIFF, GIF, BMP, PNG, JPEG, and Extended JPEG2000.
- Specify a page range to convert.
- Specify resolution (DPI) for the output image file.
- Set page width and height for the output image file.
- Rotate the PDF page to be rendered to the output image file by 0, 90, 180, or 270 degrees.
- Scale the page of the PDF to fit the page of the image.
- Option to only render the clip region of the PDF page.
- Align positions of the PDF page correspondingly with that of the output image page.

- Option to convert multipage PDF to multipage TIFF or GIF.
- Select color space and color depth for the output image file.
- Set the image quality for the output JPEG and Extended JPEG2000 images.
- Support wildcard characters in batch processing, e.g., *.pdf.
- Offer simple-to-use API.

1.2.3 Office2PDF Features

- Batch convert Microsoft Office files into PDF files.
- Finish faster with multi-threaded processing.
- Make the complex simple with sub-folder recursion processing.
- Support single-file processing and single-folder processing.
- Support password-protected Microsoft Office files.
- Support the popular Office document formats, including doc, docx, xls, xlsx, ppt and pptx.
- Add bookmarks to PDF files converted only from Microsoft Word documents.
- Provide scale options for Microsoft Excel conversion.
- Convert every sheet of a Microsoft Excel document into a single PDF file.
- Retain hyperlinks in PDF files from Microsoft Office files.
- Support converting to PDF/A compliant PDF files
- Support wildcard character in batch conversion, e.g., *.docx, *.pptx.
- Offer simple-to-use API.

1.2.4 PDFWatermark Features

- Batch add a watermark into PDF files.
- Finish faster with multi-threaded processing.
- Make the complex simple with sub-folder recursion processing.
- Support single-file processing and single-folder processing.
- Support password-protected PDF files.
- Support text, image, and PDF as the source of watermarks.
- Support stylized text watermarks that include text font, font size, font color, bold, and underline settings.
- Support image watermarks in various formats (BMP, DIB, JPG, JPEG, JPE, GIF, TIF, PNG, and TIFF).
- Support one page and multi-page PDF watermarks.
- Set and preview the watermark via the built-in GUI configuration tool.
- Rotate watermark in 360 degree rotations.
- Set opacity for watermarks.
- Scale watermark relative to target page size.
- Control the visibility of watermarks when printing.

- Control the on-screen visibility of watermarks.
- Keep watermark position and size for different page sizes.
- Determine watermark position within the document.
- Position watermark in front of or behind the document content.
- Specify a page range to add watermark.
- Export watermarks to XML files.
- Add watermark to PDF with XML configuration file.
- Set document metadata information, including title, subject, keywords, author, and creator.
- Support wildcard character in batch processing, e.g., *.pdf.
- Offer simple-to-use API.

1.2.5 PDFHeaderFooter Features

- Batch add headers and footers into PDF files.
- Finish faster with multi-threaded processing.
- Make the complex simple with sub-folder recursion processing.
- Support single-file processing and single-folder processing.
- Support password-protected PDF files.
- Set and preview the headers and footers via the built-in GUI configuration tool.
- Support stylized text for the headers and footers, including text font, font size, font color, underline and embedded font settings.
- Support margins settings.
- Determine header and footer position.
- Support dynamic content, such as date and page number, as a header or footer.
- Support page number and date format settings.
- Shrink a document to avoid overwriting the document's text and graphics.
- Keep position and size of headers and footers constant on different page sizes.
- Specify a page range to add headers and footers.
- Export headers and footers to XML files.
- Add headers and footers to PDF with XML configuration file.
- Set document metadata information, including title, subject, keywords, author, and creator.
- Support wildcard characters in batch processing, e.g., *.pdf.
- Offer simple-to-use API.

1.2.6 PDFOptimizer Features

- Batch optimize PDF files.
- Finish faster with multi-threaded processing.
- Make the complex simple with sub-folder recursion processing.

- Support single-file processing and single-folder processing.
- Support password-protected PDF files.
- Support downsampling and compression for color/grayscale and monochrome images.
- Unembed fonts in 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.
- Set document metadata information, including title, subject, keywords, author, and creator.
- Support wildcard characters in batch processing, e.g., *.pdf.
- Offer simple-to-use API.

1.2.7 PDFRedactor Features

- Batch redact or remove sensitive content from PDF files.
- Finish faster with multi-threaded processing.
- Make the complex simple with sub-folder recursion processing.
- Support single-file processing and single-folder processing.
- Support password-protected PDF files.
- Optionally set the search keyword mode for specific words/phrases or patterns.
- Specify the characters of keyword you want to redact.
- Specify a page range to apply redaction.
- Set a fill color to mark redactions.
- Overlay redaction marks with specified text.

- Set the alignment of the overlay text.
- Set the font style of the overlay text.
- Set the font size of the overlay text.
- Set the font color of the overlay text.
- Set document metadata information, including title, subject, keywords, author, and creator.
- Support wildcard characters in batch processing, e.g., *.pdf.
- Offer simple-to-use API.

1.2.8 PDFMetadata Features

- Batch set document metadata information.
- Finish faster with multi-threaded processing.
- Make the complex simple with sub-folder recursion processing.
- Support single-file processing and single-folder processing.
- Support password-protected PDF files.
- Set the title of PDF files.
- Set the subject of PDF files.
- Set the keywords of PDF files.
- Set the author of PDF files.
- Set the creator of PDF files.
- View PDF metadata information.
- Support wildcard characters in batch processing, e.g., *.pdf.
- Offer simple-to-use API.

1.2.9 PDF2Text Features

- Batch convert PDF files into plain text files.
- Finish faster with multi-threaded processing.
- Make the complex simple with sub-folder recursion processing.
- Support single-file processing and single-folder processing.
- Support password-protected PDF files.
- Specify a page range you want to extract text from.
- Get the total number of characters on each page.
- Print the characters number of each page to the screen.
- Optionally retain PDF page layout.
- Optionally display the page number in the output text files.
- Optionally convert each PDF page into individual text files.
- Convert to UTF-8 and UTF-16 encoded files.
- Add a page break to represent the end of each PDF page in the output text files.

- Support wildcard characters in batch processing, e.g., *.pdf.
- Offer simple-to-use API.

1.2.10 Text2PDF Features

- Batch convert text files into PDF files.
- Finish faster with multi-threaded processing.
- Make the complex simple with sub-folder recursion processing.
- Support single-file processing and single-folder processing.
- Support most commonly used languages.
- Set page width and height for the output PDF file.
- Set margins for each PDF page.
- Support font style, font size, font color, and line spacing settings.
- Optionally support pagination wherever there is a page break.
- Set open password for the output PDF file.
- Support password encryption to secure PDF files.
- Set document metadata information, including title, subject, keywords, author, and creator.
- Support wildcard characters in batch processing, e.g., *.txt.
- Offer simple-to-use API.

1.2.11 Html2PDF Features

- Batch convert local HTML files into PDF files.
- Convert a webpage (URL) into a PDF file.
- Finish faster with multi-threaded processing.
- Make the complex simple with sub-folder recursion processing.
- Support single-file processing and single-folder processing.
- Set page width and height for the output PDF file.
- Set margins for each PDF page.
- Set cache address to store HTML page resources temporarily.
- Set timeout to load webpages.
- Use the cookie to convert a URL that requires login to PDF.
- Apply JavaScript to webpages to handle some actions.
- Set all the page contents to one single PDF page.
- Optionally disable retaining hyperlinks in the generated PDF files.
- Rotate pages to 0, 90, 180, or 270 degrees.
- Preserve original page layout.
- Support wildcard characters in batch conversion, e.g., *.html, *.htm.
- Offer simple-to-use API.

1.2.12 PDF Page Organizer Features

- Batch manage PDF files including merging/splitting PDF files, and deleting pages from PDF files.
- Finish faster with multi-threaded processing.
- Make the complex simple with sub-folder recursion processing.
- Support single-file processing, single-folder processing and multi-file processing.
- Support password-protected PDF files.
- Extract some specific pages from input PDF file(s) to create a new PDF file.
- Merge multiple PDF files into one PDF file.
- Option to delete the bookmarks of the merged PDF file.
- Extract some specific pages from input PDF file(s) to create a set of single-page PDF files.
- Split PDF file(s) into single-page PDF files.
- Split PDF file(s) into a set of smaller PDF files with equal number of pages each.
- Delete some specific pages from input PDF file(s).
- Support wildcard characters in batch processing, e.g., *.pdf.
- Offer simple-to-use API.

1.2.13 PDFSecure Features

- Batch encrypt PDF files.
- Finish faster with multi-threaded processing.
- Make the complex simple with sub-folder recursion processing.
- Support single-file processing and single-folder processing.
- Support password-protected PDF files.
- Set user password and owner password for PDF files.
- Use ".cer" or ".pfx" file to encrypt PDF files.
- Use Windows installed certificates in Windows certificate store to encrypt PDF files.
- Support 128-bit AES, 256-bit AES, or 128-bit RC4 encryption.
- Option to encrypt the metadata of PDF files.
- Set the permission flags of PDF files, including:
  - Allow nothing (no permissions are granted).
  - Allow everything (all permissions are granted).
  - Allow printing with low resolution.
  - Allow printing with high resolution.
  - Allow filling in a form.
  - Allow commenting in the document.
  - Allow managing pages and bookmarks.
  - Allow modifying document.

  - Allow content copying for accessibility.
  - Allow extracting the contents of document.
- List the user's permissions to a PDF file.
- Set document metadata information, including title, subject, keywords, author, and creator.
- Support wildcard characters in batch processing, e.g., *.pdf.
- Offer simple-to-use API.

1.2.14 PDF2Word Features

- Batch convert PDF files to Microsoft Word files (DOCX).
- Finish faster with multi-threaded processing.
- Make the complex simple with sub-folder recursion processing.
- Support single-file processing and single-folder processing.
- Support password-protected PDF files.
- Specify a page range to convert.
- Preserve PDF page layout.
- Support wildcard characters in batch processing, e.g., *.pdf.
- Offer simple-to-use API.

1.3 Common Use Case Scenarios

1.3.1 Image2PDF Use Case Scenarios

- Batch convert images into PDF files for better image management.
- Create electronic books for users to scan paper documents to image files, and then convert them into a single PDF file.
- Make image albums for users to collect their photos and then convert them into a single PDF file.

1.3.2 PDF2Image Use Case Scenarios

- Batch convert PDF files into high-quality image files which can be used for further analysis in the medical image processing field.
- Make the migration of image-based archives quick and convenient.
- Fax servers.

1.3.3 Office2PDF Use Case Scenarios

- Batch convert Microsoft Office files into PDF files.
- Set the converted PDF files to be compliant with PDF/A standard for better archiving.

1.3.4 PDFWatermark Use Case Scenarios

- Batch add a watermark into PDF files to protect users and their copyrights. Users can use company logos, author signatures, products, PDF files or web addresses as watermarks 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.

1.3.5 PDFHeaderFooter Use Case Scenarios

- Batch add headers and footers into PDF files to include information such as the date, page numbers, or the title of the document.

1.3.6 PDFOptimizer Use Case Scenarios

- Batch optimize large PDF files for electronic document exchange or space-saving document archiving. Large PDF files are not suitable for electronic document exchange or document archiving. Users can optimize large PDF documents instead of converting them into other formats.

1.3.7 PDFRedactor Use Case Scenarios

- Batch redact or remove sensitive content or private information from PDF files. Users can redact or remove sensitive text that are visible in PDF documents before distributing them to others, which could also help prevent the receivers from tracing the documents back to the sender.

1.3.8 PDFMetadata Use Case Scenarios

- Batch set document metadata information of PDF files. Users can set the title, author, subject, keywords and creator of PDF files, which could greatly help in searching and organizing PDF files. For example, the keywords can be particularly useful for narrowing searches.

1.3.9 PDF2Text Use Case Scenarios

- Batch convert PDF files into plain text files for better archiving and indexing PDF files. It is indeed helpful for text indexing and content retrieval by creating a full-text searchable archive database.

1.3.10 Text2PDF Use Case Scenarios

- Batch convert plain text files into PDF files for better viewing or editing. It is not easy for users to view or edit plain text files, and in this case, users can convert them into PDF files for further processing.

1.3.11 Html2PDF Use Case Scenarios

- Convert a batch of local HTML files or a webpage (URL) into PDF files, making them easier to print or archive. For some large HTML files or a webpage which contain(s) many contents, it is not convenient to print or archive them directly. In this case, users can convert them into PDF file(s) which are widely used in printing and document archiving.

1.3.12 PDF Page Organizer Use Case Scenarios

- Batch manage PDF files for better archiving.
- Assemble PDF books, reports, brochures by merging different individual documents or fragments thereof.
- Split some large PDF files into a set of smaller PDF files which are suitable for electronic document exchange.
- Delete some useless or blank pages from some PDF files.

1.3.13 PDFSecure Use Case Scenarios

- Batch encrypt PDF files to protect sensitive information against misuse and unintentional alteration.
- Assign different permissions of the PDF files to different users for better rights management.

1.3.14 PDF2Word Use Case Scenarios

- Batch convert PDF files into high-quality Word files for better editing and reusing the documents. It is not easy for users to make some modifications in the PDF files, or just to copy the text or images in the PDF files, and then paste them into another document. In this case, users can convert the PDF files into Word files which excel at editing and processing documents.

1.4 System Requirements

Windows Platform：

- Windows Server 2012
- Windows Server 2008 R2
- Windows Server 2003
- Windows 10
- Windows 8.1
- Windows 7

Note *For the Office2PDF module, please make sure that you have already installed Microsoft Office 2007 SP2 or later, and the virtual printers. If Microsoft Office 2007 is not the SP2 version, please download the "Microsoft Save as PDF" plugin. For Microsoft Office 2016, please ensure that "Microsoft Print to PDF" has been set as the default Microsoft virtual printer, otherwise the Office2PDF may not work and cause a crash.*

1.5 About This Manual

This manual aims at introducing how to use the command line of Foxit PDF Toolkit. It is intended for the users who want to batch generate or process PDF files.

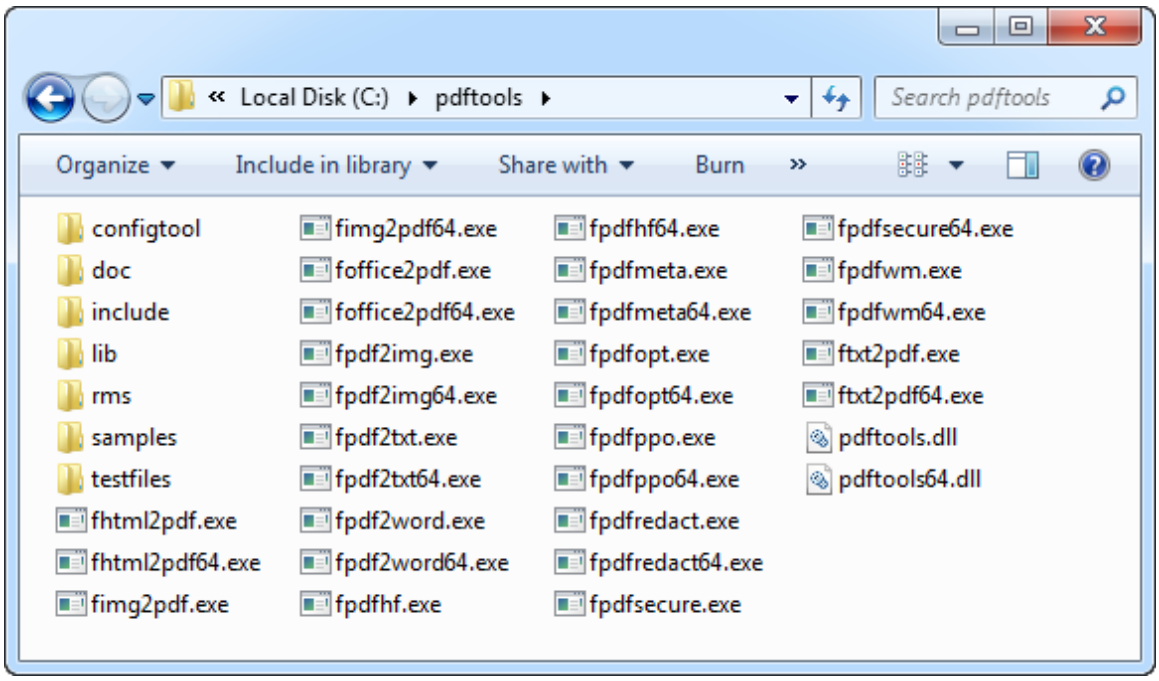
- Section 1: gives an introduction of Foxit PDF Toolkit.
- Section 2: illustrates how to install and uninstall Foxit PDF Toolkit.
- Section 3: describes basic usage of Foxit PDF Toolkit.
- Section 4: introduces Foxit Configuration Tool of PDFWatermark and PDFHeaderFooter modules.
- Section 5: shows how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

2 Installing and Uninstalling Foxit PDF Toolkit

2.1 Installation

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

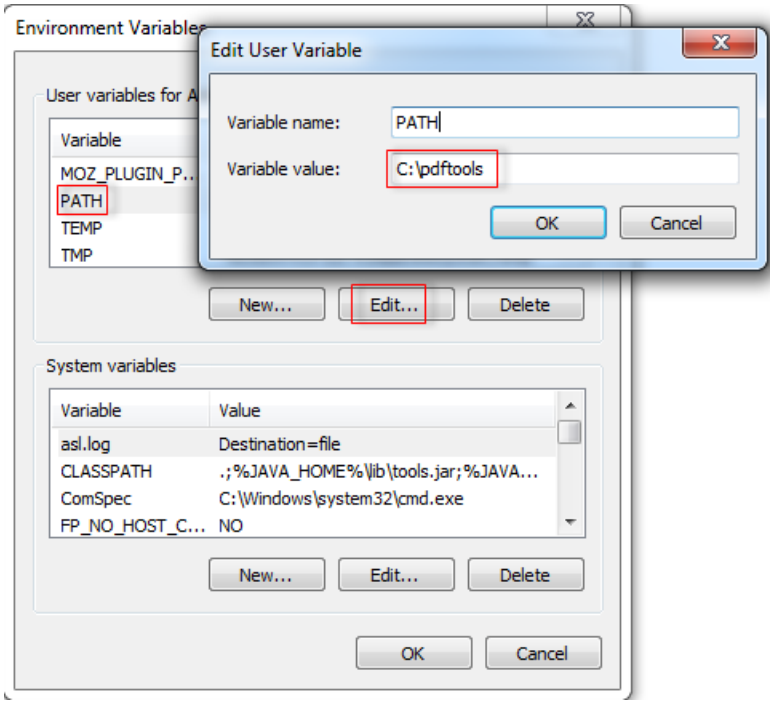
configtool:	configuration tools for pdfwatermark and pdfheaderFooter
doc:	user manual for Foxit PDF Toolkit
include:	header file for Foxit PDF Toolkit
lib:	third-part libraries and databases for Foxit PDF Toolkit
rms:	RMS command line tool
samples:	some batch samples for Foxit PDF Toolkit
testfiles: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



Foxit PDF Toolkit package

After that, open a terminal session and navigate to the installation location to run your application (except for PDFWatermark and PDFHeaderFooter modules, which are used only after using the Foxit Configuration Tool to generate configuration file containing the setting information of watermark or header/footer). For the Office2PDF module, please make sure that you have already installed Microsoft Office 2007 SP2 or later, and the virtual printers. If Microsoft Office 2007 is not the SP2 version, please download the "Microsoft Save as PDF" plugin. For Microsoft Office 2016, please ensure that "Microsoft Print to PDF" has been set as the default Microsoft virtual printer, otherwise the Office2PDF may not work and cause a crash.

Tips: You can set the installation path to Environment Variables, which allows you to use the commands in the terminal window directly without needing to change the directory to the installation location. Go to **Start**-> **Control Panel**-> **System**-> **Advanced system settings** -> **Advanced** -> **Environment Variables**, in the "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**.



Set installation path to Environment Variables

2.2 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 a 30-day free trial, during which the pages of all generated PDF files will contain our watermark, and for the PDF2Text and PDF2Word modules, you can only convert the first ten PDF pages into plain text files/Word file. After the evaluation period, customers that want to continue to

use the product should purchase a licensed version from Foxit website at https://www.foxitsoftware.com/products/pdf-toolkit/.

2.3 About License

Foxit PDF Toolkit provides three types of licenses for customers to choose.

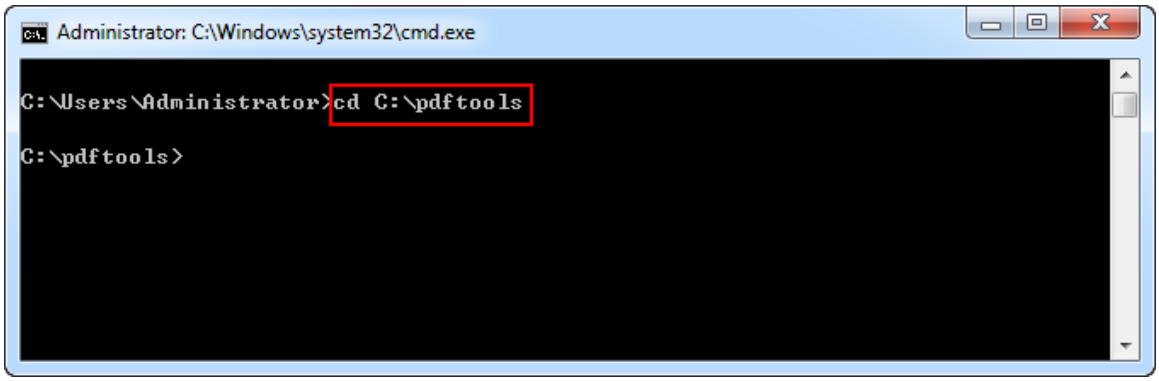
- **Desktop License**- It is only available on desktop systems, which is perfect for personal or small business use. Each license is good for one user on one machine.
- **Server License**- It is available on servers, with 8 CPUs, which is good for small-to medium-sized businesses that need higher performance on a single server. If your server has more than 8 CPUs, please contact Foxit sales team to purchase Enterprise License.
- **Enterprise License**- It is intended for large companies that need to process a large number of PDF documents on multiple high-performance servers. Enterprise License also includes features for integrating the Foxit PDF Toolkit into your own applications. Please contact Foxit sales team to purchase Enterprise License.

2.4 Registration

When you get the activation code for the module you want to purchase, please use the argument "**-register <code> <licensee>**" to activate it in the Command Prompt window. The following two steps show how to open a Command Prompt window based on Windows 7 system.

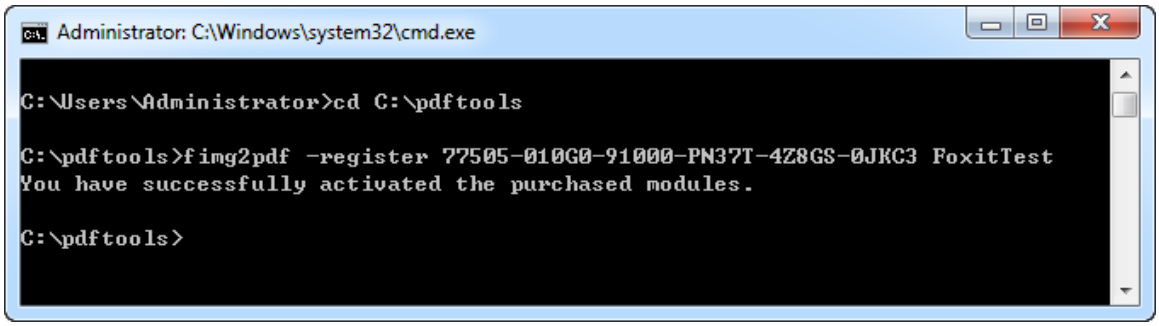
a. Click on the **Start** menu.
b. Type "***cmd***" in the *Search programs and files* box and press **Enter**.

In the opening Command Prompt window, type "***cd C:\pdftools***" to navigate to the installation location as follows.



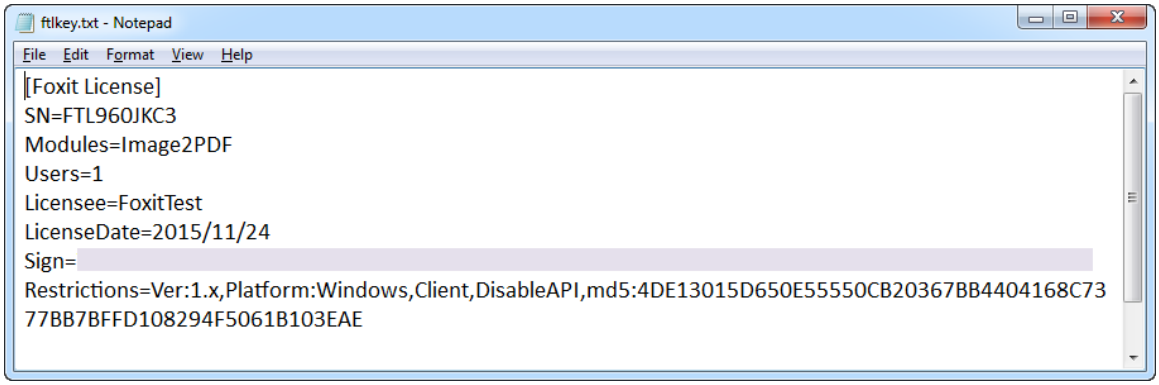
Navigate to the installation location

Assume you get an activation code (77505-010G0-91000-PN37T-4Z8GS-0JKC3) for the Image2PDF module, you may type "***fimg2pdf -register 77505-010G0-91000-PN37T-4Z8GS-0JKC3 FoxitTest***" in the Command Prompt window as shown below.



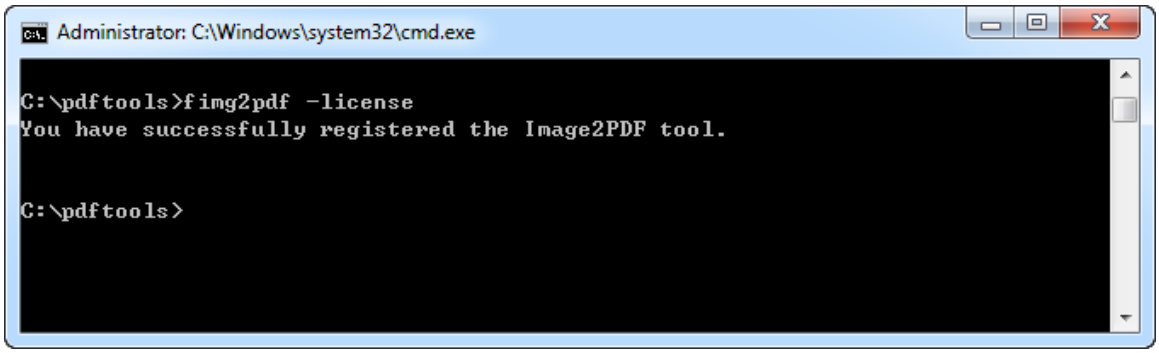
Register Foxit Convert2PDF tool

Here, "*FoxitTest*" is the licensee name you input. After activation, a key file named "**ftlkey.txt**" will be generated in the installed path with the contents shown in the following figure.



The content of the generated key file

Then you can run "***-fimg2pdf -license***" in the Command Prompt window to check the license agreement as follows.



Check the license agreement

2.5 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>> [[-width <PDF width>] [-height <PDF height>]]*
[-dpi <resolution>] [-margin <left [top right bottom]>] [-b] [-sp <password>]
[-title <title>] [-subject <subject>] [-keywords <keywords>] [-author <author>] [-creator <creator>]
[-r [recursion]] [-t <thread>] [-log <logfile>] [-l <level>]
fimg2pdf *-register <code> <licensee>*
fimg2pdf *-license*
fimg2pdf *-version/-v*
fimg2pdf *-help/-h*
fimg2pdf *-copyright*

Note:

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

Only the <*-i <srcfile/srcfolder>>* and <*-o <destfile/destfolder>>* 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

Note *For some arguments whose values are strings, users can choose whether to add quotation marks (" ") to the strings. In the manual we have added notes where adding quotation marks (" ") is required.*

Option	Parameter	Description
-i	*<-i <string>>* *e.g.* *-i c:\input\1.jpg* *-i "c:\input\1.jpg" "c:*	Specifies the input file to be converted. ▪ The input string can be the name of a single image file (.bmp, .png, .jpg, .jpx, .gif, .tif, .tiff), multiple

Option	Parameter	Description
	input\2.tif" *-i c:\input* *-i "c:\input*.jpg"*	image files, or a folder. ▪ The file name can contain the wildcard character (*). For example, use *.tiff to include all TIFF image files in a given folder. **Note** Wildcard character (*.*) is currently not supported.
-o	*<-o <string>>* *e.g.* *-o d:\output\one.pdf* *-o d:\output*	Specifies the path of the output PDF file or folder. ▪ If user specifies a path of a PDF file, (e.g. *-o d:\output\one.pdf*), all input image files will be combined into a single PDF file. ▪ If user specifies 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.
-width	*[-width <points>]* *e.g.* *-width 612*	Sets the page width of the output PDF file in points. Default: Width of input image. **Note** The **-width** and **-height** options should be used together with a set value greater than 0.
-height	*[-height <points>]* *e.g.* *-height 792*	Sets the page height of the output PDF file in points. Default: Height of input image. **Note** The **-width** and **-height** options should be used together with a set value greater than 0.
-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

Option	Parameter	Description
		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 should be less than the width of the page, and the sum of the top and bottom values should be less than the height of the page. ***-margin left top right bottom*** **-margin 20**: sets the left margin to 20 points. **-margin 10 20**: sets the left margin to 10 points and the top margin to 20 points. **-margin 10 20 0**: sets the left margin to 10 points, the top margin to 20 points, and the right margin to 0 points. **-margin 10 20 0 40**: sets 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.
-b	*[-b]* *e.g.* *-b*	Uses the filename of the image(s) to create bookmark(s) for the output PDF.
-sp	*[-sp <string>]* *e.g.* *-sp 123* *-sp welcome*	Sets the document open password of the output PDF as the "string". By default, there is no password.
-title	*[-title <string>]* *e.g.* *-title "Foxit PDF Toolkit User Manual"*	Sets title of PDF files.
-subject	*[-subject <string>]* *e.g.* *-subject "Foxit PDF Toolkit"*	Sets subject of PDF files.

Option	Parameter	Description
-keywords	*[-keywords <string>]* *e.g.* *-keywords "Foxit"*	Sets keywords of PDF files.
-author	*[-author <string>]* *e.g.* *-author "Jessie"*	Sets author of PDF files.
-creator	*[-creator <string>]* *e.g.* *-creator "Foxit PhantomPDF"* *-creator "Foxit Reader"* *-creator "Microsoft® Word 2013"*	Sets creator of PDF files. **Note** It indicates the name of the application that created the source document from which a PDF is generated. For example, if the source document was created from Microsoft® *Word 2013, then you can set the creator to* "Microsoft® *Word 2013".*
-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. • **-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** ▪ If no integer value is specified, or if the integer value is 0, then full folders will be searched. By default, the number of layers to recurse is 1, which means that only the current folder will be searched and not sub-folders. ▪ The input folder will be skipped if it is secured and the messages will be displayed.
-t	*[-t <integer>]* *e.g.* *-t 1* *-t 2*	Specifies the number of CPU threads to use. The default value is 1.
-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 messages only concerning out of memory errors. • **-l 2**: logs failure messages concerning the errors caused during execution or those returned from underlying libraries, as well as those for level 1. • **-l 3**: logs warning messages concerning the PDF files that are overwritten, as well as those for level 2. • **-l 4**: logs informational messages, as well as those for level 3. **Note** The argument (**-l**) is valid only when (**-log**) is used.
-register	*[-register <String> <String>]* *-register <code> <licensee>* *e.g.* *-register xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx Foxit*	Registers the command line tool. ▪ **<code>**: the activation code from Foxit. ▪ **<licensee>**: the Licensee name designated by the users.
-help/-h	*[-help/-h]* *e.g.* *-help* *-h*	Prints the usage information.
-version/-v	*[-version/-v]* *e.g.* *-version* *-v*	Prints the version information.
-license	*[-license]* *e.g. -license*	Prints the license agreement.

3.1.3 Basic Usage

3.1.3.1 Input and Output (-i, -o)

a) Input (-i)

- The input should be a single image file, multiple image files, or a folder. Users are not able to input multiple folders, as well as a mixture composed 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, it is recommended to enclose each input path with quotation marks (" "). In this manual, we add (" ") whenever the input files are multiple image files.*

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

-i test\3.bmp	("test" folder 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 in 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, it is recommended to enclose the input files with quotation marks (" "). In this manual, we add (" ") whenever the input files contain wildcard characters.*

Note *If the input paths or file names contain spaces, you must enclose the input files with quotation marks (" "), such as -i "c:\input file\1.jpg", -i "test\user manual.png". This rule is also used for some other arguments whose values are strings.*

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. 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. Users can input just 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 if it includes 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, where 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 the "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 PDF/Page Settings (-width, -height, -dpi, -b, -margin, -sp)

a) 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 should be used together with a set value 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
```

b) 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
```

c) 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
```

d) 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 should be less than the width of the page, and the sum of the top and bottom values should 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
```

e) Password setting (-sp)

- The optional argument (**-sp**) is used to set the open password for 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
```

3.1.3.3 Document Metadata Settings (-title, -subject, -keywords, -author, -creator)

a) Title (-title)

- The optional argument (**-title**) is used to set title of PDF files.

Usage Example

1) Set document title to "Foxit PDF Toolkit User Manual" (-title "Foxit PDF Toolkit User Manual")

```
fimg2pdf -i "c:\input\1.jpg" "c:\input\2.tif" -o d:\output\one.pdf -title "Foxit PDF Toolkit User Manual"
fimg2pdf -i c:\input -o d:\output\one.pdf -title "Foxit PDF Toolkit User Manual"
```

```
fimg2pdf -i "c:\input\*.jpg" -o d:\output\one.pdf -title "Foxit PDF Toolkit User Manual"
```

b) Subject (-subject)

- The optional argument (**-subject**) is used to set subject of PDF files.

Usage Example

1) Set document subject to "Foxit PDF Toolkit" (-subject "Foxit PDF Toolkit")

```
fimg2pdf -i "c:\input\1.jpg" "c:\input\2.tif" -o d:\output\one.pdf -subject "Foxit PDF Toolkit"
fimg2pdf -i c:\input -o d:\output\one.pdf -subject "Foxit PDF Toolkit"
fimg2pdf -i "c:\input\*.jpg" -o d:\output\one.pdf -subject "Foxit PDF Toolkit"
```

c) Keywords (-keywords)

- The optional argument (**-keywords**) is used to set keywords of PDF files.

Usage Example

1) Set document keywords to "toolkit" (-keywords "toolkit")

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

d) Author (-author)

- The optional argument (**-author**) is used to set author of PDF files.

Usage Example

1) Set document author to "Jessie" (-author "Jessie")

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

e) Creator (-creator)

- The optional argument (**-creator**) is used to set file creation application information of PDF files.

Usage Example

1) Set document creator to "Foxit Reader" (-creator "Foxit Reader")

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

3.1.3.4 Recursion Depth of Sub-folders (-r)

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

Usage Examples

1) Search the full 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, the current folder will be searched 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
```

3.1.3.5 Multi-thread Support (-t)

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

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

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
```

3.1.3.6 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 argument (**-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 the users.

Usage Example

1) Register the image2pdf tool with the code "xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx" and the licensee "Foxit" (-register xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx Foixt)

   fimg2pdf -register xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx Foixt

c) License agreement (-license)

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

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

f) Copyright information (-copyright)

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

Usage Example

1) Print the copyright information (-copyright)

```
fimg2pdf -copyright
```

3.2 PDF2Image

3.2.1 Basic Syntax

fpdf2img *<-i <srcfile/srcfolder>> <-o <destfolder>>*
[-range <page range>] [-ext <extension> [quality]] [-interlace [adam7]]
[-multipages/-mp [chunk]] [-width <image width>] [-height <image height>]
[-clip <lower-left_x> <lower-left_y> <upper-right_x> <upper_right_y>]
[-dpi <dpi>] [-colorspace/-cs <colorspace> [depth]]
[-align <position>] [-rotate <0/90/180/270>] [-fitpage]
[-op <password>] [-r [recursion]] [-t <thread>] [-log <logfile>] [-l <level>]
fpdf2img *-register <code> <licensee>*
fpdf2img *-license*
fpdf2img *-version/-v*
fpdf2img *-help/-h*
fpdf2img *-copyright*

Note:

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

Only the <*-i <srcfile/srcfolder>>* and <*-o <destfolder>>* arguments are actually required. All others are optional, which are available for controlling the conversion 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

Note *For some arguments whose values are strings, users can choose whether to add quotation marks (" ") to the strings. In the manual we have added notes where adding quotation marks (" ") is required.*

Option	Parameter	Description
-i	*<-i <string>>* *e.g.* *-i c:\input\1.pdf* *-i c:\input* *-i "c:\input*.pdf"*	Specifies the input file to be converted. ▪ The input string can be the name of a single PDF file or a folder. ▪ The file name can contain the wildcard character (*). For example, use *.pdf to include all PDF files in a given folder. **Note** Wildcard character (*.*) is currently not supported.
-o	*<-o <string>>* *e.g.* *-o d:\output*	Specifies the path of the output folder. **Note** The specified output path must already exist.
-range	*-rang <String>* *e.g.* *-range "1,5,9"* *-range "all"* *-range "even"* *-range "odd"* *-range "2-10,30"* *-range "10,50-"* *-range "odd,100-"*	Specifies a page range to convert. By default, all of the pages will be converted. ▪ Converts page 1,5, and 9: **-range "1,5,9"** ▪ Converts all pages: **-range "all"** ▪ Converts all even pages: **-range "even"** ▪ Converts all odd pages: **-range "odd"** ▪ Converts pages in the range from 2-10 and page 30: **-range "2-10,30"** ▪ Converts page 10 and all pages in the range from 50 to the last page: **-range "10,50-"** ▪ Converts all odd pages and all pages in the range from 100 to the last page: **-range "odd,100-"**

Option	Parameter	Description
-ext	*[-ext <string> [int]]* *[-ext <extension> [quality]]* *e.g.* *-ext tiff* *-ext png* *-ext bmp* *-ext gif* *-ext jpg 100* *-ext jpx 0*	Specifies the output image format and the corresponding image quality. By default, the output image format is BMP. ▪ **<extension> :** Chooses the output image file extensions. **jpg/jpeg**: JPEG (Joint Photographic Expert Group) **png**: PNG (Portable Network Graphics) **gif**: GIF (Graphics Interchange Format) **bmp/dib**: BMP (Window Bitmap) **tif/tiff**: TIFF (Tagged Image File Format) **jpx:** Extended JPEG2000 ▪ [**quality**]: The image quality. It is valid only for .jpx, .jpg and .jpeg images and will be ignored by the other image formats. For **.jpx**, allowable range: 0-5, default value: 0. 0 means lossless compression, and higher values usually result in better compression at the expense of image quality. For **.jpg** and **.jpeg**, allowable range: 1-100, default value: 100. 100 means the best image quality with the minimum image data loss. Lower values usually result in better compression at the expense of image quality.
-interlace	*[-interlace [adam7]]* *e.g.* *-interlace* *-interlace adam7*	Uses adam7 algorithm to generate the output image files. **Note** The argument (**-interlace**) is valid only for .png images. If you set it without a value, just like "-interlace", the default value is adam7.
-multipage/-mp	*[-multipage/-mp [int]]* *[-multipage/-mp [chunk]]* *e.g.* *-multipage 5* *-mp 5* *-multipage 10* *-mp 10*	Specifies how many PDF pages will be converted to an image file. All the PDF pages will be converted to an image file if the number of PDF pages is smaller than the specified value. **Note** The argument (**-multipage/-mp**) is valid only for .gif, .tiff and .tif images.

Option	Parameter	Description
-width	*[-width <pixels>]* *e.g.* *-width 612*	Sets the width of the output image file in pixels. **Note** If the arguments (**-width**) and (**-height**) are not set, the default image width is calculated by the formula DPI*(PDF_width/72). For example, if the width of the input PDF is 612 points, and the value of DPI is 96 (default), then the default image width is 816 (96*(612/72)) pixels.
-height	*[-height <pixels>]* *e.g.* *-height 792*	Sets the height of the output image file in pixels. **Note** If the arguments (**-width**) and (**-height**) are not set, the default image height is calculated by the formula DPI*(PDF_height/72). For example, if the height of the input PDF is 792 points, and the value of DPI is 96 (default), then the default image height is 1056 (96*(792/72)) pixels.
-clip	*[-clip <points> <points> <points> <points>]* *[-clip <lower-left_x> <lower-left_y> <upper-right_x> <upper_right_y>]* *e.g.* *-clip 100 120 500 600*	Specifies a clip region of the PDF page to be converted. If the third value is set to 0, which is invalid, then the PDF page width will be used as the third value automatically; and if the fourth value is set to 0, the PDF page height will be used as the fourth value automatically.
-dpi	*[-dpi <integer>]* *e.g.* *-dpi 150* *-dpi 300*	Specifies DPI (Dots Per Inch) resolution of the output image files. The default value is 96. **Note** The argument (**-dpi**) is valid only for .jpeg, .jpg, .gif, .tiff, .tif, and .png images.
-colorspace/-cs	*[-colorspace/-cs <string> [int]]* *[-colorspace/-cs <colorspace> [depth]]* *e.g.* *-colorspace rgb 24* *-colorspace cmyk 32* *-colorspace grayscale 1* *-cs rgb 32* *-cs cmyk 32* *-cs grayscale 8*	Specifies the color space and color depth to render the output image files. ▪ **<colorspace>:** Chooses the color space. Allowable values are **rgb**, **cmyk** and **grayscale**. For **.jpg, .jpeg**, **.png**, **.gif**, **.bmp**, and **.dib**, it supports rgb and grayscale. For **.tiff**, and **.tif**, it supports rgb, cmyk, and grayscale.

Option	Parameter	Description
		For .**jpx**, it only supports rgb currently. ▪ [**depth**]: Chooses the color depth. The value varies with different color spaces. For **rgb**, allowable values: 1, 8, 24, and 32. For **cmyk**, allowable values: 32. For **grayscale**, allowable values: 1, 8. **Note** The correspondence between color spaces, color depth and image formats are as follows: ▪ For .**jpg** and .**jpeg**, allowable values: rgb (24), grayscale (8). The default color space and color depth is rgb (24). ▪ For .**png**, allowable values: rgb (1, 8, 24, 32), grayscale (1, 8). The default color space and color depth is rgb (32). ▪ For .**gif**, allowable values: rgb (8), grayscale (8). The default color space and color depth is rgb (8). ▪ For .**bmp** and .**dib**, allowable values: (1, 8, 24, 32), grayscale (1, 8). The default color space and color depth is rgb (32). ▪ For **.tiff** and **.tif**, allowable values: rgb (24), cmyk (32), grayscale (1, 8). The default color space and color depth is rgb (24). ▪ For .**jpx**, allowable values: rgb (24). The output images will have better visual effect using the default color space and color depth.

Option	Parameter	Description
-align	*[-align <string/int>]* *[-align <position>]* *e.g.* *-align "tl" or -align 1* *-align "tm" or -align 2* *-align "tr" or -align 3* *-align "ml" or -align 4* *-align "mm" or -align 5* *-align "mr" or -align 6* *-align "bl" or -align 7* *-align "bm" or -align 8* *-align "br" or -align 9*	Aligns positions of the PDF page correspondingly with that of the output image page. The alignment can be set as follows: (the default value is mm or 5) • **tl or 1**: the top-left point of the PDF page matches the top-left point of the image page. • **tm or 2**: the top-middle point of the PDF page matches the top-middle point of the image page. • **tr or 3**: the top-right point of the PDF page matches the top-right point of the image page. • **ml or 4**: the middle-left point of the PDF page matches the middle-left point of the image page. • **mm or 5**: the middle-middle point of the PDF page matches the middle-middle point of the image page. • **mr or 6**: the middle-right point of the PDF page matches the middle-right point of the image page. • **bl or 7**: the bottom-left point of the PDF page matches the bottom-left point of the image page. • **bm or 8**: the bottom-middle point of the PDF page matches the bottom-middle point of the image page. • **br or 9**: the bottom-right point of the PDF page matches the bottom-right point of the image page.
-rotate	*[-rotate <0/90/180/270>]* *e.g.* *-rotate 0* *-rotate 90* *-rotate 180* *-rotate 270*	Specifies the rotation of the PDF page to be rendered to the output image file. The value *should* be 0, 90, 180 or 270, and the default value is 0.

Option	Parameter	Description
-fitpage	*[-fitpage]* *e.g.* *-fitpage*	Scales the page of the PDF to fit the page width or height of the image.
-op	*[-op<string>]* *e.g.* *-op 123* *-op welcome*	Specifies the password for the input file. Not required if the input file is not password protected.
-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. • **-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** ▪ If no integer value is specified, or if the integer value is 0, then full folders will be searched. By default, the number of layers to recurse is 1, which means that only the current folder will be searched and not sub-folders. ▪ The input PDF file or folder will be skipped if it is secured and the messages will be displayed.
-t	*[-t <integer>]* *e.g.* *-t 1* *-t 2* …	Specifies the number of CPU threads to use. The default value is 1.
-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*	Sets the log level. The default is 4. • **-l 1**: logs messages only concerning out of memory errors. • **-l 2**: logs failure messages concerning the errors caused during execution or those

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

3.2.3 Basic Usage

3.2.3.1 Input and Output (-i, -o)

a) Input (-i)

➢ The input should be a single PDF file or a folder. Users are not able to input multiple PDF files or folders, as well as a mixture composed 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 if the input file is in the current working folder. Users can input just the name of the PDF file or folder, instead of an absolute path. For example:

  -i test\1.pdf ("test" folder 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, it is recommended to enclose the input files with quotation marks (" "). In this manual, we add (" ") whenever the input files contain wildcard characters.*

Note *If the input paths or file names contain spaces, you must enclose the input files with quotation marks (" "), such as -i "c:\input file\1.pdf", -i "test\user manual.pdf". This rule is also used for some other arguments whose values are strings.*

b) Output (-o)

- The output should be a single folder. For example:

  -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. Users can input just the name of the output folder, instead of an absolute path. For example:

  -o output ("output" folder is in the current working folder)

Usage Examples

1) Convert a single PDF file to image file(s):

```
fpdf2img -i c:\input\1.pdf -o d:\output
fpdf2img -i test\2.pdf -o output
```

2) Convert the PDF files in a folder to image files:

```
fpdf2img -i c:\input -o d:\output
fpdf2img -i test -o output
fpdf2img -i c:\input\*.pdf -o d:\output
fpdf2img -i "test\*.pdf" -o output
```

3.2.3.2 Page range to convert (-range)

- The optional argument (**-range**) is used to specify a page range to convert. If this argument is not set, all of the pages will be converted. For more details about this argument, please refer to section 3.2.2 "Command Line Summary".

Usage Example

1) Convert pages 2,3 and 8 (-range "2,3,8")

```
fpdf2img -i c:\input\1.pdf -o d:\output -range "2,3,8"
fpdf2img -i test\2.pdf -o output -range "2,3,8"
fpdf2img -i c:\input -o d:\output -range "2,3,8"
fpdf2img -i test -o output -range "2,3,8"
fpdf2img -i "c:\input\*.pdf" -o d:\output -range "2,3,8"
```

2) Convert all even pages (-range "even")

```
fpdf2img -i c:\input\1.pdf -o d:\output -range "even"
fpdf2img -i test\2.pdf -o output -range "even"
fpdf2img -i c:\input -o d:\output -range "even"
fpdf2img -i test -o output -range "even"
fpdf2img -i "c:\input\*.pdf" -o d:\output -range "even"
```

3) Convert pages in the range from 2-10 and page 30 (-range "2-10,30")

```
fpdf2img -i c:\input\1.pdf -o d:\output -range "2-10,30"
fpdf2img -i test\2.pdf -o output -range "2-10,30"
fpdf2img -i c:\input -o d:\output -range "2-10,30"
fpdf2img -i test -o output -range "2-10,30"
fpdf2img -i "c:\input\*.pdf" -o d:\output -range "2-10,30"
```

4) Convert all odd pages and all pages in the range from 100 to the last page (-range "odd,100-")

```
fpdf2img -i c:\input\1.pdf -o d:\output -range "odd,100-"
fpdf2img -i test\2.pdf -o output -range "odd,100-"
fpdf2img -i c:\input -o d:\output -range "odd,100-"
fpdf2img -i test -o output -range "odd,100-"
fpdf2img -i "c:\input\*.pdf" -o d:\output -range "odd,100-"
```

3.2.3.3 Image format and image quality settings (-ext)

- The optional argument (**-ext**) is used to specifies the output image format and the corresponding image quality. If this argument is not set, the output image format is BMP by default.

  The usage of this argument is: ***-ext <extension> [quality]***.

  - <**extension**> is the output image file extensions.
  - [**quality**] is the image quality. It is valid only for .jpx, .jpg and .jpeg images and will be ignored by the other image formats.

  The supported image file extensions and corresponding image quality are shown in the following table:

Format	Extension	Quality (allowable range)
JPEG	jpg, jpeg	1-100, 100 (by default)
PNG	png	---
GIF	gif	---
BMP	bmp ,dib	---
TIFF	tiff, tif	---
Extended JPEG2000	jpx	0-5, 0 (by default)

  For .**jpg** and .**jpeg**, 100 means the best image quality with the minimum image data loss. Lower values usually result in better compression at the expense of image quality.

  For .**jpx**, 0 means lossless compression, and higher values usually result in better compression at the expense of image quality.

Usage Example

1) Set the output image format to TIFF (-ext tiff or -ext tif)

```
fpdf2img -i c:\input\1.pdf -o d:\output -ext tiff
fpdf2img -i test\2.pdf -o output -ext tiff
fpdf2img -i c:\input -o d:\output -ext tiff
fpdf2img -i test -o output -ext tiff
```

```
fpdf2img -i "c:\input\*.pdf" -o d:\output -ext tiff
```

2) Set the output image format to JPEG with image quality 95 (-ext jpg 95 or -ext jpeg 95)

```
fpdf2img -i c:\input\1.pdf -o d:\output -ext jpg 95
fpdf2img -i test\2.pdf -o output -ext jpg 95
fpdf2img -i c:\input -o d:\output -ext jpg 95
fpdf2img -i test -o output -ext jpg 95
fpdf2img -i "c:\input\*.pdf" -o d:\output -ext jpg 95
```

3) Set the output image format to JPX with image quality 2 (-ext jpx 2)

```
fpdf2img -i c:\input\1.pdf -o d:\output -ext jpx 2
fpdf2img -i test\2.pdf -o output -ext jpx 2
fpdf2img -i c:\input -o d:\output -ext jpx 2
fpdf2img -i test -o output -ext jpx 2
fpdf2img -i "c:\input\*.pdf" -o d:\output -ext jpx 2
```

3.2.3.4 Adam7 algorithm (-interlace)

- The optional argument (**-interlace**) is used to apply adam7 algorithm to generate the output image files. It is valid only for .png images. If you set it without a value, just like "-interlace", the default value is adam7.

Usage Example

1) Set the output image format to PNG and use adam7 algorithm to generate the output image files (-ext png -interlace or -ext png -interlace adam7)

```
fpdf2img -i c:\input\1.pdf -o d:\output -ext png -interlace adam7
fpdf2img -i test\2.pdf -o output -ext png -interlace adam7
fpdf2img -i c:\input -o d:\output -ext png -interlace adam7
fpdf2img -i test -o output -ext png -interlace adam7
fpdf2img -i "c:\input\*.pdf" -o d:\output -ext png -interlace adam7
```

3.2.3.5 Generate multipage image file (-multipage/-mp)

- The optional argument (**-multipage/-mp**) is used to specify how many PDF pages will be converted to an image file. All the PDF pages will be converted to an image file if the number of PDF pages is smaller than the specified value. This argument is valid only for .gif, .tiff and .tif images.

Usage Example

1) Convert PDF file(s) to multiple smaller .tiff files with 3 pages per file (-mp 3 -ext tiff)

```
fpdf2img -i c:\input\1.pdf -o d:\output -mp 3 -ext tiff
fpdf2img -i test\2.pdf -o output -mp 3 -ext tiff
fpdf2img -i c:\input -o d:\output -mp 3 -ext tiff
fpdf2img -i test -o output -mp 3 -ext tiff
fpdf2img -i "c:\input\*.pdf" -o d:\output -mp 3 -ext tiff
```

3.2.3.6 Page size setting (-width, -height)

- The optional arguments (**-width**) and (**-height**) are used to set the page width and height for the output image files in pixels.

  Since, 1 points = 1/72 inch, and pixel/DPI = inch,
  So, (points/72)*DPI = pixels.

  Note

  - *If the arguments (-**width**) and (-**height**) are not set, the default image width is calculated by the formula DPI*(PDF_width/72). For example, if the width of the input PDF is 612 points, and the value of DPI is 96 (default), then the default image width is 816 (96*(612/72)) pixels.*
  - *If either of the arguments (-**width**) and (-**height**) is not set, the value will be calculated proportionally based on the other value.*

Usage Example

1) Set the page width and height to 400 and 300 for the output image files (-width 400 -height 300)

```
fpdf2img -i c:\input\1.pdf -o d:\output -width 400 -height 300
fpdf2img -i test\2.pdf -o output -width 400 -height 300
fpdf2img -i c:\input -o d:\output -width 400 -height 300
fpdf2img -i test -o output -width 400 -height 300
```

```
fpdf2img -i "c:\input\*.pdf" -o d:\output -width 400 -height 300
```

3.2.3.7 Specify a clip region of the PDF (-clip)

- The optional argument (**-clip**) is used to specify a clip region of the PDF to be converted. The value of (-**clip**) is a list of four numbers, in points, separated using a space, giving the coordinates of a pair of diagonally opposite corners.

  The usage of this argument is: ***-clip <lower-left_x> <lower-left_y> <upper-right_x> <upper_right_y>***.

  Note *If the third value is set to 0, which is invalid, then the PDF page width will be used as the third value automatically; and if the fourth value is set to 0, the PDF page height will be used as the fourth value automatically.*

Usage Example

1) Set a clip region (*100 120 500 600*) to be converted (-clip 100 120 500 600)

```
fpdf2img -i c:\input\1.pdf -o d:\output -clip 100 120 500 600
fpdf2img -i test\2.pdf -o output -clip 100 120 500 600
fpdf2img -i c:\input -o d:\output -clip 100 120 500 600
fpdf2img -i test -o output -clip 100 120 500 600
fpdf2img -i "c:\input\*.pdf" -o d:\output -clip 100 120 500 600
```

3.2.3.8 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 96. It is valid only for .jpeg, .jpg, .gif, .tiff, .tif, and .png images.

Usage Examples

1) Set DPI to 300 for the output image file(s) (-dpi 300)

```
fpdf2img -i c:\input\1.pdf -o d:\output -ext jgp -dpi 300
fpdf2img -i test\2.pdf -o output -ext png -dpi 300
fpdf2img -i c:\input -o d:\output -ext jpeg -dpi 300
fpdf2img -i test -o output -ext tiff -dpi 300
fpdf2img -i "c:\input\*.pdf" -o d:\output -ext gif -dpi 300
```

3.2.3.9 Color space and color depth (-colorspace/-cs)

- The optional argument (**-colorspace/-cs**) is used to specify the color space and color depth to render the output image files.

  The usage of this argument is: ***-colorspace/-cs <colorspace> [depth]***.

  - <**colorspace**> chooses the color space for the output image file.
  - [**depth**] chooses the color depth for the color space.

  The correspondence between color spaces, color depth and image formats are shown in the following table:

Format	Extension	Color Space	Color depth	Default color space and color depth
JPEG	jpg, jpeg	rgb	24	rgb (24)
		grayscale	8	
PNG	png	rgb	1, 8, 24, 32	rgb (32)
		grayscale	1, 8	
GIF	gif	rgb	8	rgb (8)
		grayscale	8	
BMP	bmp, dib	rgb	1, 8, 24, 32	rgb (32)
		grayscale	1, 8	
TIFF	tiff, tif	rgb	24	rgb (24)
		cmyk	32	
		grayscale	1, 8	
Extended JPEG2000	jpx	rgb	24	rgb (24)

Note *The output images will have better visual effect using the default color space and color depth.*

Usage Example

1) Set the output image format to TIFF, and use CMYK color space and 32-bit to render the output image file(s) (-ext tiff -colorspace cmyk 32 or -ext tiff -cs cmyk 32)

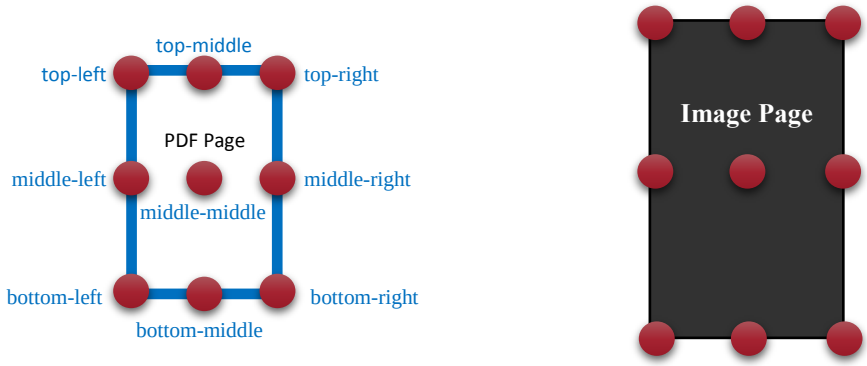
```
fpdf2img -i c:\input\1.pdf -o d:\output -ext tiff -cs cmyk 32
fpdf2img -i test\2.pdf -o output -ext tiff -cs cmyk 32
fpdf2img -i c:\input -o d:\output -ext tiff -cs cmyk 32
fpdf2img -i test -o output -ext jpg -ext tiff -cs cmyk 32
fpdf2img -i "c:\input\*.pdf" -o d:\output -ext tiff -cs cmyk 32
```

3.2.3.10 PDF page alignment (-align)

- The optional argument (**-align**) is used to align positions of the PDF page correspondingly with that of the output image page. The alignment can be set as follows: (the default value is mm or 5)

  - **tl or 1**: the top-left point of the PDF page matches the top-left point of the image page.
  - **tm or 2**: the top-middle point of the PDF page matches the top-middle point of the image page.
  - **tr or 3**: the top-right point of the PDF page matches the top-right point of the image page.
  - **ml or 4**: the middle-left point of the PDF page matches the middle-left point of the image page.
  - **mm or 5**: the middle-middle point of the PDF page matches the middle-middle point of the image page.
  - **mr or 6**: the middle-right point of the PDF page matches the middle-right point of the image page.
  - **bl or 7**: the bottom-left point of the PDF page matches the bottom-left point of the image page.
  - **bm or 8**: the bottom-middle point of the PDF page matches the bottom-middle point of the image page.
  - **br or 9**: the bottom-right point of the PDF page matches the bottom-right point of the image page.

Note *The following picture shows the position of the nine points mentioned above.*



Usage Example

1) Align the top-left point of the PDF page correspondingly with that of the output image page (-align tl or -align 1)

```
fpdf2img -i c:\input\1.pdf -o d:\output -align 1
```

```
fpdf2img -i test\2.pdf -o output -align 1
fpdf2img -i c:\input -o d:\output -align 1
fpdf2img -i test -o output -align 1
fpdf2img -i "c:\input\*.pdf" -o d:\output -align 1
```

3.2.3.11 Page rotation (-rotate)

- The optional argument (**-rotate**) is used to specify the rotation of the PDF page to be rendered to the output image file. The setting value should be 0, 90, 180 or 270, and the default value is 0.

Usage Example

1) Rotate the PDF page to be rendered to the output image file by 90 degrees (-rotate 90)

```
fpdf2img -i c:\input\1.pdf -o d:\output -rotate 90
fpdf2img -i test\2.pdf -o output -rotate 90
fpdf2img -i c:\input -o d:\output -rotate 90
fpdf2img -i test -o output -rotate 90
fpdf2img -i "c:\input\*.pdf" -o d:\output -rotate 90
```

3.2.3.12 Fit the image page (-fitpage)

- The optional argument (**-fitpage**) is used to scale the page of the PDF to fit the page width or height of the image. This argument is useful in combination with page width and height settings.

Usage Example

1) Scale the page of the PDF to fit the page of the image (-fitpage)

```
fpdf2img -i c:\input\1.pdf -o d:\output -width 400 -height 300 -fitpage
fpdf2img -i test\2.pdf -o output -width 400 -height 300 -fitpage
fpdf2img -i c:\input -o d:\output -width 400 -height 300 -fitpage
fpdf2img -i test -o output -width 400 -height 300 -fitpage
fpdf2img -i "c:\input\*.pdf" -o d:\output -width 400 -height 300 -fitpage
```

3.2.3.13 Password for the Input File (-op)

- The optional argument (**-op**) indicates the password for a password-protected input PDF file. It is not required if the input file is not password protected. There are two kinds of passwords, one is user password, and the other is owner password.

User password is also known as "open password", which protects the document against unauthorized opening and reading.

Owner password is also known as "master password", which protects the document against unauthorized editing, printing, changing, and copying. It grants users full access to the document. With the owner password, the document can not only be opened and read, but also be allowed to change the document's security settings. This means that even if the permission of modifying the PDF file was restricted, you can still modify it with the owner password.

Note

- *If the input PDF file is protected by a user password, you need to provide the password using the option "-op" to process the PDF file.*
- *If the input PDF file is protected by an owner password, there are two circumstances. If the permissions of changing the PDF file were not restricted, you can process the PDF file without providing the owner password; otherwise, if the permissions of changing the PDF file were restricted, you need to provide the password using the option "-op" to process the PDF file.*
- *If the input PDF file is protected by a user password and an owner password, there are also two circumstances. If the permissions of changing the PDF file were not restricted, you can provide either of the passwords using the option "-op" to process the PDF file; otherwise, if the permissions of changing the PDF file were restricted, you need to provide the owner password using "-op" option to process the PDF file.*

Usage Example

1) Specify the password for a password-protected input PDF file (-op 123)

```
fpdf2img -i c:\input\1.pdf -o d:\output -op 123
fpdf2img -i test\2.pdf -o output -op 123
```

2) Specify the password for all password-protected input PDF files in a given folder that share the same password (-op welcome)

```
fpdf2img -i c:\input -o d:\output -op welcome
fpdf2img -i test -o output -op welcome
fpdf2img -i "c:\input\*.pdf" -o d:\output -op welcome
```

Note *It only supports typing one value for the argument (**-op**). Only files with the same password can be processed together and files with different password need to be processed separately.*

3.2.3.14 Recursion Depth of Sub-folders (-r)

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

Usage Examples

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

```
fpdf2img -i c:\input -o d:\output -r
fpdf2img -i test -o output -r
fpdf2img -i "c:\input\*.pdf" -o d:\output -r
fpdf2img -i c:\input -o d:\output -r 0
fpdf2img -i test -o output -r 0
fpdf2img -i "c:\input\*.pdf" -o d:\output -r 0
```

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

```
fpdf2img -i c:\input -o d:\output -r 1
fpdf2img -i test -o output -r 1
fpdf2img -i "c:\input\*.pdf" -o d:\output -r 1
```

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

```
fpdf2img -i c:\input -o d:\output
fpdf2img -i test -o output
fpdf2img -i "c:\input\*.pdf" -o d:\output
```

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

```
fpdf2img -i c:\input -o d:\output -r 2
fpdf2img -i test -o output -r 2
fpdf2img -i "c:\input\*.pdf" -o d:\output -r 2
```

3.2.3.15 Multi-thread Support (-t)

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

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

Usage Example

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

```
fpdf2img -i c:\input -o d:\output -t 3
fpdf2img -i test -o output -t 3
fpdf2img -i "c:\input\*.pdf" -o d:\output -t 3
```

3.2.3.16 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 argument (**-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\pdf2img.log" and set the log level to 3 (-log d:\output\pdf2img.log -l 3)

```
fpdf2img -i c:\input -o d:\output -log d:\output\pdf2img.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 the **<licensee>** is the licensee name designated by the users.

Usage Example

1) Register the pdf2image tool with the code "xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx" and the licensee "Foxit" (-register xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx Foixt)

```
fpdf2img -register xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx Foixt
```

c) License agreement (-license)

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

Usage Example

1) Print the license agreement (-license)

```
fpdf2img -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)

```
fpdf2img -version
fpdf2img -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)

```
fpdf2img -help
fpdf2img -h
```

f) Copyright information (-copyright)

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

Usage Example

1) Print the copyright information (-copyright)

```
fpdf2img -copyright
```

3.3 Office2PDF

3.3.1 Basic Syntax

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

foffice2pdf *-register <code> <licensee>*

foffice2pdf *-license*

foffice2pdf *-version/-v*

foffice2pdf *-help/-h*

foffice2pdf *-copyright*

Note:

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

Only the <*-i <srcfile/srcfolder>>* and <*-o <destfile/destfolder>>* 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.

3.3.2 Command Line Summary

Note *For some arguments whose values are strings, users can choose whether to add quotation marks (" ") to the strings. In the manual we have added notes where adding quotation marks (" ") is required.*

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 string can be the name of a single Microsoft Office file (.doc, .docx, .xls, .xlsx, .ppt, .pptx) or a folder. ▪ The file name can contain the wildcard character (*). For example, use *.doc to include all Doc files in a given folder. **Note** The wildcard character (*.*) is currently not supported.

Option	Parameter	Description
-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 folder, (e.g. *-o D:\output*). **Note** The specified output path must already exist.
-pdfa	*[-pdfa]* *e.g.* *-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 (**-b**) is valid only for Microsoft Word files.
-scale	*[-scale<integer>]* *e.g.* *-scale 0* *-scale 1* *-scale 2* *-scale 3*	Specifies a conversion mode for Microsoft Excel files. 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 (**-scale**) is supported only on versions higher than Microsoft Office 2007 and is valid only for Microsoft Excel files.

Option	Parameter	Description
-op	*[-op<string>]*	Specifies the open password for the input file. Not required if the input file is not password protected. **Note** The output PDF file will not retain the open password from the input file.
-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. • **-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** ▪ If no integer value is specified, or if the integer value is 0, then full folders will be searched. By default, the number of layers to recurse is 1, which means that only the current folder will be searched and not sub-folders. ▪ The input Microsoft Office file or folder will be skipped if it is secured and the messages will be displayed.
-t	*[-t <integer>]* *e.g.* *-t 1* *-t 2*	Specifies the number of CPU threads to use. The default value is 1.
-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 messages only concerning out of memory errors. • **-l 2**: logs failure messages concerning the errors caused during execution or those returned from underlying libraries, as well as those for level 1.

Option	Parameter	Description
		• **-l 3**: logs warning messages concerning the PDF files that are overwritten, as well as those for level 2. • **-l 4**: logs informational messages, as well as those for level 3. **Note** The argument (**-l**) is valid only when (**-log**) is used.
-register	*[-register <String> <String>]* *-register <code> <licensee>* *e.g.* *-register xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx Foxit*	Registers the command line tool. ▪ **<code>**: the activation code from Foxit. ▪ **<licensee>**: the Licensee name designated by the users.
-help/-h	*[-help/-h]* *e.g.* *-help* *-h*	Prints the usage information.
-version/-v	*[-version/-v]* *e.g.* *-version* *-v*	Prints the version information.
-license	*[-license]* *e.g.* *-license*	Prints the license agreement.

3.3.3 Basic Usage

3.3.3.1 Input and Output (-i, -o)

a) Input (-i)

- The input should be a single Microsoft Office file or a folder. Users are not able to input multiple Office files or folders, as well as a mixture composed 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 if the input file is in the current working folder. Users can input just the name of the Microsoft Office file or folder, instead of an absolute path. For example:

-i test\2.xls	("test" folder 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 in 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, it is recommended to enclose the input files with quotation marks (" "). In this manual, we add (" ") whenever the input files contain wildcard characters.*

Note *If the input paths or file names contain spaces, you must enclose the input files with quotation marks (" "), such as -i "c:\input file\1.doc", -i "test\user manual.doc". This rule is also used for some other arguments whose values are strings.*

b) Output (-o)

- If the input is a single Microsoft Office file, you should specify the output path of a PDF file. If the input 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. Users can input just 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 into PDF files:

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

```
foffice2pdf -i test -o output
```

3.3.3.2 Bookmark (-b)

- 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 section 3.3.2 "Command Line Summary".

  Note *This argument is valid only for 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
```

3.3.3.3 Scale (-scale)

- The optional argument (**-scale**) is used to specify a conversion mode for Microsoft Excel files. For more details about this argument, please refer to section 3.3.2 "Command Line Summary".

  Note *This argument is supported only on versions higher than Microsoft Office 2007 and is valid only for Microsoft Excel files.*

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
```

3.3.3.4 PDF/A Support (-pdfa)

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

  Note *It 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)

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

3.3.3.5 Open Password (-op)

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

  Note *The output PDF file will not retain the open password from the input file.*

Usage Example

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

```
foffice2pdf -i c:\input\1.doc -o d:\output\1.pdf -op 123
foffice2pdf -i test\2.xls -o output\2.pdf -op 123
```

2) Specify the open password for all input Office files that have been protected with the same password (-op 123)

```
foffice2pdf -i c:\input -o d:\output -op 123
foffice2pdf -i test -o output -op 123
foffice2pdf -i "c:\input\*.doc" -o d:\output -op 123
```

Note *It only supports typing one value for the argument (**-op**). Only files with the same open password can be processed together and files with different open password need to be processed separately.*

3.3.3.6 Recursion Depth of Sub-folders (-r)

- The optional argument (**-r**) indicates the recursion depth of sub-folders, which is valid only when the input is a folder or a path that includes wildcard characters like "c:\input*.doc". By default, the recursion depth is 1, so the sub-folders will not be processed. For more details about this argument, please refer to section 3.3.2 "Command Line Summary".

Usage Examples

1) Search the full 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, the current folder will be searched 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
```

3.3.3.7 Multi-thread Support (-t)

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

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

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
```

3.3.3.8 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 argument (**-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\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 the users.

Usage Example

1) Register the office2pdf tool with the code "xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx" and the licensee "Foxit" (-register xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx Foixt)

   foffice2pdf -register xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx Foixt

c) License agreement (-license)

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

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 -help
foffice2pdf -h

f) Copyright information (-copyright)

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

Usage Example

1) Print the copyright information (-copyright)

foffice2pdf -copyright

3.4 PDFWatermark

3.4.1 Basic Syntax

fpdfwm *<-i <srcfile/srcfolder>> <-o <destfile/destfolder>> <-conf <xmlfile>> [-op <password>]*
[-title <title>] [-subject <subject>] [-keywords <keywords>] [-author <author>] [-creator <creator>]
[-r [recursion]] [-t <threads>] [-log <logfile>] [-l <level>]

fpdfwm *-register <code> <licensee>*

fpdfwm *-license*

fpdfwm *-version/-v*

fpdfwm *-help/-h*

fpdfwm *-copyright*

Note:

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

Only the *<-i <srcfile/srcfolder>>*, *<-o <destfile/destfolder>>* 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 generated by the built-in Foxit Configuration Tool. Full details on each are explained in the following section.

3.4.2 Command Line Summary

Note *For some arguments whose values are strings, users can choose whether to add quotation marks (" ") to the strings. In the manual we have added notes where adding quotation marks (" ") is required.*

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 string can be the name of a single PDF file or a folder. ▪ The file name can contain the wildcard character (*). For example, use *.pdf to include all PDF files in a given folder. **Note** Wildcard character (*.*) is currently not

Option	Parameter	Description
		supported.
-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>>* *e.g.* *-conf c:\watermark.xml*	Specifies the configuration file on the PDFWatermark tool. This file should be generated by the built-in Foxit Configuration Tool.
-op	*[-op<string>]*	Specifies the open password for the input file. Not required if the input file is not password protected. **Note** The output PDF file will retain the open password from the input file.
-title	*[-title <string>]* *e.g.* *-title "Foxit PDF Toolkit User Manual"*	Sets title of PDF files.
-subject	*[-subject <string>]* *e.g.* *-subject "Foxit PDF Toolkit"*	Sets subject of PDF files.
-keywords	*[-keywords <string>]* *e.g.* *-keywords "Foxit"*	Sets keywords of PDF files.
-author	*[-author <string>]* *e.g.* *-author "Jessie"*	Sets author of PDF files.

Option	Parameter	Description
-creator	*[-creator <string>]* *e.g.* *-creator "Foxit PhantomPDF"* *-creator "Foxit Reader"* *-creator "Microsoft® Word 2013"*	Sets creator of PDF files. **Note** It indicates the name of the application that created the source document from which a PDF is generated. For example, if the source document was created from Microsoft® *Word 2013, then you can set the creator to* "Microsoft® *Word 2013".*
-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. • **-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** ▪ If no integer value is specified, or if the integer value is 0, then full folders will be searched. By default, the number of layers to recurse is 1, which means that only the current folder will be searched and not sub-folders. ▪ The input PDF file or folder will be skipped if it is secured and the messages will be displayed.
-t	*[-t <integer>]* *e.g.* *-t 1* *-t 2*	Specifies the number of CPU threads to use. The default value is 1.
-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 messages only concerning out of memory errors. • **-l 2**: logs failure messages concerning the errors caused during execution or those returned from underlying libraries, as well as those for level 1.

Option	Parameter	Description
		• **-l 3**: logs warning messages concerning the PDF files that are overwritten, as well as those for level 2. • **-l 4**: logs informational messages, as well as those for level 3. **Note** The argument (**-l**) is valid only when (**-log**) is used.
-register	*[-register <String> <String>]* *-register <code> <licensee>* *e.g.* *-register xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx Foxit*	Registers the command line tool. ▪ **<code>**: the activation code from Foxit. ▪ **<licensee>**: the Licensee name designated by the users.
-help/-h	*[-help/-h]* *e.g.* *-help* *-h*	Prints the usage information.
-version/-v	*[-version/-v]* *e.g.* *-version* *-v*	Prints the version information.
-license	*[-license]* *e.g.* *-license*	Prints the license agreement.

3.4.3 Basic Usage

3.4.3.1 Required Arguments (-i, -o, -conf)

a) Input (-i)

- The input should be a single PDF file or a folder. Users are not able to input multiple PDF files or folders, as well as a mixture composed 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 if the input file is in the current working folder. Users can input just the name of the PDF file or folder, instead of an absolute path. For example:

-i test\2.pdf	("test" folder 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, it is recommended to enclose the input files with quotation marks (" "). In this manual, we add (" ") whenever the input files contain wildcard characters.*

Note *If the input paths or file names contain spaces, you must enclose the input files with quotation marks (" "), such as -i "c:\input file\1.pdf", -i "test\user manual.pdf". This rule is also used for some other arguments whose values are strings.*

b) Output (-o)

- If the input is a single PDF file, you should specify the output path of a PDF file. If the input 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. Users can input just 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 built-in Foxit Configuration Tool (*fpdfwmconf.exe* or *fpdfwmconf64.exe*). For more details about watermark settings, please refer to section 4.1.1 "Watermark Settings". Users 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. Users can input just 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 a watermark into PDF files:

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

3.4.3.2 Password for the Input File (-op)

- The optional argument (**-op**) indicates the password for a password-protected input PDF file. It is not required if the input file is not password protected. There are two kinds of passwords, one is user password, and the other is owner password.

  User password is also known as "open password", which protects the document against unauthorized opening and reading.

  Owner password is also known as "master password", which protects the document against unauthorized editing, printing, changing, and copying. It grants users full access to the document. With the owner password, the document can not only be opened and read, but also be allowed to change the document's security settings. This means that even if the permission of modifying the PDF file was restricted, you can still modify it with the owner password.

  Note

  - *The output PDF file will retain the password from the input file.*
  - *If the input PDF file is protected by a user password, you need to provide the password using the option "-op" to process the PDF file.*
  - *If the input PDF file is protected by an owner password, there are two circumstances. If the permissions of changing the PDF file were not restricted, you can process the PDF file without providing the owner password; otherwise, if the permissions of changing the PDF file were restricted, you need to provide the password using the option "-op" to process the PDF file.*
  - *If the input PDF file is protected by a user password and an owner password, there are also two circumstances. If the permissions of changing the PDF file were not restricted, you can*

provide either of the passwords using the option "-op" to process the PDF file; otherwise, if the permissions of changing the PDF file were restricted, you need to provide the owner password using "-op" option to process the PDF file.

Usage Example

1) Specify the password for a password-protected 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 password for all password-protected input PDF files in a given folder that share the same password (-op welcome)

```
fpdfwm -i test -o output -conf conf_wm.xlm -op welcome
fpdfwm -i c:\input -o d:\output -conf c:\conf_wm.xml -op welcome
fpdfwm -i "c:\input\*.pdf -o d:\output -conf c:\conf_wm.xml -op welcome
```

Note *It only supports typing one value for the argument (**-op**). Only files with the same password can be processed together and files with different password need to be processed separately.*

3.4.3.3 Document Metadata Settings (-title, -subject, -keywords, -author, -creator)

a) Title (-title)

- The optional argument (**-title**) is used to set title of PDF files.

Usage Example

1) Set document title to "Foxit PDF Toolkit User Manual" (-title "Foxit PDF Toolkit User Manual")

```
fpdfwm -i test -o output -conf conf_wm.xml -title "Foxit PDF Toolkit User Manual"
fpdfwm -i "c:\input\*.pdf" -o d:\output -conf c:\conf_wm.xml -title "Foxit PDF Toolkit User Manual"
fpdfwm -i c:\input\1.pdf -o d:\output\1_wm.pdf -conf c:\conf_wm.xml -title "Foxit PDF Toolkit User
Manual"
```

b) Subject (-subject)

- The optional argument (**-subject**) is used to set subject of PDF files.

Usage Example

1) Set document subject to "Foxit PDF Toolkit" (-subject "Foxit PDF Toolkit")

```
fpdfwm -i test -o output -conf conf_wm.xml -subject "Foxit PDF Toolkit"
fpdfwm -i "c:\input\*.pdf" -o d:\output -conf c:\conf_wm.xml -subject "Foxit PDF Toolkit"
fpdfwm -i c:\input\1.pdf -o d:\output\1_wm.pdf -conf c:\conf_wm.xml -subject "Foxit PDF Toolkit"
```

c) Keywords (-keywords)

- The optional argument (**-keywords**) is used to set keywords of PDF files.

Usage Example

1) Set document keywords to "toolkit" (-keywords "toolkit")

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

d) Author (-author)

- The optional argument (**-author**) is used to set author of PDF files.

Usage Example

1) Set document author to "Jessie" (-author "Jessie")

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

e) Creator (-creator)

- The optional argument (**-creator**) is used to set file creation application information of PDF files.

Usage Example

1) Set document creator to "Foxit Reader" (-creator "Foxit Reader")

```
fpdfwm -i test -o output -conf conf_wm.xml -creator "Foxit Reader"
```

```
fpdfwm -i "c:\input\*.pdf" -o d:\output -conf c:\conf_wm.xml -creator "Foxit Reader"
fpdfwm -i c:\input\1.pdf -o d:\output\1_wm.pdf -conf c:\conf_wm.xml -creator "Foxit Reader"
```

3.4.3.4 Recursion Depth of Sub-folders (-r)

- The optional argument (**-r**) indicates the recursion depth of sub-folders, which is valid only when the input is a folder or a path that includes wildcard characters like "c:\input*.pdf". By default, the recursion depth is 1, so the sub-folders will not be processed. For more details about this argument, please refer to section 3.4.2 "Command Line Summary".

Usage Examples

1) Search the full 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, the current folder will be searched 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
```

3.4.3.5 Multi-thread Support (-t)

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

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

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
```

3.4.3.6 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 argument (**-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\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 the users.

Usage Example

1) Register the pdfwatermark tool with the code "xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx" and the licensee "Foxit" (-register xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx Foixt)

```
fpdfwm -register xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx Foixt
```

c) License agreement (-license)

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

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
```

f) Copyright information (-copyright)

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

Usage Example

2) Print the copyright information (-copyright)

```
fpdfwm -copyright
```

3.5 PDFHeaderFooter

3.5.1 Basic Syntax

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

Note:

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

Only the *<-i <srcfile/srcfolder>>*, *<-o <destfile/destfolder>>* 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 generated by the built-in Foxit Configuration Tool. Full details on each are explained in the following section.

3.5.2 Command Line Summary

Note *For some arguments whose values are strings, users can choose whether to add quotation marks (" ") to the strings. In the manual we have added notes where adding quotation marks (" ") is required.*

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 string can be the name of a single PDF file or a folder. ▪ The file name can contain the wildcard character (*). For example, use *.pdf to include all PDF files in a given folder.

Option	Parameter	Description
		Note Wildcard character (*.*) is currently not supported.
-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>>* *e.g.* *-conf c:\input\hf.xml*	Specifies the configuration file on the PDFHeaderFooter tool. This file should be generated by the built-in Foxit Configuration Tool. **Note** This argument (**-conf**) will only be used if (**-mode**) is set to 1 or 2.
-overlay	*[-overlay]* *e.g.* *-overlay*	Overlays an existing header/footer. **Note** This argument (**-overlay**) is valid only when (**-mode**) is set to 1.
-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 password protected. **Note** The output PDF file will retain the open password from the input file.

Option	Parameter	Description
-title	*[-title <string>]* *e.g.* *-title "Foxit PDF Toolkit User Manual"*	Sets title of PDF files.
-subject	*[-subject <string>]* *e.g.* *-subject "Foxit PDF Toolkit"*	Sets subject of PDF files.
-keywords	*[-keywords <string>]* *e.g.* *-keywords "Foxit"*	Sets keywords of PDF files.
-author	*[-author <string>]* *e.g.* *-author "Jessie"*	Sets author of PDF files.
-creator	*[-creator <string>]* *e.g.* *-creator "Foxit PhantomPDF"* *-creator "Foxit Reader"* *-creator "Microsoft® Word 2013"*	Sets creator of PDF files. **Note** It indicates the name of the application that created the source document from which a PDF is generated. For example, if the source document was created from Microsoft® *Word 2013, then you can set the creator to* "Microsoft® *Word 2013".*
-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. • **-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** ▪ If no integer value is specified, or if the integer value is 0, then full folders will be searched. By default, the number of layers to recurse is 1, which means that only the current folder will be searched and not sub-folders. ▪ The input PDF file or folder will be skipped if it is secured and the messages will be displayed.

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

3.5.3 Basic Usage

3.5.3.1 Required Arguments (-i, -o, -mode, -conf)

a) Input (-i)

- The input should be a single PDF file or a folder. Users are not able to input multiple PDF files or folders, as well as a mixture composed 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 if the input file is in the current working folder. Users can input just the name of the PDF file or folder, instead of an absolute path. For example:

  -i test\2.pdf ("test" folder 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, it is recommended to enclose the input files with quotation marks (" "). In this manual, we add (" ") whenever the input files contain wildcard characters.*

Note *If the input paths or file names contain spaces, you must enclose the input files with quotation marks (" "), such as -i "c:\input file\1.pdf", -i "test\user manual.pdf". This rule is also used for some other arguments whose values are strings.*

b) Output (-o)

- If the input is a single PDF file, you should specify the output path of a PDF file. If the input 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. Users can input just 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 3.5.2 "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 built-in Foxit Configuration Tool (*fpdfhfconf.exe* or *fpdfhfconf64.exe*). For more details about header/footer settings, please refer to section 4.2.1 "Header/Footer Settings". Users 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. Users can input just 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 "c:\input\*.pdf" -o d:\output -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 "c:\input\*.pdf" -o d:\output -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 "c:\input\*.pdf" -o d:\output -mode 3
fpdfhf -i test -o output -mode 3
```

3.5.3.2 Overlay (-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
```

3.5.3.3 Password for the Input File (-op)

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

  User password is also known as "open password", which protects the document against unauthorized opening and reading.

  Owner password is also known as "master password", which protects the document against unauthorized editing, printing, changing, and copying. It grants users full access to the document. With the owner password, the document can not only be opened and read, but also be allowed to change the document's security settings. This means that even if the permission of modifying the PDF file was restricted, you can still modify it with the owner password.

  Note

  - *The output PDF file will retain the password from the input file.*
  - *If the input PDF file is protected by a user password, you need to provide the password using the option "-op" to process the PDF file.*

- *If the input PDF file is protected by an owner password, there are two circumstances. If the permissions of changing the PDF file were not restricted, you can process the PDF file without providing the owner password; otherwise, if the permissions of changing the PDF file were restricted, you need to provide the password using the option "-op" to process the PDF file.*
- *If the input PDF file is protected by a user password and an owner password, there are also two circumstances. If the permissions of changing the PDF file were not restricted, you can provide either of the passwords using the option "-op" to process the PDF file; otherwise, if the permissions of changing the PDF file were restricted, you need to provide the owner password using "-op" option to process the PDF file.*

Usage Example

1) Specify the password for a password-protected 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 password for all password-protected input PDF files in a given folder that share the same password (-op welcome)

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

Note *It only supports typing one value for the argument (**-op**). Only files with the same password can be processed together and files with different password need to be processed separately.*

3.5.3.4 Document Metadata Settings (-title, -subject, -keywords, -author, -creator)

a) Title (-title)

- The optional argument (**-title**) is used to set title of PDF files.

Usage Example

1) Set document title to "Foxit PDF Toolkit User Manual" (-title "Foxit PDF Toolkit User Manual")

```
fpdfhf -i c:\input\1.pdf -o d:\output\1_hf.pdf -mode 3 -title "Foxit PDF Toolkit User Manual"
```

```
fpdfhf -i test -o output -mode 2 -conf conf_hf.xml -title "Foxit PDF Toolkit User Manual"
fpdfhf -i "c:\input\*.pdf" -o d:\output -mode 1 -conf c:\conf_hf.xml -title "Foxit PDF Toolkit User Manual"
```

b) Subject (-subject)

- The optional argument (**-subject**) is used to set subject of PDF files.

Usage Example

1) Set document subject to "Foxit PDF Toolkit" (-subject "Foxit PDF Toolkit")

```
fpdfhf -i c:\input\1.pdf -o d:\output\1_hf.pdf -mode 3 -subject "Foxit PDF Toolkit"
fpdfhf -i test -o output -mode 2 -conf conf_hf.xml -subject "Foxit PDF Toolkit"
fpdfhf -i "c:\input\*.pdf" -o d:\output -mode 1 -conf c:\conf_hf.xml -subject "Foxit PDF Toolkit"
```

c) Keywords (-keywords)

- The optional argument (**-keywords**) is used to set keywords of PDF files.

Usage Example

1) Set document keywords to "toolkit" (-keywords "toolkit")

```
fpdfhf -i c:\input\1.pdf -o d:\output\1_hf.pdf -mode 3 -keywords "toolkit"
fpdfhf -i test -o output -mode 2 -conf conf_hf.xml -keywords "toolkit"
fpdfhf -i "c:\input\*.pdf" -o d:\output -mode 1 -conf c:\conf_hf.xml -keywords "toolkit"
```

d) Author (-author)

- The optional argument (**-author**) is used to set author of PDF files.

Usage Example

1) Set document author to "Jessie" (-author "Jessie")

```
fpdfhf -i c:\input\1.pdf -o d:\output\1_hf.pdf -mode 3 -author "Jessie"
fpdfhf -i test -o output -mode 2 -conf conf_hf.xml -author "Jessie"
fpdfhf -i "c:\input\*.pdf" -o d:\output -mode 1 -conf c:\conf_hf.xml -author "Jessie"
```

e) Creator (-creator)

- The optional argument (**-creator**) is used to set file creation application information of PDF files.

Usage Example

1) Set document creator to "Foxit Reader" (-creator "Foxit Reader")

```
fpdfhf -i c:\input\1.pdf -o d:\output\1_hf.pdf -mode 3 -creator "Foxit Reader"
fpdfhf -i test -o output -mode 2 -conf conf_hf.xml -creator "Foxit Reader"
fpdfhf -i "c:\input\*.pdf" -o d:\output -mode 1 -conf c:\conf_hf.xml -creator "Foxit Reader"
```

3.5.3.5 Recursion Depth of Sub-folders (-r)

- The optional argument (**-r**) indicates the recursion depth of sub-folders, which is valid only when the input is a folder or a path that includes wildcard characters like "c:\input*.pdf". By default, the recursion depth is 1, so the sub-folders will not be processed. For more details about this argument, please refer to section 3.5.2 "Command Line Summary".

Usage Examples

1) Search the full 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, the current folder will be searched 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
```

3.5.3.6 Multi-thread Support (-t)

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

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

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
```

3.5.3.7 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 argument (**-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\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 the users.

Usage Example

1) Register the pdfheaderfooter tool with the code "xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx" and the licensee "Foxit" (-register xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx Foixt)

```
fpdfhf -register xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx Foixt
```

c) License agreement (-license)

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

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
```

f) Copyright information (-copyright)

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

Usage Example

3) Print the copyright information (-copyright)

```
fpdfhf -copyright
```

3.6 PDFOptimizer

3.6.1 Basic Syntax

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

Note:

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

Only the *<-i <srcfile/srcfolder>>* and *<-o <destfile/destfolder>>* 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.

3.6.2 Command Line Summary

Note *For some arguments whose values are strings, users can choose whether to add quotation marks (" ") to the strings. In the manual we have added notes where adding quotation marks (" ") is required.*

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 string can be the name of a single PDF file or a folder. ▪ The file name can contain the wildcard character (*). For example, use *.pdf to include all PDF files in a given folder. **Note** Wildcard character (*.*) is currently not supported.

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.
-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** should 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*	Compresses color/grayscale images. ▪ <**algorithm**>: Chooses compression algorithm. **j**: JPEG **j2**: JPEG2000 **h**: High Compression ▪ <**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** ▪ High Compression algorithm only supports PDF

Option	Parameter	Description
	-cc j2 high 256 *-cc j2 max 256* *-cc h min* *-cc h low* *-cc h medium* *-cc h high* *-cc h max*	version 1.5 or higher. ▪ If the argument (**-cc**) is not set, images will not be compressed.
-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** should 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* *-cm highcompression*	Compresses monochrome images. ▪ <**algorithm**>: Chooses compression algorithm. **jbig2**: JBIG2 **ccitt**: CCITT Group 4 **runlength**: RUN LENGTH **highcompression**: High Compression ▪ [**level**]: Chooses quality level for JBIG2 algorithm only. Allowable levels are lossless and lossy. **Note** ▪ High Compression algorithm only supports PDF version 1.5 or higher. ▪ If the argument (**-cm**) is not set, images will not be

Option	Parameter	Description
		compressed.
-rd	*[-rd]* *e.g. -rd*	If this flag is set, images will be optimized only if there is a reduction in size.
-u	*[-u]* *e.g.* *-u*	Unembeds all fonts in selected PDF document(s).
-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 password protected. **Note** The output PDF file will retain the open password from the input file.

Option	Parameter	Description
-title	*[-title <string>]* *e.g.* *-title "Foxit PDF Toolkit User Manual"*	Sets title of PDF files.
-subject	*[-subject <string>]* *e.g.* *-subject "Foxit PDF Toolkit"*	Sets subject of PDF files.
-keywords	*[-keywords <string>]* *e.g.* *-keywords "Foxit"*	Sets keywords of PDF files.
-author	*[-author <string>]* *e.g.* *-author "Jessie"*	Sets author of PDF files.
-creator	*[-creator <string>]* *e.g.* *-creator "Foxit PhantomPDF"* *-creator "Foxit Reader"* *-creator "Microsoft® Word 2013"*	Sets creator of PDF files. **Note** It indicates the name of the application that created the source document from which a PDF is generated. For example, if the source document was created from Microsoft® *Word 2013, then you can set the creator to* "Microsoft® *Word 2013".*
-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. • **-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** ▪ If no integer value is specified, or if the integer value is 0, then full folders will be searched. By default, the number of layers to recurse is 1, which means that only the current folder will be searched and not sub-folders. ▪ The input PDF file or folder will be skipped if it is

Option	Parameter	Description
		secured and the messages will be displayed.
-t	*[-t <integer>]* *e.g.* *-t 1* *-t 2*	Specifies the number of CPU threads to use. The default value is 1.
-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 messages only concerning out of memory errors. • **-l 2**: logs failure messages concerning the errors caused during execution or those returned from underlying libraries, as well as those for level 1. • **-l 3**: logs warning messages concerning the PDF files that are overwritten, as well as those for level 2. • **-l 4**: logs informational messages, as well as those for level 3. **Note** The argument (**-l**) is valid only when (**-log**) is used.
-register	*[-register <string> <string>]* *-register <code> <licensee>* *e.g.* *-register xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx Foxit*	Registers the command line tool. ▪ **<code>**: the activation code from Foxit. ▪ **<licensee>**: the Licensee name designated by the users.
-help/-h	*[-help/-h]* *e.g.* *-help* *-h*	Prints the usage information.
-version/-v	*[-version/-v]* *e.g.* *-version* *-v*	Prints the version information.
-license	*[-license]* *e.g. -license*	Prints the license agreement.

3.6.3 Basic Usage

3.6.3.1 Input and Output (-i, -o)

a) Input (-i)

- The input should be a single PDF file or a folder. Users are not able to input multiple PDF files or folders, as well as a mixture composed 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 if the input file is in the current working folder. Users can input just the name of the PDF file or folder, instead of an absolute path. For example:

  -i test\2.pdf ("test" folder 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, it is recommended to enclose the input files with quotation marks (" "). In this manual, we add (" ") whenever the input files contain wildcard characters.*

Note *If the input paths or file names contain spaces, you must enclose the input files with quotation marks (" "), such as -i "c:\input file\1.pdf", -i "test\user manual.pdf". This rule is also used for some other arguments whose values are strings.*

b) Output (-o)

- If the input is a single PDF file, you should specify the output path of a PDF file. If the input 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. Users can input just 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 a single PDF file:

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

2) Optimize PDF files in a folder:

   fpdfopt -i test -o output
   fpdfopt -i c:\input -o d:\output
   fpdfopt -i "c:\input*.pdf" -o d:\output

3.6.3.2 PDF/Page Settings (-dc, -cc, -dm, -cm, -rd, -u, -d, -cl)

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 3.6.2 "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 the 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 the 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, JPEG2000 or High Compression algorithm. For more details about this argument, please refer to section 3.6.2 "Command Line Summary".

Usage Example

1) Compress color/grayscale images with medium quality JEPG (-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 maximum quality JEPG2000 (-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
```

3) Compress color/grayscale images with high quality High Compression (-cc h high)

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

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 3.6.2 "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 the 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 the 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, Run Length or High Compression algorithm. For more details about this argument, please refer to section 3.6.2 "Command Line Summary".

Usage Example

1) Compress monochrome images with lossless quality JBIG2 (-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
```

3) Compress monochrome images with High Compression (-cm highcompression)

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

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 argument (**-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 3.6.2 "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 section 3.6.2 "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"
```

3.6.3.3 Password for the Input File (-op)

- The optional argument (**-op**) indicates the password for a password-protected input PDF file. It is not required if the input file is not password protected. There are two kinds of passwords, one is user password, and the other is owner password.

  User password is also known as "open password", which protects the document against unauthorized opening and reading.

  Owner password is also known as "master password", which protects the document against unauthorized editing, printing, changing, and copying. It grants users full access to the document. With the owner password, the document can not only be opened and read, but also be allowed to change the document's security settings. This means that even if the permission of modifying the PDF file was restricted, you can still modify it with the owner password.

  Note

  - *The output PDF file will retain the password from the input file.*
  - *If the input PDF file is protected by a user password, you need to provide the password using the option "-op" to process the PDF file.*
  - *If the input PDF file is protected by an owner password, there are two circumstances. If the permissions of changing the PDF file were not restricted, you can process the PDF file without providing the owner password; otherwise, if the permissions of changing the PDF file were restricted, you need to provide the password using the option "-op" to process the PDF file.*
  - *If the input PDF file is protected by a user password and an owner password, there are also two circumstances. If the permissions of changing the PDF file were not restricted, you can provide either of the passwords using the option "-op" to process the PDF file; otherwise, if*

the permissions of changing the PDF file were restricted, you need to provide the owner password using "-op" option to process the PDF file.

Usage Example

1) Specify the password for a password-protected 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 password for all password-protected input PDF files in a given folder that share the same password (-op welcome)

```
fpdfopt -i c:\input -o d:\output -op welcome
fpdfopt -i test -o output -op welcome
fpdfopt -i "c:\input\*.pdf" -o d:\output -op welcome
```

Note *It only supports typing one value for the argument (**-op**). Only files with the same password can be processed together and files with different password need to be processed separately.*

3.6.3.4 Document Metadata Settings (-title, -subject, -keywords, -author, -creator)

a) Title (-title)

- The optional argument (**-title**) is used to set title of PDF files.

Usage Example

1) Set document title to "Foxit PDF Toolkit User Manual" (-title "Foxit PDF Toolkit User Manual")

```
fpdfopt -i c:\input\1.pdf -o d:\output\1_opt.pdf -title "Foxit PDF Toolkit User Manual"
fpdfopt -i test -o output -title "Foxit PDF Toolkit User Manual"
fpdfopt -i "c:\input\*.pdf" -o d:\output -title "Foxit PDF Toolkit User Manual"
```

b) Subject (-subject)

- The optional argument (**-subject**) is used to set subject of PDF files.

Usage Example

1) Set document subject to "Foxit PDF Toolkit" (-subject "Foxit PDF Toolkit")

```
fpdfopt -i c:\input\1.pdf -o d:\output\1_opt.pdf -subject "Foxit PDF Toolkit"
fpdfopt -i test -o output -subject "Foxit PDF Toolkit"
fpdfopt -i "c:\input\*.pdf" -o d:\output -subject "Foxit PDF Toolkit"
```

c) Keywords (-keywords)

- The optional argument (**-keywords**) is used to set keywords of PDF files.

Usage Example

1) Set document keywords to "toolkit" (-keywords "toolkit")

```
fpdfopt -i c:\input\1.pdf -o d:\output\1_opt.pdf -keywords "toolkit"
fpdfopt -i test -o output -keywords "toolkit"
fpdfopt -i "c:\input\*.pdf" -o d:\output -keywords "toolkit"
```

d) Author (-author)

- The optional argument (**-author**) is used to set author of PDF files.

Usage Example

1) Set document author to "Jessie" (-author "Jessie")

```
fpdfopt -i c:\input\1.pdf -o d:\output\1_opt.pdf -author "Jessie"
fpdfopt -i test -o output -author "Jessie"
fpdfopt -i "c:\input\*.pdf" -o d:\output -author "Jessie"
```

e) Creator (-creator)

- The optional argument (**-creator**) is used to set file creation application information of PDF files.

Usage Example

1) Set document creator to "Foxit Reader" (-creator "Foxit Reader")

```
fpdfopt -i c:\input\1.pdf -o d:\output\1_opt.pdf -creator "Foxit Reader"
fpdfopt -i test -o output -creator "Foxit Reader"
fpdfopt -i "c:\input\*.pdf" -o d:\output -creator "Foxit Reader"
```

3.6.3.5 Recursion Depth of Sub-folders (-r)

- The optional argument (**-r**) indicates the recursion depth of sub-folders, which is valid only when the input is a folder or a path that includes wildcard characters like "c:\input*.pdf". By default, the recursion depth is 1, so the sub-folders will not be processed. For more details about this argument, please refer to section 3.6.2 "Command Line Summary".

Usage Examples

1) Search the full 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, the current folder will be searched 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
```

3.6.3.6 Multi-thread Support (-t)

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

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

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
```

3.6.3.7 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 argument (**-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 3.6.2 "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 the users.

Usage Example

1) Register the pdfoptimizer tool with the code "xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx" and the licensee "Foxit" (-register xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx Foixt)

```
fpdfopt -register xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx Foixt
```

c) License agreement (-license)

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

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
```

f) Copyright information (-copyright)

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

Usage Example

4) Print the copyright information (-copyright)

```
fpdfopt -copyright
```

3.7 PDFRedactor

3.7.1 Basic Syntax

fpdfredact *<-i <srcfile/srcfolder>> <-o <destfile/destfolder>> <-mode <operation mode>>*
<-keyword <searchword>/<<pattern> <country>>>
[-character <singlepos>/<<startpos> <endpos>>] [-range <page range>] [-markcolor <R> <G> <B>]
[-tx <text>] [-ta <alignment>] [-font <fontname>] [-fs <fontsize>] [-fontcolor <R> <G> <B>] [-form]
[-op <password>] [-title <title>] [-subject <subject>] [-keywords <keywords>] [-author <author>]
[-creator <creator>] [-r [recursion]] [-t <thread>] [-log <logfile>] [-l <level>]
fpdfredact *-register <code> <licensee>*
fpdfredact *-license*
fpdfredact *-version/-v*
fpdfredact *-help/-h*
fpdfredact *-copyright*

Note:

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

Only the *<-i <srcfile/srcfolder>>*, *<-o <destfile/destfolder>>*, *<-mode <operation mode>>* and *<-keyword <searchword>/<<pattern> <country>>>* 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. Full details on each are explained in the following section.

3.7.2 Command Line Summary

Note *For some arguments whose values are strings, users can choose whether to add quotation marks (" "). In the manual we have added notes where adding quotation marks (" ") is required.*

Option	Parameter	Description
-i	*<-i <string>>* *e.g.* *-i c:\input\1.pdf* *-i c:\input* *-i "c:\input*.pdf"*	Specifies the input file to be redacted. ▪ The input string can be the name of a single PDF file or a folder. ▪ The file name can contain the wildcard

Option	Parameter	Description
		character (*). For example, use *.pdf to include all PDF files in a given folder. **Note** Wildcard character (*.*) is currently not supported.
-o	*<-o <string>>* *e.g.* *-o d:\output\1_reda.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_reda.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*	Specifies the mode of the keyword to be searched. • **-mode 1**: sets the mode of the keyword to a word or phrase. • **-mode 2**: sets the mode of the keyword to a pattern.
-keyword	*<-keyword <string>/<<integer> <integer>>>* *<-keyword <word/phrase> / <<pattern> <country>>>* *e.g.* *-keyword "Foxit"* *-keyword "Foxit PDF Toolkit"* *-keyword 1 1* *-keyword 1 2* *-keyword 2 3* *-keyword 3 4* *-keyword 4 5* *-keyword 5 8* *-keyword 5 10* *…*	Specifies the search keyword. There are two types of keyword and the "**-mode**" argument determines which one to use. ▪ **-keyword <word/phrase>** is used when **-mode 1** is set. And **<word/phrase>** is a string with quotation marks. ▪ **-keyword <pattern> <country>** is used when **-mode 2** is set. **<pattern>** value could be: **1:** Phone Number **2:** Date **3:** Social Security Number **4:** Email Address **5:** Credit Card Number **<country>** value could be: **1:** EN_US **2:** China **3:** China_TW&HK **4:** French

Option	Parameter	Description
		5: German **6:** Japan **7:** Korean **8:** Brazil **9:** Spain **10:** Turkey **Note** If the keyword contains special characters like quotation marks and slashes, you should add the escape character (\) before them.
-character	*[-character <integer>/<<integer> <integer>>]* *-character <singlepos>/ <<startpos> <endpos>>* *e.g.* *-character 3* *-character 1 4*	Specifies the characters of the keyword you want to redact. By default, all the characters of the keyword will be redacted. ▪ **-character <singlepos>**: redacts only the ***singlepos*** character of the keyword. For example, *-character 3*: redacts only the third character of the keyword. ▪ **-character <startpos> <endpos>**: redacts the characters of the keyword in the range from ***startpos*** to ***endpos***. For example, *-character 1 4*: redacts the first character to the fourth character of the keyword.
-range	*-rang <String>* *e.g.* *-range "1,5,9"* *-range "all"* *-range "even"* *-range "odd"* *-range "2-10,30"* *-range "10,50-"* *-range "odd,100-"*	Specifies a page range to apply redaction. By default, all of the pages will apply redaction. ▪ Applies redaction to pages 1,5, and 9: **-range "1,5,9"** ▪ Applies redaction to all of the pages: **-range "all"** ▪ Applies redaction to all even pages: **-range "even"** ▪ Applies redaction to all odd pages: **-range "odd"** ▪ Applies redaction to pages in the range from 2-10 and page 30:

Option	Parameter	Description
		-range "2-10,30" ▪ Applies redaction to page 10 and all pages in the range from 50 to the last page: **-range "10,50-"** ▪ Applies redaction to all odd pages and all pages in the range from 100 to the last page: **-range "odd,100-"**
-markcolor	*[-markcolor <integer> <integer> <integer>]* *-markcolor <R> <G> <B>* *e.g.* *-markcolor 0 0 0* *-markcolor 255 255 0* *-markcolor 255 255 255* *…*	Specifies the fill color used to mark for redaction with the RGB color model. By default, the fill color is black. Allowable range of the values for each RGB component is 0-255.
-tx	*[-tx <string>]* *-tx <text>* *e.g.* *-tx "secret"*	Overlays redaction marks with the specified text. **Note** If the keyword contains special characters like quotation marks and slashes, you should add the escape character (\) before them.
-ta	*[-ta <integer>]* *-ta <alignment>* *e.g.* *-ta 1* *-ta 2* *-ta 3*	Sets the alignment of the text designated by the **-tx** option. The default value is 1. • **-ta 1**: aligns the text left. • **-ta 2**: aligns the text center. • **-ta 3**: aligns the text right. **Note** The (**-ta**) is valid only when (**-tx**) is used.
-font	*[-font <string>]* *-font <fontname>* *e.g.* *-font "Calibri"* *-font "Helvetica"* *-font "Arial"* *…*	Sets the font style of the text designated by the **-tx** option. **Note** ▪ The font style you set should be installed on a local environment, otherwise the default font style will be used. ▪ The (**-font**) is valid only when (**-tx**) is used.

Option	Parameter	Description
-fs	*[-fs <integer>]* *-fs <fontsize>* *e.g.* *-fs 8* *-fs 11* *-fs 72*	Sets the font size of the text designated by the **-tx** option. Default value: 9. Allowable range: 8-72. **Note** The (**-fs**) is valid only when (**-tx**) is used.
-fontcolor	*[-fontcolor <integer> <integer> <integer>]* *-fontcolor <R> <G> <B>* *e.g.* *-fontcolor 0 0 0* *-fontcolor 255 255 0* *-fontcolor 255 255 255* …	Specifies the font color of the text designated by the **-tx** option with the RGB color model. By default, the font color is green. Allowable range of the values for each RGB component is 0-255. **Note** The (**-fontcolor**) is valid only when (**-tx**) is used.
-form	*[-form]*	If this argument is set, both PDF forms and main body of text will be searched for the specified keyword, otherwise only main body of text will be searched. **Note** If the keyword in the PDF form is generated by JavaScript, it will not be searched.
-op	*[-op<string>]*	Specifies the open password for the input file. Not required if the input file is not password protected. **Note** The output PDF file will retain the open password from the input file.
-title	*[-title <string>]* *e.g.* *-title "Foxit PDF Toolkit User Manual"*	Sets title of PDF files.
-subject	*[-subject <string>]* *e.g.* *-subject "Foxit PDF Toolkit"*	Sets subject of PDF files.
-keywords	*[-keywords <string>]* *e.g.* *-keywords "Foxit"*	Sets keywords of PDF files.

Option	Parameter	Description
-author	*[-author <string>]* *e.g.* *-author "Jessie"*	Sets author of PDF files.
-creator	*[-creator <string>]* *e.g.* *-creator "Foxit PhantomPDF"* *-creator "Foxit Reader"* *-creator "Microsoft® Word 2013"*	Sets creator of PDF files. **Note** It indicates the name of the application that created the source document from which a PDF is generated. For example, if the source document was created from Microsoft® *Word 2013, then you can set the creator to* "Microsoft® *Word 2013".*
-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. • **-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** ▪ If no integer value is specified, or if the integer value is 0, then full folders will be searched. By default, the number of layers to recurse is 1, which means that only the current folder will be searched and not sub-folders. ▪ The input PDF file or folder will be skipped if it is secured and the messages will be displayed.
-t	*[-t <integer>]* *e.g.* *-t 1* *-t 2*	Specifies the number of CPU threads to use. The default value is 1.
-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.*	Sets the log level. The default is 4. • **-l 1**: logs messages only concerning out of

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

3.7.3 Basic Usage

3.7.3.1 Required Arguments (-i, -o, -mode, -keyword)

a) Input (-i)

➢ The input should be a single PDF file or a folder. Users are not able to input multiple PDF files or folders, as well as a mixture composed 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 if the input file is in the current working folder. Users can input just the name of the PDF file or folder, instead of an absolute path. For example:

-i test\2.pdf ("test" folder 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, it is recommended to enclose the input files with quotation marks (" "). In this manual, we add (" ") whenever the input files contain wildcard characters.*

Note *If the input paths or file names contain spaces, you must enclose the input files with quotation marks (" "), such as -i "c:\input file\1.pdf", -i "test\user manual.pdf". This rule is also used for some other arguments whose values are strings.*

b) Output (-o)

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

-o d:\output\1_reda.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. Users can input just the name of the output PDF file or folder, instead of an absolute path. For example:

-o output\2_reda.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 for specifying the mode of the keyword to be searched. There are two types of keywords: one is a word or phrase, and the other is a pattern. For example,

  -mode 1 (keyword is a word or phrase)
  -mode 2 (keyword is a pattern)

d) Search Keyword (-keyword)

- The search keyword argument (**-keyword**) is required in the command line and its value is dependent on the setting of the argument (**-mode**). For more details about this argument, please refer to section 3.7.2 "Command Line Summary". The **-mode** and **-keyword** arguments are grouped together. For example,

  -mode 1 -keyword "Foxit"
  -mode 1 -keyword "Foxit Software"
  -mode 2 -keyword 1 2
  -mode 2 -keyword 2 8

Usage Examples

1) Redact(Remove) the sensitive content "Foxit" from PDF file(s):

```
fpdfredact -i c:\input\1.pdf -o d:\output\1_reda.pdf -mode 1 -keyword "Foxit"
fpdfredact -i test\2.pdf -o output\2_reda.pdf -mode 1 -keyword "Foxit"
fpdfredact -i c:\input -o d:\output -mode 1 -keyword "Foxit"
fpdfredact -i test -o output -mode 1 -keyword "Foxit"
fpdfredact -i "c:\input\*.pdf" -o d:\output -mode 1 -keyword "Foxit"
```

2) Redact(Remove) the sensitive content "Phone Number" pattern in "En_US" country from PDF file(s):

```
fpdfredact -i c:\input\1.pdf -o d:\output\1_reda.pdf -mode 2 -keyword 1 1
fpdfredact -i test\2.pdf -o output\2_reda.pdf -mode 2 -keyword 1 1
fpdfredact -i c:\input -o d:\output -mode 2 -keyword 1 1
fpdfredact -i test -o output -mode 2 -keyword 1 1
fpdfredact -i "c:\input\*.pdf" -o d:\output -mode 2 -keyword 1 1
```

3.7.3.2 Redaction Settings (-character, -range, -markcolor, -tx, -ta, -font, -fs, -fontcolor, -form)

a) Redaction range of the Keyword (-character)

- The optional argument (**-character**) is used to specify the characters of the keyword you want to redact. If this argument is not set, all the characters of the keyword will be redacted by default. For more details about this argument, please refer to section 3.7.2 "Command Line Summary".

Usage Example

1) Redact only the third character of the keyword (-character 3)

```
fpdfredact -i c:\input\1.pdf -o d:\output\1_reda.pdf -mode 1 -keyword "Foxit" -character 3
fpdfredact -i test\2.pdf -o output\2_reda.pdf -mode 2 -keyword 1 1 -character 3
fpdfredact -i c:\input -o d:\output -mode 1 -keyword "Foxit Software" -character 3
fpdfredact -i test -o output -mode 2 -keyword 1 1 -character 3
fpdfredact -i "c:\input\*.pdf" -o d:\output -mode 2 -keyword 2 6 -character 3
```

2) Redact the first character to the fourth character of the keyword (-character 1 4)

```
fpdfredact -i c:\input\1.pdf -o d:\output\1_reda.pdf -mode 1 -keyword "Foxit" -character 1 4
fpdfredact -i test\2.pdf -o output\2_reda.pdf -mode 2 -keyword 1 1 -character 1 4
fpdfredact -i c:\input -o d:\output -mode 1 -keyword "Foxit Software" -character 1 4
fpdfredact -i test -o output -mode 2 -keyword 1 1 -character 1 4
fpdfredact -i "c:\input\*.pdf" -o d:\output -mode 2 -keyword 2 6 -character 1 4
```

b) Page range to apply redaction (-range)

- The optional argument (**-range**) is used to specify a page range to apply redaction. If this argument is not set, all of the pages will apply redaction. For more details about this argument, please refer to section 3.7.2 "Command Line Summary".

Usage Example

1) Apply redaction to pages 2,3 and 8 (-range "2,3,8")

```
fpdfredact -i c:\input\1.pdf -o d:\output\1_reda.pdf -mode 1 -keyword "Foxit" -range "2,3,8"
fpdfredact -i test\2.pdf -o output\2_reda.pdf -mode 2 -keyword 1 1 -range "2,3,8"
fpdfredact -i c:\input -o d:\output -mode 1 -keyword "Foxit Software" -range "2,3,8"
fpdfredact -i test -o output -mode 2 -keyword 1 1 -range "2,3,8"
fpdfredact -i "c:\input\*.pdf" -o d:\output -mode 2 -keyword 2 6 -range "2,3,8"
```

2) Apply redaction to all even pages (-range "even")

```
fpdfredact -i c:\input\1.pdf -o d:\output\1_reda.pdf -mode 1 -keyword "Foxit" -range "even"
fpdfredact -i test\2.pdf -o output\2_reda.pdf -mode 2 -keyword 1 1 -range "even"
fpdfredact -i c:\input -o d:\output -mode 1 -keyword "Foxit Software" -range "even"
fpdfredact -i test -o output -mode 2 -keyword 1 1 -range "even"
fpdfredact -i "c:\input\*.pdf" -o d:\output -mode 2 -keyword 2 6 -range "even"
```

3) Apply redaction to pages in the range from 2-10 and page 30 (-range "2-10,30")

```
fpdfredact -i c:\input\1.pdf -o d:\output\1_reda.pdf -mode 1 -keyword "Foxit" -range "2-10,30"
fpdfredact -i test\2.pdf -o output\2_reda.pdf -mode 2 -keyword 1 1 -range "2-10,30"
fpdfredact -i c:\input -o d:\output -mode 1 -keyword "Foxit Software" -range "2-10,30"
fpdfredact -i test -o output -mode 2 -keyword 1 1 -range "2-10,30"
fpdfredact -i "c:\input\*.pdf" -o d:\output -mode 2 -keyword 2 6 -range "2-10,30"
```

4) Apply redaction to all odd pages and all pages in the range from 100 to the last page (-range "odd,100-")

```
fpdfredact -i c:\input\1.pdf -o d:\output\1_reda.pdf -mode 1 -keyword "Foxit" -range "odd,100-"
fpdfredact -i test\2.pdf -o output\2_reda.pdf -mode 2 -keyword 1 1 -range "odd,100-"
fpdfredact -i c:\input -o d:\output -mode 1 -keyword "Foxit Software" -range "odd,100-"
fpdfredact -i test -o output -mode 2 -keyword 1 1 -range "odd,100-"
fpdfredact -i "c:\input\*.pdf" -o d:\output -mode 2 -keyword 2 6 -range "odd,100-"
```

c) Fill color for redaction marks (-markcolor)

- The optional argument (**-markcolor**) is used to specify the fill color used to mark for redaction with the RGB color model. If this argument is not set, the fill color is black by default. The allowable range of the values for each RGB component is from 0 to 255.

Usage Example

1) Set the fill color to blue for the redaction marks (-markcolor 0 0 255)

```
fpdfredact -i c:\input\1.pdf -o d:\output\1_reda.pdf -mode 1 -keyword "Foxit" -markcolor 0 0 255
fpdfredact -i test\2.pdf -o output\2_reda.pdf -mode 2 -keyword 1 1 -markcolor 0 0 255
fpdfredact -i c:\input -o d:\output -mode 1 -keyword "Foxit Software" -markcolor 0 0 255
fpdfredact -i test -o output -mode 2 -keyword 1 1 -markcolor 0 0 255
fpdfredact -i "c:\input\*.pdf" -o d:\output -mode 2 -keyword 2 6 -markcolor 0 0 255
```

d) Overlay text (-tx)

- The optional argument (**-tx**) is used to add overlay text to redaction marks. Overlay text appears on top of redaction marks, which is useful for sensitive content that needs to be removed after redaction. Users can specify custom text to appear over the redaction marks.

Usage Example

1) Use overlay text "secret" to appear on sensitive text "Foxit" selected for redaction (-tx "secret")

```
fpdfredact -i c:\input\1.pdf -o d:\output\1_reda.pdf -mode 1 -keyword "Foxit" -tx "secret"
fpdfredact -i test\2.pdf -o output\2_reda.pdf -mode 2 -keyword 1 1 -tx "secret"
fpdfredact -i c:\input -o d:\output -mode 1 -keyword "Foxit Software" -tx "secret"
fpdfredact -i test -o output -mode 2 -keyword 1 1  -tx "secret"
fpdfredact -i "c:\input\*.pdf" -o d:\output -mode 2 -keyword 2 6 -tx "secret"
```

e) Text alignment (-ta)

- The optional argument (**-ta**) is used to set alignment of the overlay text designated by the **-tx** option. It is valid only when (**-tx**) is used. The default value is 1, which will be left aligned.

Usage Example

1) Align the text to the left (-ta 1)

```
fpdfredact -i c:\input\1.pdf -o d:\output\1_reda.pdf -mode 1 -keyword "Foxit" -tx "secret" -ta 1
fpdfredact -i test\2.pdf -o output\2_reda.pdf -mode 2 -keyword 1 1 -tx "secret" -ta 1
fpdfredact -i c:\input -o d:\output -mode 1 -keyword "Foxit Software" -tx "secret" -ta 1
fpdfredact -i test -o output -mode 2 -keyword 1 1  -tx "secret" -ta 1
fpdfredact -i "c:\input\*.pdf" -o d:\output -mode 2 -keyword 2 6 -tx "secret" -ta 1
```

2) Align the text to the center (-ta 2)

```
fpdfredact -i c:\input\1.pdf -o d:\output\1_reda.pdf -mode 1 -keyword "Foxit" -tx "secret" -ta 2
fpdfredact -i test\2.pdf -o output\2_reda.pdf -mode 2 -keyword 1 1 -tx "secret" -ta 2
fpdfredact -i c:\input -o d:\output -mode 1 -keyword "Foxit Software" -tx "secret" -ta 2
fpdfredact -i test -o output -mode 2 -keyword 1 1  -tx "secret" -ta 2
fpdfredact -i "c:\input\*.pdf" -o d:\output -mode 2 -keyword 2 6 -tx "secret" -ta 2
```

3) Align the text to the right (-ta 3)

```
fpdfredact -i c:\input\1.pdf -o d:\output\1_reda.pdf -mode 1 -keyword "Foxit" -tx "secret" -ta 3
```

```
fpdfredact -i test\2.pdf -o output\2_reda.pdf -mode 2 -keyword 1 1 -tx "secret" -ta 3
fpdfredact -i c:\input -o d:\output -mode 1 -keyword "Foxit Software" -tx "secret" -ta 3
fpdfredact -i test -o output -mode 2 -keyword 1 1  -tx "secret" -ta 3
fpdfredact -i "c:\input\*.pdf" -o d:\output -mode 2 -keyword 2 6 -tx "secret" -ta 3
```

f) Text font (-font)

- The optional argument (**-font**) is used to set the font style of the overlay text designated by the **-tx** option. It is valid only when (**-tx**) is used.

  Note *The font style you set should be installed on a local environment, otherwise the default font style will be used.*

Usage Example

1) Set the font of the overlay text to "Calibri" (-font "Calibri")

```
fpdfredact -i c:\input\1.pdf -o d:\output\1_reda.pdf -mode 1 -keyword "Foxit" -tx "secret" -font "Calibri"
fpdfredact -i test\2.pdf -o output\2_reda.pdf -mode 2 -keyword 1 1 -tx "secret" -font "Calibri"
fpdfredact -i c:\input -o d:\output -mode 1 -keyword "Foxit Software" -tx "secret" -font "Calibri"
fpdfredact -i test -o output -mode 2 -keyword 1 1  -tx "secret" -font "Calibri"
fpdfredact -i "c:\input\*.pdf" -o d:\output -mode 2 -keyword 2 6 -tx "secret" -font "Calibri"
```

g) Text font size (-fs)

- The optional argument (**-fs**) is used to set the font size of the overlay text designated by the **-tx** option. It is valid only when (**-tx**) is used. The default value is set to 9, and the allowable range is from 8 to 72.

Usage Example

1) Set the font size of the overlay text to 12 (-fs 12)

```
fpdfredact -i c:\input\1.pdf -o d:\output\1_reda.pdf -mode 1 -keyword "Foxit" -tx "secret" -fs 12
fpdfredact -i test\2.pdf -o output\2_reda.pdf -mode 2 -keyword 1 1 -tx "secret" -fs 12
fpdfredact -i c:\input -o d:\output -mode 1 -keyword "Foxit Software" -tx "secret" -fs 12
fpdfredact -i test -o output -mode 2 -keyword 1 1  -tx "secret" -fs 12
fpdfredact -i "c:\input\*.pdf" -o d:\output -mode 2 -keyword 2 6 -tx "secret" -fs 12
```

h) Text font color (-fontcolor)

- The optional argument (**-fontcolor**) is used to set the font color of the overlay text designated by the **-tx** option with the RGB color model. It is valid only when (**-tx**) is used. By default, the font color is green. The allowable range of the values for each RGB component is from 0 to 255.

Usage Example

1) Set the font color of the overlay text to yellow (-fontcolor 255 255 0)

```
fpdfredact -i c:\input\1.pdf -o d:\output\1_reda.pdf -mode 1 -keyword "Foxit" -tx "xx" -fontcolor 255 255 0
fpdfredact -i test\2.pdf -o output\2_reda.pdf -mode 2 -keyword 1 1 -tx "xx" -fontcolor 255 255 0
fpdfredact -i c:\input -o d:\output -mode 1 -keyword "Foxit Software" -tx "xx" -fontcolor 255 255 0
fpdfredact -i test -o output -mode 2 -keyword 1 1  -tx "xx" -fontcolor 255 255 0
fpdfredact -i "c:\input\*.pdf" -o d:\output -mode 2 -keyword 2 6 -tx "xx" -fontcolor 255 255 0
```

i) Form (-form)

- The optional argument (**-form**) is used to search the specified keyword in both PDF forms and the main body of text. If this argument is not set, only main body of text will be searched.

  Note *If the keyword in the PDF form is generated by JavaScript, it will not be searched.*

Usage Example

1) Search the specified keyword in both PDF forms and the main body of text (-form)

```
fpdfredact -i c:\input\1.pdf -o d:\output\1_reda.pdf -mode 1 -keyword "Foxit" -form
fpdfredact -i test\2.pdf -o output\2_reda.pdf -mode 2 -keyword 1 1 -form
fpdfredact -i c:\input -o d:\output -mode 1 -keyword "Foxit Software" -form
fpdfredact -i test -o output -mode 2 -keyword 1 1 -form
fpdfredact -i "c:\input\*.pdf" -o d:\output -mode 2 -keyword 2 6 -form
```

3.7.3.3 Password for the Input File (-op)

- The optional argument (**-op**) indicates the password for a password-protected input PDF file. It is not required if the input file is not password protected. There are two kinds of passwords, one is user password, and the other is owner password.

User password is also known as "open password", which protects the document against unauthorized opening and reading.

Owner password is also known as "master password", which protects the document against unauthorized editing, printing, changing, and copying. It grants users full access to the document. With the owner password, the document can not only be opened and read, but also be allowed to change the document's security settings. This means that even if the permission of modifying the PDF file was restricted, you can still modify it with the owner password.

Note

- *The output PDF file will retain the password from the input file.*
- *If the input PDF file is protected by a user password, you need to provide the password using the option "-op" to process the PDF file.*
- *If the input PDF file is protected by an owner password, there are two circumstances. If the permissions of changing the PDF file were not restricted, you can process the PDF file without providing the owner password; otherwise, if the permissions of changing the PDF file were restricted, you need to provide the password using the option "-op" to process the PDF file.*
- *If the input PDF file is protected by a user password and an owner password, there are also two circumstances. If the permissions of changing the PDF file were not restricted, you can provide either of the passwords using the option "-op" to process the PDF file; otherwise, if the permissions of changing the PDF file were restricted, you need to provide the owner password using "-op" option to process the PDF file.*

Usage Example

1) Specify the password for a password-protected input PDF file (-op 123)

```
fpdfredact -i c:\input\1.pdf -o d:\output\1_reda.pdf -mode 1 -keyword "Foxit" -op 123
fpdfredact -i test\2.pdf -o output\2_reda.pdf -mode 2 -keyword 1 1 -op 123
```

2) Specify the password for all password-protected input PDF files in a given folder that share the same password (-op welcome)

```
fpdfredact -i c:\input -o d:\output -mode 1 -keyword "Foxit" -op welcome
fpdfredact -i test -o output -mode 2 -keyword 1 1 -op welcome
fpdfredact -i "c:\input\*.pdf" -o d:\output -mode 2 -keyword 2 8 -op welcome
```

Note *It only supports typing one value for the argument (**-op**). Only files with the same password can be processed together and files with different password need to be processed separately.*

3.7.3.4 Document Metadata Settings (-title, -subject, -keywords, -author, -creator)

a) Title (-title)

- The optional argument (**-title**) is used to set title of PDF files.

Usage Example

1) Set document title to "Foxit PDF Toolkit User Manual" (-title "Foxit PDF Toolkit User Manual")

```
fpdfredact -i c:\input\1.pdf -o d:\output\1_reda.pdf -mode 1 -keyword "Foxit" -title "Foxit PDF Toolkit User Manual"
fpdfredact -i test\2.pdf -o output\2_reda.pdf -mode 2 -keyword 1 1 -title "Foxit PDF Toolkit User Manual"
fpdfredact -i c:\input -o d:\output -mode 1 -keyword "Foxit Software" -title "Foxit PDF Toolkit User Manual"
fpdfredact -i test -o output -mode 2 -keyword 1 1  -title "Foxit PDF Toolkit User Manual"
fpdfredact -i "c:\input\*.pdf" -o d:\output -mode 2 -keyword 2 6 -title "Foxit PDF Toolkit User Manual"
```

b) Subject (-subject)

- The optional argument (**-subject**) is used to set subject of PDF files.

Usage Example

1) Set document subject to "Foxit PDF Toolkit" (-subject "Foxit PDF Toolkit")

```
fpdfredact -i c:\input\1.pdf -o d:\output\1_reda.pdf -mode 1 -keyword "Foxit" -subject "Foxit PDF Toolkit"
fpdfredact -i test\2.pdf -o output\2_reda.pdf -mode 2 -keyword 1 1 -subject "Foxit PDF Toolkit"
fpdfredact -i c:\input -o d:\output -mode 1 -keyword "Foxit Software" -subject "Foxit PDF Toolkit"
fpdfredact -i test -o output -mode 2 -keyword 1 1 -subject "Foxit PDF Toolkit"
fpdfredact -i "c:\input\*.pdf" -o d:\output -mode 2 -keyword 2 6 -subject "Foxit PDF Toolkit"
```

c) Keywords (-keywords)

- The optional argument (**-keywords**) is used to set keywords of PDF files.

Usage Example

1) Set document keywords to "toolkit" (-keywords "toolkit")

```
fpdfredact -i c:\input\1.pdf -o d:\output\1_reda.pdf -mode 1 -keyword "Foxit" -keywords "toolkit"
fpdfredact -i test\2.pdf -o output\2_reda.pdf -mode 2 -keyword 1 1 -keywords "toolkit"
fpdfredact -i c:\input -o d:\output -mode 1 -keyword "Foxit Software" -keywords "toolkit"
fpdfredact -i test -o output -mode 2 -keyword 1 1 -keywords "toolkit"
fpdfredact -i "c:\input\*.pdf" -o d:\output -mode 2 -keyword 2 6 -keywords "toolkit"
```

d) Author (-author)

- The optional argument (**-author**) is used to set author of PDF files.

Usage Example

1) Set document author to "Jessie" (-author "Jessie")

```
fpdfredact -i c:\input\1.pdf -o d:\output\1_reda.pdf -mode 1 -keyword "Foxit" -author "Jessie"
fpdfredact -i test\2.pdf -o output\2_reda.pdf -mode 2 -keyword 1 1 -author "Jessie"
fpdfredact -i c:\input -o d:\output -mode 1 -keyword "Foxit Software" -author "Jessie"
fpdfredact -i test -o output -mode 2 -keyword 1 1 -author "Jessie"
fpdfredact -i "c:\input\*.pdf" -o d:\output -mode 2 -keyword 2 6 -author "Jessie"
```

e) Creator (-creator)

- The optional argument (**-creator**) is used to set file creation application information of PDF files.

Usage Example

1) Set document creator to "Foxit Reader" (-creator "Foxit Reader")

```
fpdfredact -i c:\input\1.pdf -o d:\output\1_reda.pdf -mode 1 -keyword "Foxit" -creator "Foxit
Reader"
fpdfredact -i test\2.pdf -o output\2_reda.pdf -mode 2 -keyword 1 1 -creator "Foxit Reader"
fpdfredact -i c:\input -o d:\output -mode 1 -keyword "Foxit Software" -creator "Foxit Reader"
fpdfredact -i test -o output -mode 2 -keyword 1 1 -creator "Foxit Reader"
fpdfredact -i "c:\input\*.pdf" -o d:\output -mode 2 -keyword 2 6 -creator "Foxit Reader"
```

3.7.3.5 Recursion Depth of Sub-folders (-r)

- The optional argument (**-r**) indicates the recursion depth of sub-folders, which is valid only when the input is a folder or a path that includes wildcard characters like "c:\input*.pdf". By default, the recursion depth is 1, so the sub-folders will not be processed. For more details about this argument, please refer to section 3.7.2 "Command Line Summary".

Usage Examples

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

```
fpdfredact -i test -o output -mode 1 -keyword "Foxit" -r
fpdfredact -i c:\input -o d:\output -mode 2 -keyword 2 5 -r
fpdfredact -i c:\input\*.pdf -o d:\output -mode 2 -keyword 3 6 -r
fpdfredact -i test -o output -mode 1 -keyword "Foxit" -r 0
fpdfredact -i c:\input -o d:\output -mode 2 -keyword 2 5 -r 0
fpdfredact -i "c:\input\*.pdf" -o d:\output -mode 2 -keyword 3 6 -r 0
```

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

```
fpdfredact -i test -o output -mode 1 -keyword "Foxit" -r 1
fpdfredact -i c:\input -o d:\output -mode 2 -keyword 2 5 -r 1
fpdfredact -i "c:\input\*.pdf" -o d:\output -mode 2 -keyword 3 6 -r 1
```

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

```
fpdfredact -i test -o output -mode 1 -keyword "Foxit"
fpdfredact -i c:\input -o d:\output -mode 2 -keyword 2 5
fpdfredact -i "c:\input\*.pdf" -o d:\output -mode 2 -keyword 3 6
```

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

```
fpdfredact -i test -o output -mode 1 -keyword "Foxit" -r 2
fpdfredact -i c:\input -o d:\output -mode 2 -keyword 2 5 -r 2
fpdfredact -i "c:\input\*.pdf" -o d:\output -mode 2 -keyword 3 6 -r 2
```

3.7.3.6 Multi-thread Support (-t)

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

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

Usage Example

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

```
fpdfredact -i test -o output -mode 1 -keyword "Foxit" -t 3
fpdfredact -i c:\input -o d:\output -mode 2 -keyword 2 5 -t 3
fpdfredact -i "c:\input\*.pdf" -o d:\output -mode 2 -keyword 3 6 -t 3
```

3.7.3.7 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 argument (**-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 3.7.2 "Command Line Summary".

Usage Example

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

```
fpdfredact -i c:\input -o d:\output -mode 1 -keyword "Foxit" -log d:\output\pdfredactor.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 the **<licensee>** is the licensee name designated by the users.

Usage Example

1) Register the pdfredactor tool with the code "xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx" and the licensee "Foxit" (-register xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx Foixt)

```
fpdfredact -register xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx Foixt
```

c) License agreement (-license)

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

Usage Example

1) Print the license agreement (-license)

```
fpdfredact -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)

```
fpdfredact -version
fpdfredact -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)

```
fpdfredact -help
fpdfredact -h
```

f) Copyright information (-copyright)

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

Usage Example

5) Print the copyright information (-copyright)

```
fpdfredact -copyright
```

3.8 PDFMetadata

3.8.1 Basic Syntax

fpdfmeta *<-i <srcfile/srcfolder>> <-o <destfile/destfolder>> [-title <title>] [-subject <subject>]*
[-keywords <keywords>] [-author <author>] [-creator <creator>] [-op <password>]
[-r [recursion]] [-t <thread>] [-log <logfile>] [-l <level>]

fpdfmeta *<-i <srcfile>> <-properties> [-t 1] [-op <password>] [-log <logfile>] [-l <level>]*

fpdfmeta *-register <code> <licensee>*

fpdfmeta *-license*

fpdfmeta *-version/-v*

fpdfmeta *-help/-h*

fpdfmeta *-copyright*

Note:

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

If you want to set document metadata for PDF files, you should set not only *<-i <srcfile/srcfolder>>* and *<-o <destfile/destfolder>>* arguments, but also at least one of the following arguments (-title, -subject, -keywords,-author and -creator). If you just want to view PDF metadata information, the *<-i <srcfile>>* and *<-properties>* arguments are required.
All others are optional, which are available for controlling the process as desired. The arguments could be given in any order. Full details on each are explained in the following section.

3.8.2 Command Line Summary

Note *For some arguments whose values are strings, users can choose whether to add quotation marks (" "). In the manual we have added notes where adding quotation marks (" ") is required.*

Option	Parameter	Description
-i	*<-i <string>>* *e.g.* *-i c:\input\1.pdf* *-i c:\input* *-i "c:\input*.pdf"*	Specifies the input file to be processed. ▪ The input string can be the name of a single PDF file or a folder. ▪ The file name can contain the wildcard

Option	Parameter	Description
		character (*). For example, use *.pdf to include all PDF files in a given folder. **Note** Wildcard character (*.*) is currently not supported.
-o	*<-o <string>>* *e.g.* *-o d:\output\1_meta.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_meta.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.
-title	*[-title <string>]* *e.g.* *-title "Foxit PDF Toolkit User Manual"*	Sets title of PDF files.
-subject	*[-subject <string>]* *e.g.* *-subject "Foxit PDF Toolkit"*	Sets subject of PDF files.
-keywords	*[-keywords <string>]* *e.g.* *-keywords "Foxit"* *-keywords "img2pdf office2pdf pdfwatermark pdfmetadata"*	Sets keywords of PDF files.
-author	*[-author <string>]* *e.g.* *-author "Jessie"*	Sets author of PDF files.
-creator	*[-creator <string>]* *e.g.* *-creator "Foxit PhantomPDF"* *-creator "Foxit Reader"* *-creator "Microsoft® Word 2013"*	Sets creator of PDF files. **Note** It indicates the name of the application that created the source document from which a PDF is generated. For example, if the source document was created from Microsoft® *Word 2013, then you can set the creator to* "Microsoft® *Word 2013".*

Option	Parameter	Description
-properties	*<-properties>* *e.g.* *-properties*	Views metadata information of the PDF file designated by the **-i** option, such as Title, Subject, Author, Keywords and Creator. **Note** The argument (**-properties**) can only be used with **-i**, **-t**, **-op**, **-log** and **-l** options. The value of **-i** should be a PDF file, and the value of **-t** should be 1 if it is set.
-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 password protected. **Note** The output PDF file will retain the open password from the input file.
-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. • **-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** ▪ If no integer value is specified, or if the integer value is 0, then full folders will be searched. By default, the number of layers to recurse is 1, which means that only the current folder will be searched and not sub-folders. ▪ The input PDF file or folder will be skipped if it is secured and the messages will be displayed.
-t	*[-t <integer>]* *e.g.* *-t 1* *-t 2*	Specifies the number of CPU threads to use. The default value is 1.
-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 messages only concerning out of memory errors. • **-l 2**: logs failure messages concerning the errors caused during execution or those returned from underlying libraries, as well as those for level 1. • **-l 3**: logs warning messages concerning the PDF files that are overwritten, as well as those for level 2. • **-l 4**: logs informational messages, as well as those for level 3. **Note** The argument (**-l**) is valid only when (**-log**) is used.
-register	*[-register <String> <String>]* *-register <code> <licensee>* *e.g.* *-register* xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx *Foxit*	Registers the command line tool. ▪ **<code>**: the activation code from Foxit. ▪ **<licensee>**: the Licensee name designated by the users.
-help/-h	*[-help]/[-h]* *e.g.* *-help* *-h*	Prints the usage information.
-version/-v	*[-version]/[-v]* *e.g.* *-version* *-v*	Prints the version information.
-license	*[-license]* *e.g.* *-license*	Prints the license agreement.

3.8.3 Basic Usage

3.8.3.1 Input and Output (-i, -o)

a) Input (-i)

- The input should be a single PDF file or a folder. Users are not able to input multiple PDF files or folders, as well as a mixture composed 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 if the input file is in the current working folder. Users can input just the name of the PDF file or folder, instead of an absolute path. For example:

-i test\2.pdf	("test" folder 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, it is recommended to enclose the input files with quotation marks (" "). In this manual, we add (" ") whenever the input files contain wildcard characters.*

Note *If the input paths or file names contain spaces, you must enclose the input files with quotation marks (" "), such as -i "c:\input file\1.pdf", -i "test\user manual.pdf". This rule is also used for some other arguments whose values are strings.*

b) Output (-o)

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

-o d:\output\1_meta.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. Users can input just the name of the output PDF file or folder, instead of an absolute path. For example:

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

-o output ("output" folder is in the current working folder)

3.8.3.2 Document Metadata Settings (-title, -subject, -keywords, -author, -creator)

a) Title (-title)

- The optional argument (**-title**) is used to set title of PDF files.

Usage Example

1) Set document title to "Foxit PDF Toolkit User Manual" (-title "Foxit PDF Toolkit User Manual")

```
fpdfmeta -i c:\input\1.pdf -o d:\output\1_meta.pdf -title "Foxit PDF Toolkit User Manual"
fpdfmeta -i test\2.pdf -o output\2_meta.pdf -title "Foxit PDF Toolkit User Manual"
fpdfmeta -i test -o output -title "Foxit PDF Toolkit User Manual"
fpdfmeta -i c:\input -o d:\output -title "Foxit PDF Toolkit User Manual"
fpdfmeta -i "c:\input\*.pdf" -o d:\output -title "Foxit PDF Toolkit User Manual"
```

b) Subject (-subject)

- The optional argument (**-subject**) is used to set subject of PDF files.

Usage Example

1) Set document subject to "Foxit PDF Toolkit" (-subject "Foxit PDF Toolkit")

```
fpdfmeta -i c:\input\1.pdf -o d:\output\1_meta.pdf -subject "Foxit PDF Toolkit"
fpdfmeta -i test\2.pdf -o output\2_meta.pdf -subject "Foxit PDF Toolkit"
fpdfmeta -i test -o output -subject "Foxit PDF Toolkit"
fpdfmeta -i c:\input -o d:\output -subject "Foxit PDF Toolkit"
fpdfmeta -i "c:\input\*.pdf" -o d:\output -subject "Foxit PDF Toolkit"
```

c) Keywords (-keywords)

- The optional argument (**-keywords**) is used to set keywords of PDF files.

Usage Example

1) Set document keywords to "Foxit" (-keywords "Foxit")

```
fpdfmeta -i c:\input\1.pdf -o d:\output\1_meta.pdf -keywords "Foxit"
fpdfmeta -i test\2.pdf -o output\2_meta.pdf -keywords "Foxit"
fpdfmeta -i test -o output -keywords "Foxit"
fpdfmeta -i c:\input -o d:\output -keywords "Foxit"
fpdfmeta -i "c:\input\*.pdf" -o d:\output -keywords "Foxit"
```

2) Set document keywords to "image2pdf pdfmetadata pdfwatermark" (-keywords "image2pdf pdfmetadata pdfwatermark")

```
fpdfmeta -i c:\input\1.pdf -o d:\output\1_meta.pdf -keywords "image2pdf pdfmetadata pdfwatermark"
fpdfmeta -i test\2.pdf -o output\2_meta.pdf -keywords "image2pdf pdfmetadata pdfwatermark"
fpdfmeta -i test -o output -keywords "image2pdf pdfmetadata pdfwatermark"
fpdfmeta -i c:\input -o d:\output -keywords "image2pdf pdfmetadata pdfwatermark"
fpdfmeta -i "c:\input\*.pdf" -o d:\output -keywords "image2pdf pdfmetadata pdfwatermark"
```

d) Author (-author)

- The optional argument (**-author**) is used to set author of PDF files.

Usage Example

1) Set document author to "Jessie" (-author "Jessie")

```
fpdfmeta -i c:\input\1.pdf -o d:\output\1_meta.pdf -author "Jessie"
fpdfmeta -i test\2.pdf -o output\2_meta.pdf -author "Jessie"
fpdfmeta -i test -o output -author "Jessie"
fpdfmeta -i c:\input -o d:\output -author "Jessie"
fpdfmeta -i "c:\input\*.pdf" -o d:\output -author "Jessie"
```

e) Creator (-creator)

- The optional argument (**-creator**) is used to set file creation application information of PDF files.

Usage Example

1) Set document creator to "Foxit Reader" (-creator "Foxit Reader")

```
fpdfmeta -i c:\input\1.pdf -o d:\output\1_meta.pdf -creator "Foxit Reader"
fpdfmeta -i test\2.pdf -o output\2_meta.pdf -creator "Foxit Reader"
```

```
fpdfmeta -i test -o output -creator "Foxit Reader"
fpdfmeta -i c:\input -o d:\output -creator "Foxit Reader"
fpdfmeta -i "c:\input\*.pdf" -o d:\output -creator "Foxit Reader"
```

3.8.3.3 Password for the Input File (-op)

- The optional argument (**-op**) indicates the password for a password-protected input PDF file. It is not required if the input file is not password protected. There are two kinds of passwords, one is user password, and the other is owner password.

  User password is also known as "open password", which protects the document against unauthorized opening and reading.

  Owner password is also known as "master password", which protects the document against unauthorized editing, printing, changing, and copying. It grants users full access to the document. With the owner password, the document can not only be opened and read, but also be allowed to change the document's security settings. This means that even if the permission of modifying the PDF file was restricted, you can still modify it with the owner password.

  Note

  - *The output PDF file will retain the password from the input file.*
  - *If the input PDF file is protected by a user password, you need to provide the password using the option "-op" to process the PDF file.*
  - *If the input PDF file is protected by an owner password, there are two circumstances. If the permissions of changing the PDF file were not restricted, you can process the PDF file without providing the owner password; otherwise, if the permissions of changing the PDF file were restricted, you need to provide the password using the option "-op" to process the PDF file.*
  - *If the input PDF file is protected by a user password and an owner password, there are also two circumstances. If the permissions of changing the PDF file were not restricted, you can provide either of the passwords using the option "-op" to process the PDF file; otherwise, if the permissions of changing the PDF file were restricted, you need to provide the owner password using "-op" option to process the PDF file.*

Usage Example

1) Specify the password for a password-protected input PDF file (-op 123)

```
fpdfmeta -i c:\input\1.pdf -o d:\output\1_meta.pdf -author "Jessie" -subject "Foxit PDF Toolkit" -
keywords "Foxit" -op 123
```

fpdfmeta -i test\2.pdf -o output\2_meta.pdf -title "Foxit PDF Toolkit User Manual" -creator "Foxit Phantom" -op 123

2) Specify the password for all password-protected input PDF files in a given folder that share the same password (-op welcome)

fpdfmeta -i c:\input -o d:\output -author "Jessie" -subject "Foxit PDF Toolkit" -keywords "Foxit" -op welcome
fpdfmeta -i test -o output -title "Foxit PDF Toolkit User Manual" -creator "Foxit Phantom" -op welcome
fpdfmeta -i "c:\input*.pdf" -o d:\output -title "Foxit PDF Toolkit User Manual" -op welcome

Note *It only supports typing one value for the argument (-op). Only files with the same password can be processed together and files with different password need to be processed separately.*

3.8.3.4 View metadata information (-properties)

- The optional argument (**-properties**) is used to view metadata information of the PDF file designated by the **-i** option.

  Note *The argument (**-properties**) can only be used with **-i**, **-t**, **-op**, **-log** and **-l** options. The value of **-i** should be a PDF file, and the value of **-t** should be 1 if it is set.*

Usage Example

1) View the metadata information of the document (foxit.pdf) (-i c:\foxit.pdf -properties)

fpdfmeta -i c:\foxit.pdf -properties

2) View the metadata information of the document (user_manual.pdf) and specify the open password (-i c:\user_manual.pdf -op 123 -properties)

fpdfmeta -i c:\user_manual.pdf -op 123 -properties

3.8.3.5 Recursion Depth of Sub-folders (-r)

- The optional argument (**-r**) indicates the recursion depth of sub-folders, which is valid only when the input is a folder or a path that includes wildcard characters like "c:\input*.pdf". By default, the recursion depth is 1, so the sub-folders will not be processed. For more details about this argument, please refer to section 3.8.2 "Command Line Summary".

Usage Examples

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

```
fpdfmeta -i test -o output -title "Foxit PDF Toolkit User Manual" -author "Jessie" -r
fpdfmeta -i c:\input -o d:\output -subject "Foxit PDF Toolkit" -keywords "Foxit" -r
fpdfmeta -i c:\input\*.pdf -o d:\output -creator "Foxit Reader"-r
fpdfmeta -i test -o output -title "Foxit PDF Toolkit User Manual" -author "Jessie" -r 0
fpdfmeta -i c:\input -o d:\output -subject "Foxit PDF Toolkit" -keywords "Foxit" -r 0
fpdfmeta -i "c:\input\*.pdf" -o d:\output -creator "Foxit Reader" -r 0
```

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

```
fpdfmeta -i test -o output -title "Foxit PDF Toolkit User Manual" -author "Jessie" -r 1
fpdfmeta -i c:\input -o d:\output -subject "Foxit PDF Toolkit" -keywords "Foxit" -r 1
fpdfmeta -i "c:\input\*.pdf" -o d:\output -creator "Foxit Reader" -r 1
```

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

```
fpdfmeta -i test -o output -title "Foxit PDF Toolkit User Manual" -author "Jessie"
fpdfmeta -i c:\input -o d:\output -subject "Foxit PDF Toolkit" -keywords "Foxit"
fpdfmeta -i "c:\input\*.pdf" -o d:\output -creator "Foxit Reader"
```

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

```
fpdfmeta -i test -o output -title "Foxit PDF Toolkit User Manual" -author "Jessie" -r 2
fpdfmeta -i c:\input -o d:\output -subject "Foxit PDF Toolkit" -keywords "Foxit" -r 2
fpdfmeta -i "c:\input\*.pdf" -o d:\output -creator "Foxit Reader" -r 2
```

3.8.3.6 Multi-thread Support (-t)

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

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

Usage Example

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

```
fpdfmeta -i test -o output -title "Foxit PDF Toolkit User Manual" -author "Jessie" -t 3
fpdfmeta -i c:\input -o d:\output -subject "Foxit PDF Toolkit" -keywords "Foxit" -t 3
fpdfmeta -i "c:\input\*.pdf" -o d:\output -creator "Foxit Reader" -t 3
```

3.8.3.7 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 argument (**-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 3.8.2 "Command Line Summary".

Usage Example

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

```
fpdfmeta -i c:\input -o d:\output -keywords "Foxit" -log d:\output\pdfmetadata.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 the **<licensee>** is the licensee name designated by the users.

Usage Example

1) Register the pdfmetadata tool with the code "xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx" and the licensee "Foxit" (-register xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx Foixt)

```
fpdfmeta -register xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx Foixt
```

c) License agreement (-license)

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

Usage Example

1) Print the license information (-license)

```
fpdfmeta -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)

```
fpdfmeta -version
fpdfmeta -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)

```
fpdfmeta -help
fpdfmeta -h
```

f) Copyright information (-copyright)

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

Usage Example

6) Print the copyright information (-copyright)

```
fpdfmeta -copyright
```

3.9 PDF2Text

3.9.1 Basic Syntax

fpdf2txt *<-i <srcfile/srcfolder>> <-o <destfile/destfolder>> [-range <page range>] [-charcount] [-printcount] [-notype] [-nopagenumber] [-singlepage] [-encoding <text encoding>] [-breakpage] [-op <password>] [-r [recursion]] [-t <thread>] [-log <logfile>] [-l <level>]*

fpdf2txt *-register <code> <licensee>*

fpdf2txt *-license*

fpdf2txt *-version/-v*

fpdf2txt *-help/-h*

fpdf2txt *-copyright*

Note:

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

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

3.9.2 Command Line Summary

Note *For some arguments whose values are strings, users can choose whether to add quotation marks (" ") to the strings. In the manual we have added notes where adding quotation marks (" ") is required.*

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

Option	Parameter	Description
-o	*<-o <string>>* *e.g.* *-o d:\output\1.txt* *-o d:\output*	Specifies the path of the output text file or folder. ▪ If the input is a single PDF file, the output should be a single text file, (e.g. *-o d:\output\1.txt*). ▪ 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.
-range	*[-rang <String>]* *e.g.* *-range "1,5,9"* *-range "all"* *-range "even"* *-range "odd"* *-range "2-10,30"* *-range "10,50-"* *-range "odd,100-"*	Specifies a page range to convert. By default, all of the pages will be converted. ▪ Converts page 1,5, and 9: **-range "1,5,9"** ▪ Converts all pages: **-range "all"** ▪ Converts all even pages: **-range "even"** ▪ Converts all odd pages: **-range "odd"** ▪ Converts pages in the range from 2-10 and page 30: **-range "2-10,30"** ▪ Converts page 10 and all pages in the range from 50 to the last page: **-range "10,50-"** ▪ Converts all odd pages and all pages in the range from 100 to the last page: **-range "odd,100-"**
-charcount	*[-charcount]* *e.g.* *-charcount*	Gets the total number of characters on each page. **Note** The content of generated text files will only contain the number of characters on each page, if this argument is set.
-printcount	*[-printcount]* *e.g.* *-printcount*	Prints the number of characters to the screen. **Note:** The argument (**-printcount**) is valid only when (**-charcount**) is used.

Option	Parameter	Description
-notype	*[-notype]* *e.g.* *-notype*	Disables retaining the PDF page layout for the output text files.
-nopagenumber	*[-nopagenumber]* *e.g.* *-nopagenumber*	Disables displaying the page number in the output text files.
-singlepage	*[-singlepage]* *e.g.* *-singlepage*	Converts each PDF page into individual text files.
-encoding	*[-encoding <sting>]* *<-encoding <text encoding>>* *e.g.* *-encoding UTF8* *-encoding UTF16*	Specifies Unicode text encoding. The allowable value is UTF8 and UTF16. The default is UTF8.
-breakpage	*[-breakpage]* *e.g.* *-breakpage*	Adds a page break to represent the end of each PDF page in the output text files. **Note** A page break in text files is encoded with the Form Feed (new page) ASCII character (12 (0xC in hexadecimal)).
-op	*[-op<string>]* *e.g.* *-op 123* *-op welcome*	Specifies the password for the input file. Not required if the input file is not password protected.
-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. • **-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** ▪ If no integer value is specified, or if the integer value is 0, then full folders will be searched. By default, the number of layers to recurse is 1, which means that only the current folder will

Option	Parameter	Description
		be searched and not sub-folders. ▪ The input PDF file or folder will be skipped if it is secured and the messages will be displayed.
-t	*[-t <integer>]* *e.g.* *-t 1* *-t 2* *…*	Specifies the number of CPU threads to use. The default value is 1.
-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 messages only concerning out of memory errors. • **-l 2**: logs failure messages concerning the errors caused during execution or those returned from underlying libraries, as well as those for level 1. • **-l 3**: logs warning messages concerning the PDF files that are overwritten, as well as those for level 2. • **-l 4**: logs informational messages, as well as those for level 3. **Note** The argument (**-l**) is valid only when (**-log**) is used.
-register	*[-register <String> <String>]* *-register <code> <licensee>* *e.g.* *-register* xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx *Foxit*	Registers the command line tool. ▪ **<code>**: the activation code from Foxit. ▪ **<licensee>**: the Licensee name designated by the users.
-help/-h	*[-help]/[-h]* *e.g.* *-help* *-h*	Prints the usage information.
-version/-v	*[-version]/[-v]* *e.g.*	Prints the version information.

Option	Parameter	Description
	-version *-v*	
-license	*[-license]* *e.g.* *-license*	Prints the license agreement.

3.9.3 Basic Usage

3.9.3.1 Input and Output (-i, -o)

a) Input (-i)

- The input should be a single PDF file or a folder. Users are not able to input multiple PDF files or folders, as well as a mixture composed 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 if the input file is in the current working folder. Users can input just the name of the PDF file or folder, instead of an absolute path. For example:

  -i test\1.pdf ("test" folder 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, it is recommended to enclose the input files with quotation marks (" "). In this manual, we add (" ") whenever the input files contain wildcard characters.*

Note *If the input paths or file names contain spaces, you must enclose the input files with quotation marks (" "), such as -i "c:\input file\1.pdf", -i "test\user manual.pdf". This rule is also used for some other arguments whose values are strings.*

b) Output (-o)

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

  -o d:\output\output.txt (a single text 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. Users can input just the name of the output text file or folder, instead of an absolute path. For example:

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

Usage Examples

1) Convert a single PDF file to text file:

```
fpdf2txt -i c:\input\1.pdf -o d:\output\1.txt
fpdf2txt -i test\2.pdf -o output\2.txt
```

2) Convert the PDF files in a folder to text files:

```
fpdf2txt -i c:\input -o d:\output
fpdf2txt -i test -o output
fpdf2txt -i c:\input\*.pdf -o d:\output
fpdf2txt -i "test\*.pdf" -o output
```

3.9.3.2 Conversion Settings (-range, -charcount, -printcount, -notype, -nopagenumber, -singlepage, -encoding, -breakpage)

a) Page range to convert (-range)

- The optional argument (**-range**) is used to specify a page range to convert. If this argument is not set, all of the pages will be converted. For more details about this argument, please refer to section 3.9.2 "Command Line Summary".

Usage Example

1) Convert pages 2,3 and 8 (-range "2,3,8")

```
fpdf2txt -i c:\input\1.pdf -o d:\output\1.txt -range "2,3,8"
fpdf2txt -i test\2.pdf -o output\2.txt -range "2,3,8"
fpdf2txt -i c:\input -o d:\output -range "2,3,8"
fpdf2txt -i test -o output -range "2,3,8"
fpdf2txt -i "c:\input\*.pdf" -o d:\output -range "2,3,8"
```

2) Convert all even pages (-range "even")

```
fpdf2txt -i c:\input\1.pdf -o d:\output\1.txt -range "even"
fpdf2txt -i test\2.pdf -o output\2.txt -range "even"
fpdf2txt -i c:\input -o d:\output -range "even"
fpdf2txt -i test -o output -range "even"
fpdf2txt -i "c:\input\*.pdf" -o d:\output -range "even"
```

3) Convert pages in the range from 2-10 and page 30 (-range "2-10,30")

```
fpdf2txt -i c:\input\1.pdf -o d:\output\1.txt -range "2-10,30"
fpdf2txt -i test\2.pdf -o output\2.txt -range "2-10,30"
fpdf2txt -i c:\input -o d:\output -range "2-10,30"
fpdf2txt -i test -o output -range "2-10,30"
fpdf2txt -i "c:\input\*.pdf" -o d:\output -range "2-10,30"
```

4) Convert all odd pages and all pages in the range from 100 to the last page (-range "odd,100-")

```
fpdf2txt -i c:\input\1.pdf -o d:\output\1.txt -range "odd,100-"
fpdf2txt -i test\2.pdf -o output\2.txt -range "odd,100-"
fpdf2txt -i c:\input -o d:\output -range "odd,100-"
fpdf2txt -i test -o output -range "odd,100-"
fpdf2txt -i "c:\input\*.pdf" -o d:\output -range "odd,100-"
```

b) Total number of characters on each page (-charcount)

- The optional argument (-charcount) is used to get the total number of characters on each page. The content of generated text files will only contain the number of characters on each page, if this argument is set.

Usage Example

1) Get the total number of characters on each page (-charcount)

```
fpdf2txt -i c:\input\1.pdf -o d:\output\1.txt -charcount
fpdf2txt -i test\2.pdf -o output\2.txt -charcount
fpdf2txt -i c:\input -o d:\output -charcount
fpdf2txt -i test -o output -charcount
fpdf2txt -i "c:\input\*.pdf" -o d:\output -charcount
```

c) Print characters number to the screen (-printcount)

- The optional argument (-printcount) is used to print the number of characters to the screen. It is valid only when (**-charcount**) is used.

Usage Example

1) Print the number of characters to the screen (-printcount)

```
fpdf2txt -i c:\input\1.pdf -o d:\output\1.txt -charcount -printcount
fpdf2txt -i test\2.pdf -o output\2.txt -charcount -printcount
fpdf2txt -i c:\input -o d:\output -charcount -printcount
fpdf2txt -i test -o output -charcount -printcount
fpdf2txt -i "c:\input\*.pdf" -o d:\output -charcount -printcount
```

d) Ignore the PDF page layout (-notype)

- The optional argument (**-notype**) is used to ignore the PDF page layout. If this argument is set, the text file will not retain the PDF page layout.

Usage Example

1) Ignore the PDF page layout (-notype)

```
fpdf2txt -i c:\input\1.pdf -o d:\output\1.txt -notype
fpdf2txt -i test\2.pdf -o output\2.txt -notype
fpdf2txt -i c:\input -o d:\output -notype
fpdf2txt -i test -o output -notype
fpdf2txt -i "c:\input\*.pdf" -o d:\output -notype
```

e) Disable displaying the page number (-nopagenumber)

- The optional argument (**-nopagenumber**) is used to disable displaying the page number for the output text files. If this argument is set, the text file will not display the page number.

Usage Example

1) Disable displaying the page number for the output text files (-nopagenumber)

```
fpdf2txt -i c:\input\1.pdf -o d:\output\1.txt -nopagenumber
fpdf2txt -i test\2.pdf -o output\2.txt -nopagenumber
fpdf2txt -i c:\input -o d:\output -nopagenumber
fpdf2txt -i test -o output -nopagenumber
fpdf2txt -i "c:\input\*.pdf" -o d:\output -nopagenumber
```

f) Convert each PDF page into individual text files (-singlepage)

- The optional argument (**-singlepage**) is used to convert each PDF page into individual text files.

Usage Example

1) Convert each PDF page into individual text files (-singlepage)

```
fpdf2txt -i c:\input\1.pdf -o d:\output\1.txt -singlepage
fpdf2txt -i test\2.pdf -o output\2.txt -singlepage
fpdf2txt -i c:\input -o d:\output -singlepage
fpdf2txt -i test -o output -singlepage
fpdf2txt -i "c:\input\*.pdf" -o d:\output -singlepage
```

g) Text encoding (-encoding)

- The optional argument (**-encoding**) is used to specify Unicode text encoding. The allowable value is UTF8 and UTF16. By default, the text encoding is UTF8.

Usage Example

1) Set text encoding to UTF16 (-encoding UTF16)

```
fpdf2txt -i c:\input\1.pdf -o d:\output\1.txt -encoding UTF16
fpdf2txt -i test\2.pdf -o output\2.txt -encoding UTF16
fpdf2txt -i c:\input -o d:\output -encoding UTF16
fpdf2txt -i test -o output -encoding UTF16
```

fpdf2txt -i "c:\input*.pdf" -o d:\output -encoding UTF16

h) Add a page break to represent the end of each PDF page (-breakpage)

- The optional argument (**-breakpage**) is used to add a page break to represent the end of each PDF page in the output text files. The page break is useful if you want to convert the generated text files into PDF files.

  Note *A page break in text files is encoded with the Form Feed (new page) ASCII character (12 (0xC in hexadecimal)).*

Usage Example

1) Add a page break to represent the end of each PDF page in the output text files (-breakpage)

```
fpdf2txt -i c:\input\1.pdf -o d:\output\1.txt -breakpage
fpdf2txt -i test\2.pdf -o output\2.txt -breakpage
fpdf2txt -i c:\input -o d:\output -breakpage
fpdf2txt -i "c:\input\*.pdf" -o d:\output -breakpage
```

3.9.3.3 Password for the Input File (-op)

- The optional argument (**-op**) indicates the password for a password-protected input PDF file. It is not required if the input file is not password protected. There are two kinds of passwords, one is user password, and the other is owner password.

  User password is also known as "open password", which protects the document against unauthorized opening and reading.

  Owner password is also known as "master password", which protects the document against unauthorized editing, printing, changing, and copying. It grants users full access to the document. With the owner password, the document can not only be opened and read, but also be allowed to change the document's security settings. This means that even if the permission of modifying the PDF file was restricted, you can still modify it with the owner password.

  Note

  - *If the input PDF file is protected by a user password, you need to provide the password using the option "-op" to process the PDF file.*
  - *If the input PDF file is protected by an owner password, there are two circumstances. If the permissions of changing the PDF file were not restricted, you can process the PDF file without*

providing the owner password; otherwise, if the permissions of changing the PDF file were restricted, you need to provide the password using the option "-op" to process the PDF file.

- *If the input PDF file is protected by a user password and an owner password, there are also two circumstances. If the permissions of changing the PDF file were not restricted, you can provide either of the passwords using the option "-op" to process the PDF file; otherwise, if the permissions of changing the PDF file were restricted, you need to provide the owner password using "-op" option to process the PDF file.*

Usage Example

1) Specify the password for a password-protected input PDF file (-op 123)

```
fpdf2txt -i c:\input\1.pdf -o d:\output\1.txt -op 123
fpdf2txt -i test\2.pdf -o output\2.txt -op 123
```

2) Specify the password for all password-protected input PDF files in a given folder that share the same password (-op welcome)

```
fpdf2txt -i c:\input -o d:\output -op welcome
fpdf2txt -i test -o output -op welcome
fpdf2txt -i "c:\input\*.pdf" -o d:\output -op welcome
```

Note *It only supports typing one value for the argument (**-op**). Only files with the same password can be processed together and files with different password need to be processed separately.*

3.9.3.4 Recursion Depth of Sub-folders (-r)

- The optional argument (**-r**) indicates the recursion depth of sub-folders, which is valid only when the input is a folder or a path that includes wildcard characters like "c:\input*.pdf". By default, the recursion depth is 1, so the sub-folders will not be processed. For more details about this argument, please refer to section 3.9.2 "Command Line Summary".

Usage Examples

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

```
fpdf2txt -i c:\input -o d:\output -r
fpdf2txt -i test -o output -r
fpdf2txt -i "c:\input\*.pdf" -o d:\output -r
fpdf2txt -i c:\input -o d:\output -r 0
```

```
fpdf2txt -i test -o output -r 0
fpdf2txt -i "c:\input\*.pdf" -o d:\output -r 0
```

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

```
fpdf2txt -i c:\input -o d:\output -r 1
fpdf2txt -i test -o output -r 1
fpdf2txt -i "c:\input\*.pdf" -o d:\output -r 1
```

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

```
fpdf2txt -i c:\input -o d:\output
fpdf2txt -i test -o output
fpdf2txt -i "c:\input\*.pdf" -o d:\output
```

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

```
fpdf2txt -i c:\input -o d:\output -r 2
fpdf2txt -i test -o output -r 2
fpdf2txt -i "c:\input\*.pdf" -o d:\output -r 2
```

3.9.3.5 Multi-thread Support (-t)

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

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

Usage Example

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

```
fpdf2txt -i c:\input -o d:\output -t 3
fpdf2txt -i test -o output -t 3
fpdf2txt -i "c:\input\*.pdf" -o d:\output -t 3
```

3.9.3.6 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 argument (**-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 3.9.2 "Command Line Summary".

Usage Example

1) Save the log file to "d:\output\pdf2text.log" and set the log level to 3 (-log d:\output\pdf2text.log -l 3)

```
fpdf2txt -i c:\input -o d:\output -log d:\output\pdf2text.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 the **<licensee>** is the licensee name designated by the users.

Usage Example

1) Register the pdf2text tool with the code "xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx" and the licensee "Foxit" (-register xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx Foixt)

```
fpdf2txt -register xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx Foixt
```

c) License agreement (-license)

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

Usage Example

1) Print the license agreement (-license)

```
fpdf2txt -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)

```
fpdf2txt -version
```

fpdf2txt -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)

fpdf2txt -help
fpdf2txt -h

f) Copyright information (-copyright)

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

Usage Example

1) Print the copyright information (-copyright)

fpdf2txt -copyright

3.10 Text2PDF

3.10.1 Basic Syntax

ftxt2pdf *<-i <srcfile/srcfolder>> <-o <destfile/destfolder>> [-width <PDF width>] [-height <PDF height>]*
[-margin <left [top right bottom]>] [-breakpage] [-linespace <point>]
[-font <fontname>] [-fs <fontsize>] [-fontcolor <R> <G> <B>] [-sp <password>]
[-title <title>] [-subject <subject>] [-keywords <keywords>] [-author <author>] [-creator <creator>]
[-r [recursion]] [-t <thread>] [-log <logfile>] [-l <level>]

ftxt2pdf *-register <code> <licensee>*

ftxt2pdf *-license*

ftxt2pdf *-version/-v*

ftxt2pdf *-help/-h*

ftxt2pdf *-copyright*

Note:

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

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

3.10.2 Command Line Summary

Note *For some arguments whose values are strings, users can choose whether to add quotation marks (" ") to the strings. In the manual we have added notes where adding quotation marks (" ") is required.*

Option	Parameter	Description
-i	*<-i <string>>* *e.g.* *-i c:\input\1.txt* *-i c:\input* *-i "c:\input*.txt"*	Specifies the input file to be converted. ▪ The input string can be the name of a single text file or a folder. ▪ The file name can contain the wildcard character (*). For example, use *.txt to include all text files in a given folder. **Note** Wildcard character (*.*) is currently not supported.
-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 text 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 folder, (e.g. *-o d:\output*). **Note** The specified output path must already exist.
-width	*[-width <points>]* *e.g.* *-width 612*	Sets the page width of the output PDF file in points. Default width value: 595 points. Allowable range: 8-14400 points.
-height	*[-height <points>]* *e.g.* *-height 792*	Sets the page height of the output PDF file in points. Default height value: 842 points. Allowable range: 8-14400 points.
-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 margin values: 60 72 60 72. Allowable values: 0-size of page in points; in addition, the sum of the left and right values should be less than the width of the page, and the sum of the top and bottom values should be less than the height of the page. ***-margin left top right bottom*** **-margin 20**: sets the left margin to 20 points. **-margin 10 20**: sets the left margin to 10 points and the top margin to 20 points. **-margin 10 20 0**: sets the left margin to 10

Option	Parameter	Description
		points, the top margin to 20 points, and the right margin to 0 points. **-margin 10 20 0 40**: sets 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.
-breakpage	*[-breakpage]* *e.g.* *-breakpage*	If this flag is set, the output PDF file will be paginated wherever there is a page break. **Note** A page break in text files is encoded with the Form Feed (new page) ASCII character (12 (0xC in hexadecimal)).
-linespace	*[-linespace <point(pt)>]* *e.g.* *-linespace 0* *-linespace 5*	Sets the line spacing of the output PDF file. Default value: 0. Allowable range: 0-72pt.
-font	*[-font <string>]* *-font <fontname>* *e.g.* *-font Calibri* *-font Helvetica* *-font Arial* …	Sets the font style of the text in the output PDF files. **Note** The font style you set should be installed on a local environment, otherwise the default font style will be used.
-fs	*[-fs <point(pt)>]* *-fs <fontsize>* *e.g.* *-fs 8* *-fs 11* *-fs 72*	Sets the font size of the text in the output PDF files. Default value: 9pt. Allowable range: 8-72pt.
-fontcolor	*[-fontcolor <integer> <integer> <integer>]* *-fontcolor <R> <G> <B>* *e.g.* *-fontcolor 0 0 0* *-fontcolor 255 255 0*	Sets the font color of the text in the output PDF files. By default, the font color of the output PDF is black. Allowable range of the values for each RGB component is 0-255.

Option	Parameter	Description
	-fontcolor 255 255 255 …	
-sp	*[-sp<string>]* *e.g.* *-sp welcome*	Sets the document open password of the output PDF as the "string". By default, there is no password.
-title	*[-title <string>]* *e.g.* *-title "Foxit PDF Toolkit User Manual"*	Sets title of PDF files.
-subject	*[-subject <string>]* *e.g.* *-subject "Foxit PDF Toolkit"*	Sets subject of PDF files.
-keywords	*[-keywords <string>]* *e.g.* *-keywords "Foxit"*	Sets keywords of PDF files.
-author	*[-author <string>]* *e.g.* *-author "Jessie"*	Sets author of PDF files.
-creator	*[-creator <string>]* *e.g.* *-creator "Foxit PhantomPDF"* *-creator "Foxit Reader"* *-creator "Microsoft® Word 2013"*	Sets creator of PDF files. **Note** It indicates the name of the application that created the source document from which a PDF is generated. For example, if the source document was created from *Microsoft® Word 2013,* then you can set the creator to "*Microsoft® Word 2013*".
-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. • **-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**

Option	Parameter	Description
		▪ If no integer value is specified, or if the integer value is 0, then full folders will be searched. By default, the number of layers to recurse is 1, which means that only the current folder will be searched and not sub-folders. ▪ The input folder will be skipped if it is secured and the messages will be displayed.
-t	*[-t <integer>]* *e.g.* *-t 1* *-t 2* *…*	Specifies the number of CPU threads to use. The default value is 1.
-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 messages only concerning out of memory errors. • **-l 2**: logs failure messages concerning the errors caused during execution or those returned from underlying libraries, as well as those for level 1. • **-l 3**: logs warning messages concerning the PDF files that are overwritten, as well as those for level 2. • **-l 4**: logs informational messages, as well as those for level 3. **Note** The argument (**-l**) is valid only when (**-log**) is used.
-register	*[-register <String> <String>]* *-register <code> <licensee>* *e.g.* *-register* xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx *Foxit*	Registers the command line tool. ▪ **<code>**: the activation code from Foxit. ▪ **<licensee>**: the Licensee name designated by the users.

Option	Parameter	Description
-help/-h	*[-help]/[-h]* *e.g.* *-help* *-h*	Prints the usage information.
-version/-v	*[-version]/[-v]* *e.g.* *-version* *-v*	Prints the version information.
-license	*[-license]* *e.g.* *-license*	Prints the license agreement.

3.10.3 Basic Usage

3.10.3.1 Input and Output (-i, -o)

a) Input (-i)

- The input should be a single text file or a folder. Users are not able to input multiple files or folders, as well as a mixture composed of folders and text files. For example:

  -i c:\input\1.txt (a single text file)
  -i c:\input (a single folder)

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

  -i test\1.txt ("test" folder 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 files. For example:

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

  Note *When using wildcard characters in the input files, it is recommended to enclose the input files with quotation marks (" "). In this manual, we add (" ") whenever the input files contain wildcard characters.*

Note *If the input paths or file names contain spaces, you must enclose the input files with quotation marks (" "), such as -i "c:\input file\1.txt", -i "test\user manual.txt". This rule is also used for some other arguments whose values are strings.*

b) Output (-o)

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

  -o d:\output\output.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. Users can input just the name of the output PDF file or folder, instead of an absolute path. For example:

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

Usage Examples

1) Convert a single text file to PDF:

```
ftxt2pdf -i c:\input\1.txt -o d:\output\1.pdf
ftxt2pdf -i test\2.txt -o output\2.pdf
```

2) Convert the text files in a folder to PDF:

```
ftxt2pdf -i c:\input -o d:\output
ftxt2pdf -i test -o output
ftxt2pdf -i c:\input\*.txt -o d:\output
ftxt2pdf -i test\*.txt -o output
```

3.10.3.2 PDF/page Settings for Conversion (-width, -height, -margin, -breakpage, -linespace, -sp)

a) 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. The default width and height are 595 and 842 points respectively with the allowable range of 8-14400 points.

Usage Example

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

```
ftxt2pdf -i c:\input\1.txt -o d:\output\1.pdf -width 400 -height 300
ftxt2pdf -i c:\input -o d:\output -width 400 -height 300
ftxt2pdf -i test -o output -width 400 -height 300
ftxt2pdf -i "c:\input\*.txt" -o d:\output -width 400 -height 300
```

b) Margin setting (-margin)

- The optional argument (-margin) is used to set size of margin for each PDF page in points. The default margin values are 60 72 60 72. For more details about this argument, please refer to section 3.10.2 "Command Line Summary".

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

Usage Example

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

```
ftxt2pdf -i c:\input\1.txt -o d:\output\1.pdf -margin 20
ftxt2pdf -i c:\input -o d:\output -margin 20
ftxt2pdf -i test -o output -margin 20
ftxt2pdf -i "c:\input\*.txt" -o d:\output -margin 20
```

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

```
ftxt2pdf -i c:\input\1.txt -o d:\output\1.pdf -margin 20 10
ftxt2pdf -i c:\input -o d:\output -margin 20 10
ftxt2pdf -i test -o output -margin 20 10
ftxt2pdf -i "c:\input\*.txt" -o d:\output -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)

```
ftxt2pdf -i c:\input\1.txt -o d:\output\1.pdf -margin 10 10 30
ftxt2pdf -i c:\input -o d:\output -margin 10 10 30
ftxt2pdf -i test -o output -margin 10 10 30
ftxt2pdf -i "c:\input\*.txt" -o d:\output -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)

```
ftxt2pdf -i c:\input\1.txt -o d:\output\1.pdf -margin 10 10 30 20
ftxt2pdf -i c:\input -o d:\output -margin 10 10 30 20
ftxt2pdf -i test -o output -margin 10 10 30 20
ftxt2pdf -i "c:\input\*.txt" -o d:\output -margin 10 10 30 20
```

c) Support pagination (-breakpage)

- The optional argument (**-breakpage**) indicates the output PDF file will be paginated wherever there is a page break.

  Note *A page break in text files is encoded with the Form Feed (new page) ASCII character (12 (0xC in hexadecimal)).*

Usage Example

1) Support pagination wherever there is a page break (-breakpage)

```
ftxt2pdf -i c:\input\1.txt -o d:\output\1.pdf -breakpage
ftxt2pdf -i c:\input -o d:\output -breakpage
ftxt2pdf -i test -o output -breakpage
ftxt2pdf -i "c:\input\*.txt" -o d:\output -breakpage
```

d) Line spacing setting (-linespace)

- The optional argument (**-linespace**) is used to set the line spacing of the output PDF file. The default value is 0, and the allowable range is from 0 to 72pt.

Usage Example

1) Set the line spacing to 12pt (-linespace 12)

```
ftxt2pdf -i c:\input\1.txt -o d:\output\1.pdf -linespace 12
```

```
ftxt2pdf -i c:\input -o d:\output -linespace 12
ftxt2pdf -i test -o output -linespace 12
ftxt2pdf -i "c:\input\*.txt" -o d:\output -linespace 12
```

e) Password setting (-sp)

- The optional argument (**-sp**) is used to set the open password for 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)

```
ftxt2pdf -i c:\input\1.txt -o d:\output\1.pdf -sp welcome
ftxt2pdf -i c:\input -o d:\output -sp welcome
ftxt2pdf -i test -o output -sp welcome
ftxt2pdf -i "c:\input\*.txt" -o d:\output -sp welcome
```

3.10.3.3 Font Settings (-font, -fs, -fontcolor)

a) Text font (-font)

- The optional arguments (**-font**) is used to set the font style of the text in the output PDF files.

  Note *The font style you set should be installed on a local environment, otherwise the default font style will be used.*

Usage Example

1) Set the font style to "Calibri" (-font "Calibri")

```
ftxt2pdf -i c:\input\1.txt -o d:\output\1.pdf -font "Calibri"
ftxt2pdf -i c:\input -o d:\output -font "Calibri"
ftxt2pdf -i test -o output -font "Calibri"
ftxt2pdf -i "c:\input\*.txt" -o d:\output -font "Calibri"
```

b) Text font size (-fs)

- The optional argument (**-fs**) is used to set the font size of the text in the output PDF files. The default value is set to 9pt, and the allowable range is from 8 to 72pt.

Usage Example

1) Set the font size to 12pt (-fs 12)

```
ftxt2pdf -i c:\input\1.txt -o d:\output\1.pdf -fs 12
ftxt2pdf -i c:\input -o d:\output -fs 12
ftxt2pdf -i test -o output -fs 12
ftxt2pdf -i "c:\input\*.txt" -o d:\output -fs 12
```

c) Text font color (-fontcolor)

- The optional argument (**-fontcolor**) is used to set the font color of the text in the output PDF files. By default, the font color is black. The allowable range of the values for each RGB component is from 0 to 255.

Usage Example

1) Set the font color to blue (-fontcolor 0 0 255)

```
ftxt2pdf -i c:\input\1.txt -o d:\output\1.pdf -fontcolor 0 0 255
ftxt2pdf -i c:\input -o d:\output -fontcolor 0 0 255
ftxt2pdf -i test -o output -fontcolor 0 0 255
ftxt2pdf -i "c:\input\*.txt" -o d:\output -fontcolor 0 0 255
```

3.10.3.4 Document Metadata Settings (-title, -subject, -keywords, -author, -creator)

a) Title (-title)

- The optional argument (**-title**) is used to set title of PDF files.

Usage Example

1) Set document title to "Foxit PDF Toolkit User Manual" (-title "Foxit PDF Toolkit User Manual")

```
ftxt2pdf -i c:\input\1.txt -o d:\output\1.pdf -title "Foxit PDF Toolkit User Manual"
ftxt2pdf -i c:\input -o d:\output -title "Foxit PDF Toolkit User Manual"
ftxt2pdf -i test -o output -title "Foxit PDF Toolkit User Manual"
ftxt2pdf -i "c:\input\*.txt" -o d:\output -title "Foxit PDF Toolkit User Manual"
```

b) Subject (-subject)

- The optional argument (**-subject**) is used to set subject of PDF files.

Usage Example

1) Set document subject to "Foxit PDF Toolkit" (-subject "Foxit PDF Toolkit")

```
ftxt2pdf -i c:\input\1.txt -o d:\output\1.pdf -subject "Foxit PDF Toolkit"
ftxt2pdf -i c:\input -o d:\output -subject "Foxit PDF Toolkit"
ftxt2pdf -i test -o output -subject "Foxit PDF Toolkit"
ftxt2pdf -i "c:\input\*.txt" -o d:\output -subject "Foxit PDF Toolkit"
```

c) Keywords (-keywords)

- The optional argument (**-keywords**) is used to set keywords of PDF files.

Usage Example

1) Set document keywords to "Foxit" (-keywords "Foxit")

```
ftxt2pdf -i c:\input\1.txt -o d:\output\1.pdf -keywords "Foxit"
ftxt2pdf -i c:\input -o d:\output -keywords "Foxit"
ftxt2pdf -i test -o output -keywords "Foxit"
ftxt2pdf -i "c:\input\*.txt" -o d:\output -keywords "Foxit"
```

d) Author (-author)

- The optional argument (**-author**) is used to set an author of PDF files.

Usage Example

1) Set document author to "Jessie" (-author "Jessie")

```
ftxt2pdf -i c:\input\1.txt -o d:\output\1.pdf -author "Jessie"
ftxt2pdf -i c:\input -o d:\output -author "Jessie"
ftxt2pdf -i test -o output -author "Jessie"
ftxt2pdf -i "c:\input\*.txt" -o d:\output -author "Jessie"
```

e) Creator (-creator)

- The optional argument (**-creator**) is used to set file creation application information of PDF files.

Usage Example

1) Set document creator to "Foxit Reader" (-creator "Foxit Reader")

```
ftxt2pdf -i c:\input\1.txt -o d:\output\1.pdf -creator "Foxit Reader"
ftxt2pdf -i c:\input -o d:\output -creator "Foxit Reader"
ftxt2pdf -i test -o output -creator "Foxit Reader"
ftxt2pdf -i "c:\input\*.txt" -o d:\output -creator "Foxit Reader"
```

3.10.3.5 Recursion Depth of Sub-folders (-r)

- The optional argument (**-r**) indicates the recursion depth of sub-folders, which is valid only when the input is a folder or a path that includes wildcard characters like "c:\input*.txt". By default, the recursion depth is 1, so the sub-folders will not be processed. For more details about this argument, please refer to section 3.10.2 "Command Line Summary".

Usage Examples

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

```
ftxt2pdf -i c:\input -o d:\output -r
ftxt2pdf -i test -o output -r
ftxt2pdf -i "c:\input\*.txt" -o d:\output -r
ftxt2pdf -i c:\input -o d:\output -r 0
ftxt2pdf -i test -o output -r 0
ftxt2pdf -i "c:\input\*.txt" -o d:\output -r 0
```

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

```
ftxt2pdf -i c:\input -o d:\output -r 1
ftxt2pdf -i test -o output -r 1
ftxt2pdf -i "c:\input\*.txt" -o d:\output -r 1
```

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

```
ftxt2pdf -i c:\input -o d:\output
ftxt2pdf -i test -o output
ftxt2pdf -i "c:\input\*.txt" -o d:\output
```

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

```
ftxt2pdf -i c:\input -o d:\output -r 2
ftxt2pdf -i test -o output -r 2
ftxt2pdf -i "c:\input\*.txt" -o d:\output -r 2
```

3.10.3.6 Multi-thread Support (-t)

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

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

Usage Example

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

```
ftxt2pdf -i c:\input -o d:\output -t 3
ftxt2pdf -i test -o output -t 3
ftxt2pdf -i "c:\input\*.txt" -o d:\output -t 3
```

3.10.3.7 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 argument (**-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 3.10.2 "Command Line Summary".

Usage Example

1) Save the log file to "d:\output\text2pdf.log" and set the log level to 3 (-log d:\output\text2pdf.log -l 3)

```
ftxt2pdf -i c:\input -o d:\output -log d:\output\text2pdf.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 the **<licensee>** is the licensee name designated by the users.

Usage Example

1) Register the text2pdf tool with the code "xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx" and the licensee "Foxit" (-register xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx Foixt)

ftxt2pdf -register xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx Foixt

c) License agreement (-license)

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

Usage Example

1) Print the license agreement (-license)

ftxt2pdf -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)

ftxt2pdf -version
ftxt2pdf -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)

ftxt2pdf -help
ftxt2pdf -h

f) Copyright information (-copyright)

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

Usage Example

1) Print the copyright information (-copyright)

```
ftxt2pdf -copyright
```

3.11 Html2PDF

3.11.1 Basic Syntax

fhtml2pdf *<-i <srcfile/url/srcfolder>> <-o <destfile/destfolder>> [-width <PDF width>] [-height <PDF height>] [-margin <left [top right bottom]>] [-cache <cache folder>] [-timeout <load timeout>] [-singlepage] [-nolink] [-rotate <0/90/180/270>] [-checklazyload] [-cookies <cookie file>] [-js <JavaScript file>] [-r [recursion]] [-t <thread>] [-log <logfile>] [-l <level>]*

fhtml2pdf *-register <code> <licensee>*

fhtml2pdf *-license*

fhtml2pdf *-version/-v*

fhtml2pdf *-help/-h*

fhtml2pdf *-copyright*

Note:

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

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

3.11.2 Command Line Summary

Note *For some arguments whose values are strings, users can choose whether to add quotation marks (" ") to the strings. In the manual we have added notes where adding quotation marks (" ") is required.*

Option	Parameter	Description
-i	*<-i <string>>* *e.g.* *-i "www.foxitsoftware.com"* *-i c:\input\1.htm* *-i c:\input* *-i "c:\input*.html"*	Specifies the input file to be converted. ▪ The input string can be the name of a single HTML file or a folder, or a URL. ▪ The file name can contain the wildcard character (*). For example, use *.html to include all HTML files in a given folder. **Note** Wildcard characters (*.*) is currently not

Option	Parameter	Description
		supported.
-o	*<-o <string>>* *e.g.* *-o d:\output\1..pdf* *-o d:\output*	Specifies the path of the output PDF file or folder. **Note** ▪ The specified output path must already exist. ▪ If the input is a folder, the output should be a folder, (e.g. *-i c:\input -o d:\output*).
-width	*[-width <points>]* *e.g.* *-width 612*	Sets the page width of the output PDF file in points. Default width value: 842 points. Allowable range: 16-14400 points.
-height	*[-height <points>]* *e.g.* *-height 792*	Sets the page height of the output PDF file in points. Default height value: 595 points. Allowable range: 16-14400 points.
-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 margin values are 10 10 10 10. Allowable values: 0-size of page in points; in addition, the sum of the left and right values should be less than the width of the page, and the sum of the top and bottom values should be less than the height of the page. ***-margin left top right bottom*** **-margin 20**: sets the left margin to 20 points. **-margin 10 20**: sets the left margin to 10 points and the top margin to 20 points. **-margin 10 20 0**: sets the left margin to 10 points, the top margin to 20 points, and the right margin to 0 points. **-margin 10 20 0 40**: sets 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.
-cache	*[-cache <string>]* *-cache <folder>* *e.g.* *-cache "d:\resources"*	Sets cache address to store HTML page resources temporarily. The page resources of the converted webpage will be downloaded and stored to this cache address

Option	Parameter	Description
		first, and then will be deleted after conversion. **Note** If this argument is not set, a folder named "cache" will be generated in the installation folder.
-timeout	*[-timeout <integer>]* *-timeout <loading timeout>* *e.g.* *-timeout 30*	Sets timeout in seconds to load webpages. Default value is 120s. The webpages will not continue to be loaded when the time is used up. **Note** The **-timeout** argument should be set to a value greater than 15. If users set a value less than 15, the timeout value will be set to 15.
-singlepage	*[-singlepage]* *e.g.* *-singlepage*	Sets all the page contents to one single PDF page.
-nolink	*[-nolink]* *e.g.* *-nolink*	Converts the input to PDF files with no link annotations retained.
-rotate	*[-rotate <0/90/180/270>]* *e.g.* *-rotate 0* *-rotate 90* *-rotate 180* *-rotate 270*	Sets page rotation for output PDF files. The value should be 0, 90, 180 or 270, and the default value is 0.
-checklazyload	*[-checklazyload]* *e.g.* *-checklazyload*	Improves the conversion quality if the webpages include lazy loading elements or if the network/hardware performance is not good enough. The tool will spend at least 5 seconds waiting for loading the web elements before starting conversion. **Note** The **-checklazyload** argument is useful in the following two situations: ▪ Some special long webpages use lazy loading design pattern to make the page load faster and reduce server load.

Option	Parameter	Description
		▪ The performance of the network or the hardware you use is not good enough.
-cookies	*[-cookies <string>]* *e.g.* *-cookies cookie.txt*	Specifies the path of the cookies file which stores the authorization information of a URL that you want to convert.
-js	*[-js <string>]* *e.g.* *-js login.js*	Specifies the path of the JavaScript file which stores some JS scripts that will be applied to webpages to handle some actions, such as login simulation, closing popup window, scrolling to obtain the asynchronous data and more.
-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. • **-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** ▪ If no integer value is specified, or if the integer value is 0, then full folders will be searched. By default, the number of layers to recurse is 1, which means that only the current folder will be searched and not sub-folders. ▪ The input folder will be skipped if it is secured and the messages will be displayed.
-t	*[-t <integer>]* *e.g.* *-t 1* *-t 2* …	Specifies the number of CPU threads to use. The default value is 1.
-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 messages only concerning out of memory errors. • **-l 2**: logs failure messages concerning the errors caused during execution or those returned from underlying libraries, as well as those for level 1. • **-l 3**: logs warning messages concerning the PDF files that are overwritten, as well as those for level 2. • **-l 4**: logs informational messages, as well as those for level 3. **Note** The argument (**-l**) is valid only when (**-log**) is used.
-register	*[-register <String> <String>]* *-register <code> <licensee>* *e.g.* *-register xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx Foxit*	Registers the command line tool. ▪ **<code>**: the activation code from Foxit. ▪ **<licensee>**: the Licensee name designated by the users.
-help/-h	*[-help]/[-h]* *e.g.* *-help* *-h*	Prints the usage information.
-version/-v	*[-version]/[-v]* *e.g.* *-version* *-v*	Prints the version information.
-license	*[-license]* *e.g.* *-license*	Prints the license agreement.

3.11.3 Basic Usage

3.11.3.1 Input and Output (-i, -o)

a) Input (-i)

- The input should be a single HTML file, a URL, or a folder. Users are not able to input multiple HTML files (or URLs) or folders, as well as a mixture composed of folders and HTML files (or URLs). For example:

  -i c:\input\1.html (a single HTML file)
  -i "www.foxitsoftware.com" (a URL)
  -i c:\input (a single folder)

  Note *If the input is a URL, it is recommended to enclose it with quotation marks (" "). In particular, if the string of the URL contains special characters, the quotation marks are required. In this manual, we add (" ") whenever the input is a URL.*

- It supports relative paths if the input file is in the current working folder. Users can input just the name of the HTML file or folder, instead of an absolute path. For example:

  -i test\2.htm ("test" folder 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 HTML files in specified formats. For example:

  -i "c:\input*.html" (Only convert HTML with HTML format)
  -i "test*.htm" (Only convert HTML with HTM format)

  Note *When using wildcard characters in the input files, it is recommended to enclose the input files with quotation marks (" "). In this manual, we all add (" ") when the input files use wildcard characters.*

Note *If the input paths or file names contain spaces, you must enclose the input files with quotation marks (" "), such as -i "c:\input file\1.html", -i "test\user manual.html". This rule is also used for some other arguments whose values are strings.*

b) Output (-o)

- If the input is a single HTML file or a URL, you should specify the output folder, including the output file if needed, while if the input is a single folder, you can only specify the output folder. For example:

  -i c:\input\1.html -o d:\output\ 1.pdf
  -i "www.foxitsoftware.com" -o d:\output
  -i c:\input -o d:\output

Note

- *The specified output path must already exist.*
- *If you specify the output folder without the output file when the input is a single HTML file, the output PDF will be named with the source file's name by default.*
- *If you specify the output folder without the output file when the input is a URL, the output PDF will be named with the title of the URL by default.*

- The output also supports relative paths if the specified output location is in the current working folder. Users can input just 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 a HTML file or a URL into a PDF file:

```
fhtml2pdf -i c:\input\1.html -o d:\output\1.pdf
fhtml2pdf -i test\2.html -o output
fhtml2pdf -i "www.foxitsoftware.com" -o output\foxit.pdf
```

2) Convert HTML files in a folder into PDF files:

```
fhtml2pdf -i c:\input -o d:\output
fhtml2pdf -i test -o output
```

3.11.3.2 PDF/Page Settings (-width, -height, -margin, -singlepage, -nolink, -rotate)

a) 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. The default width and height are 842 and 595 points respectively with the allowable range of 16-14400 points.

Usage Example

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

```
fhtml2pdf -i c:\input -o d:\output -width 400 -height 300
fhtml2pdf -i test\1.html -o d:\output\1.pdf -width 400 -height 300
fhtml2pdf -i "c:\input\*.html" -o d:\output -width 400 -height 300
fhtml2pdf -i "www.foxitsoftware.com" -o output -width 400 -height 300
```

b) Margin setting (-margin)

- The optional argument (-margin) is used to set size of margin for each PDF page in points. The default margin values are 10 10 10 10. For more details about this argument, please refer to section 3.11.2 "Command Line Summary".

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

Usage Example

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

```
fhtml2pdf -i c:\input -o d:\output -margin 20
fhtml2pdf -i test\1.html -o d:\output\1.pdf -margin 20
fhtml2pdf -i "c:\input\*.html" -o d:\output -margin 20
fhtml2pdf -i "www.foxitsoftware.com" -o output -margin 20
```

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

```
fhtml2pdf -i c:\input -o d:\output -margin 20 10
fhtml2pdf -i test\1.html -o d:\output\1.pdf -margin 20 10
fhtml2pdf -i "c:\input\*.html" -o d:\output -margin 20 10
fhtml2pdf -i "www.foxitsoftware.com" -o output -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)

```
fhtml2pdf -i c:\input -o d:\output -margin 10 10 30
fhtml2pdf -i test\1.html -o d:\output\1.pdf -margin 10 10 30
fhtml2pdf -i "c:\input\*.html" -o d:\output -margin 10 10 30
fhtml2pdf -i "www.foxitsoftware.com" -o output -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)

```
fhtml2pdf -i c:\input -o d:\output -margin 10 10 30 20
fhtml2pdf -i test\1.html -o d:\output\1.pdf -margin 10 10 30 20
fhtml2pdf -i "c:\input\*.html" -o d:\output -margin 10 10 30 20
fhtml2pdf -i "www.foxitsoftware.com" -o output -margin 10 10 30 20
```

c) Single page (-singlepage)

- The optional argument (**-singlepage**) is used to set all the page contents to one single PDF page.

Usage Example

1) Set all the page contents to one single PDF page (-singlepage)

```
fhtml2pdf -i c:\input -o d:\output -singlepage
fhtml2pdf -i test\1.html -o d:\output\1.pdf -singlepage
fhtml2pdf -i "c:\input\*.html" -o d:\output -singlepage
fhtml2pdf -i "www.foxitsoftware.com" -o output -singlepage
```

d) Disable retaining hyperlinks (-nolink)

- The optional argument (**-nolink**) is used to convert the input to PDF files with no link annotations retained. If users set this argument, no action will be triggered when they click the links in the output PDF file.

Usage Example

1) Convert the input to PDF files with no link annotations (-nolink)

```
fhtml2pdf -i c:\input -o d:\output -nolink
fhtml2pdf -i test\1.html -o d:\output\1.pdf -nolink
fhtml2pdf -i "c:\input\*.html" -o d:\output -nolink
fhtml2pdf -i "www.foxitsoftware.com" -o output -nolink
```

e) Page rotation (-rotate)

- The optional argument (**-rotate**) is used to set page rotation for the output PDF files. The setting value should be 0, 90, 180 or 270 and the default value is 0.

Usage Example

1) Set page rotation to 90 degree (-rotate 90)

```
fhtml2pdf -i c:\input -o d:\output -rotate 90
fhtml2pdf -i test\1.html -o d:\output\1.pdf -rotate 90
fhtml2pdf -i "c:\input\*.html" -o d:\output -rotate 90
fhtml2pdf -i "www.foxitsoftware.com" -o output -rotate 90
```

3.11.3.3 Cache Address Setting (-cache)

- The optional argument (**-cache**) is used to set cache address to store HTML page resources temporarily. The page resources of the converted webpage will be downloaded and stored to this cache address first, and then will be deleted after conversion. If this argument is not set, a folder named "cache" will be generated in the installation folder.

Usage Example

1) Set cache address to "*d:\resources*" (-cache "d:\resources")

```
fhtml2pdf -i c:\input -o d:\output -cache "d:\resources"
fhtml2pdf -i test\1.html -o d:\output\1.pdf -cache "d:\resources"
fhtml2pdf -i "c:\input\*.html" -o d:\output -cache "d:\resources"
fhtml2pdf -i "www.foxitsoftware.com" -o output -cache "d:\resources"
```

3.11.3.4 Timeout Setting (-timeout)

- The optional argument (**-timeout**) is used to set timeout in seconds to load webpages. The webpages will not continue to be loaded when the time is used up. The default value is 120s, and the timeout value will be set to 15 if users set a value less than 15.

Usage Example

1) Set the timeout to "200s" to load a webpage (-timeout 200)

```
fhtml2pdf -i c:\input -o d:\output -timeout 200
fhtml2pdf -i test\1.html -o d:\output\1.pdf -timeout 200
fhtml2pdf -i "c:\input\*.html" -o d:\output -timeout 200
fhtml2pdf -i "www.foxitsoftware.com" -o output -timeout 200
```

3.11.3.5 Check lazy load (-checklazyload)

- The optional argument (**-checklazyload**) is used to improve the conversion quality if the webpages include lazy loading elements or if the network/hardware performance is not good enough.

  Note *The **-checklazyload** argument is useful in the following two situations:*

  - *Some special long webpages use lazy loading design pattern to make the page load faster and reduce server load, therefore some web elements are designed to be delayed loading, which will affect the conversion quality.*
  - *If the network or hardware performance is not good enough, the web elements loading will be influenced, which will also affect the conversion quality.*

  If this argument is set, the tool will spend at least 5 seconds waiting for loading the web elements before starting conversion, which can help improve the conversion quality.

Usage Example

1) Improve the conversion quality if the webpages include lazy loading elements or if the network/hardware performance is not good enough (-checklazyload)

```
fhtml2pdf -i c:\input -o d:\output -checklazyload
fhtml2pdf -i test\1.html -o d:\output\1.pdf -checklazyload
fhtml2pdf -i "c:\input\*.html" -o d:\output -checklazyload
fhtml2pdf -i "www.foxitsoftware.com" -o output -checklazyload
```

3.11.3.6 Cookie File Setting (-cookies)

- The optional argument (**-cookies**) is used to specify the path of the cookies file which stores the authorization information of a URL that you want to convert.

  The **-cookies** argument is useful for the online webpages (except for some financial websites) that require authorization information, such as username, user password, and some other verification code which are required when you sign in the website. To convert such webpages, users should sign in the website first, export the cookie file from the webpage, and then do the conversion with the exported cookie file.

  For example, for an e-book store, one user wants to convert the favorite page into PDF. It is evident that the user should sign in the e-book store first, then he/she can browse his/her favorite. That means the favorite page requires authorization information, so the user should export the cookie file

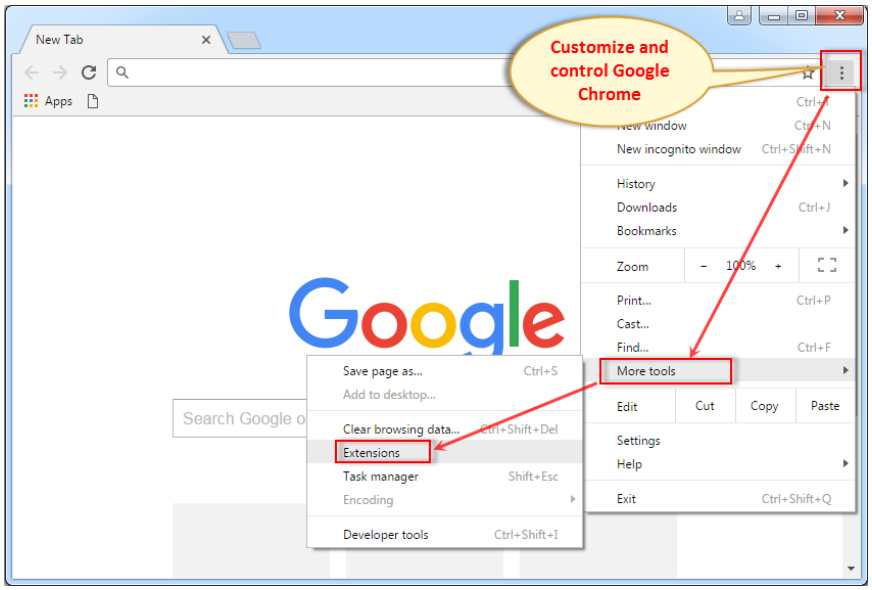
from the favorite page, and then convert the page to PDF using the "-cookies" argument with the exported cookie file. Otherwise, maybe only the login page will be converted into PDF.

How to export the cookie file from a login webpage?

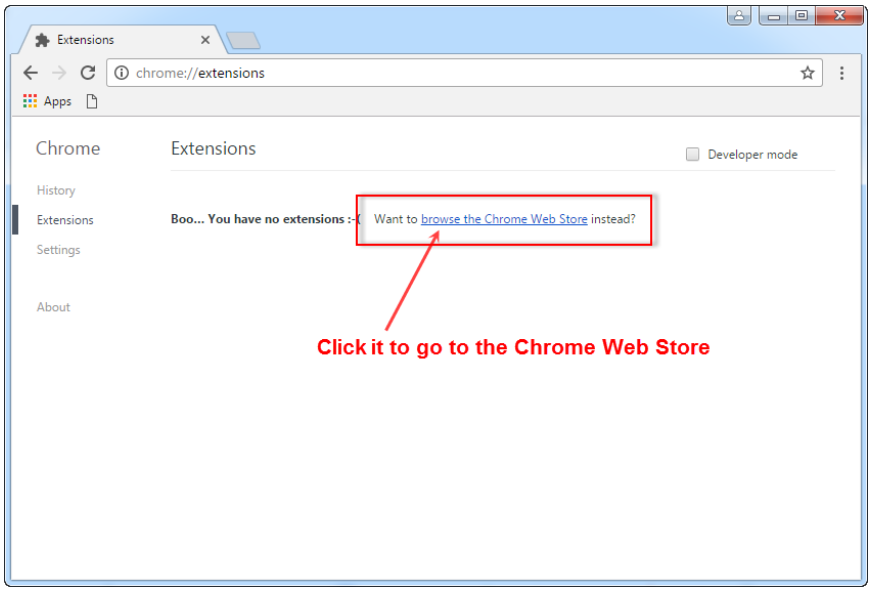
Most browsers have already provided a plugin called "Edit This Cookie". It is a cookie manager that allows you to add, delete, edit, search, protect and block cookies. In this manual, it takes Chrome browser as an example to show you how to install the plugin and how to export the cookie file from a login webpage.

To install the plugin, follow the steps below:

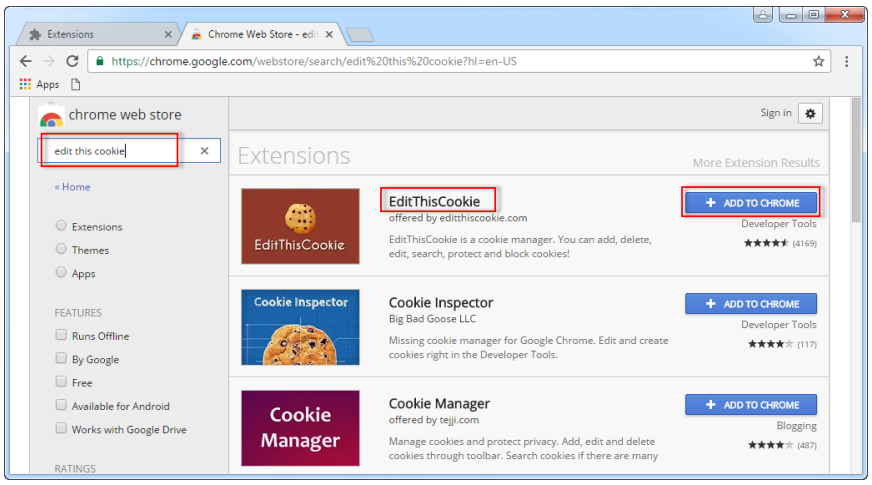
a) Open Google Chrome browser, click the ⋮ (Customize and control Google Chrome) button, and then select **More tools -> Extensions** as shown in the following figure.



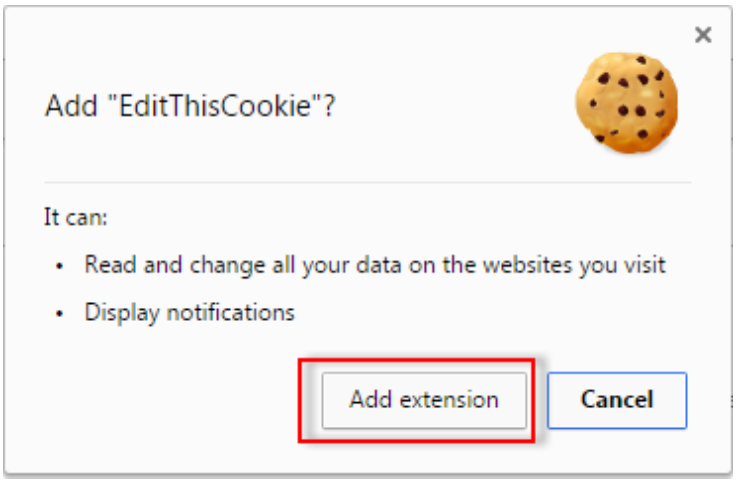
b) Click on **browse the Chrome Web Store** to go to the extensions store as follows.



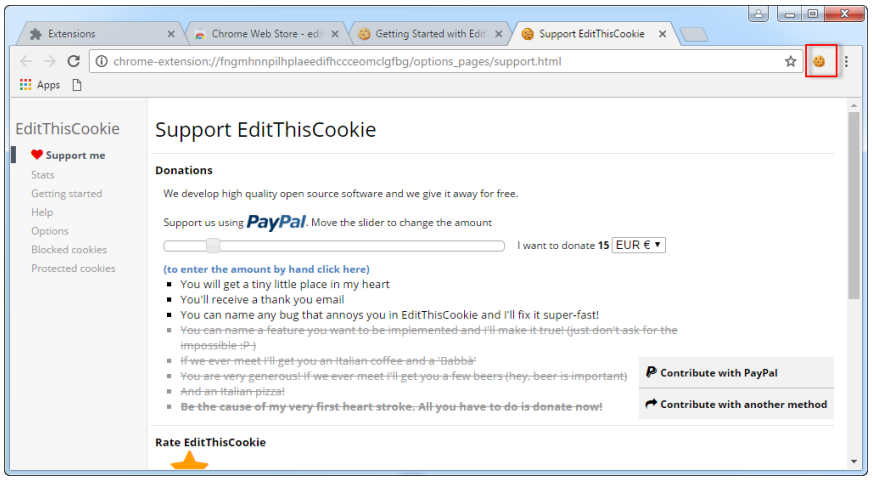
c) In the store, type the "edit this cookie" to search for the plugin, and then add it to Chrome as shown in the following figure.



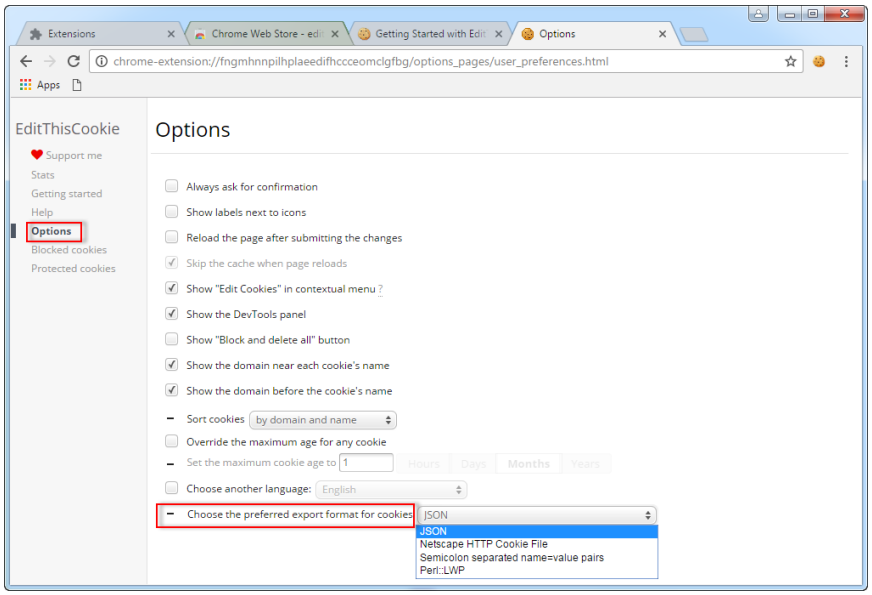
d) A window will popup asking you whether to add this extension. Click **Add extension**.



e) After adding, the plugin will appear on the top toolbar as shown in the following figure.



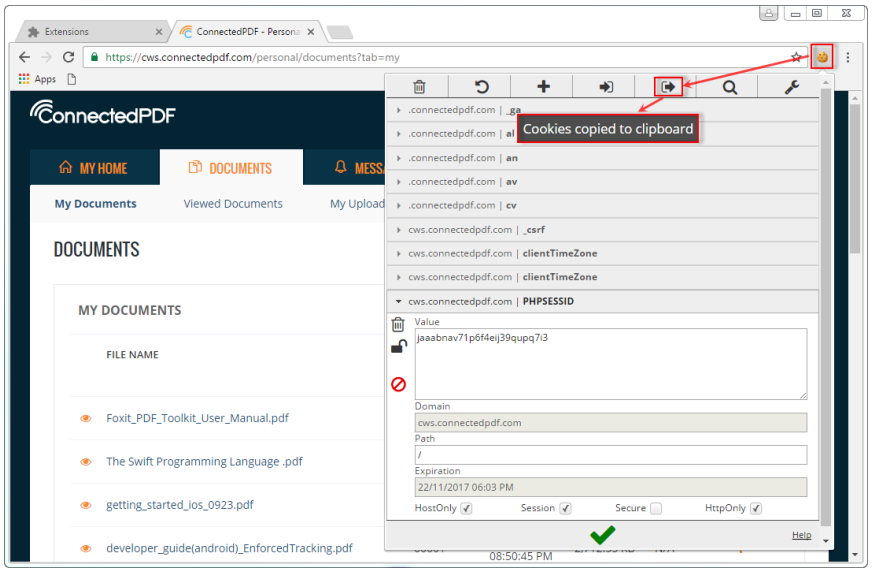
Note *Foxit Html2PDF currently supports three types of export format for cookies:* ***JSON****,* ***Semicolon separated name=value pairs*** *and* ***Perl::LWP****. The default and recommended format is JSON. If you want to change the format, click on* ***Options****, find* ***choose the preferred export format for cookies****, and then choose the format you wish as shown in the following figure.*



To export the cookie file from a login webpage, follow the steps below:

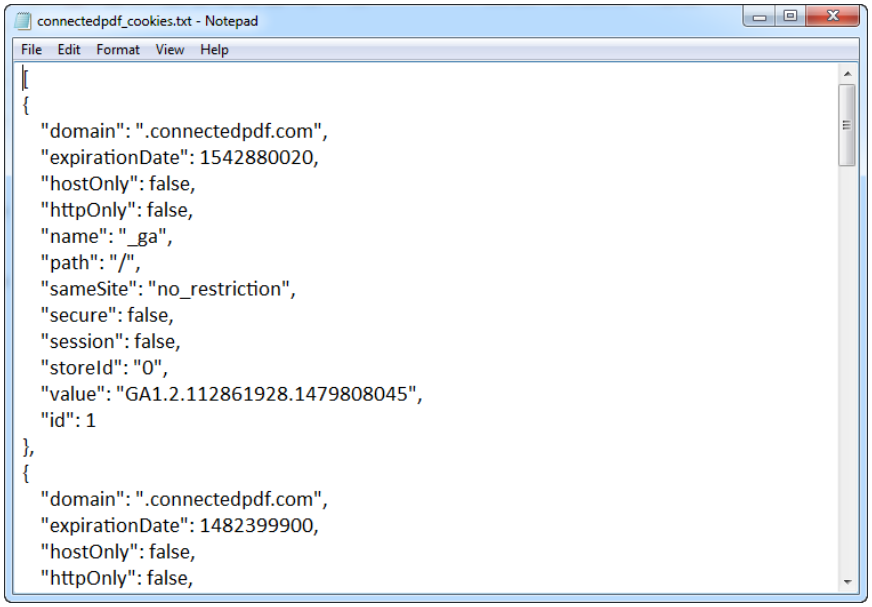
a) Open the webpage you want to convert, making sure the webpage requires authorization information. In this manual, it takes Foxit ConnectedPDF personal document page as an example: https://cws.connectedpdf.com/personal/documents?tab=my (I have already signed in this website).

b) Click the plug in the top toolbar, and then choose the export button as shown in the following figure. After clicking the export button, it will prompt that cookies copied to clipboard.



c) Create a txt file and paste the cookies into the txt file. For example, in the installation folder, create a txt file called connectedpdf_cookies.txt, and then paste the exported cookies into it.

Some contents of the cookies are displayed in the following figure with JSON format. Now, the cookies for the Foxit ConnectedPDF personal document page have been exported successfully.



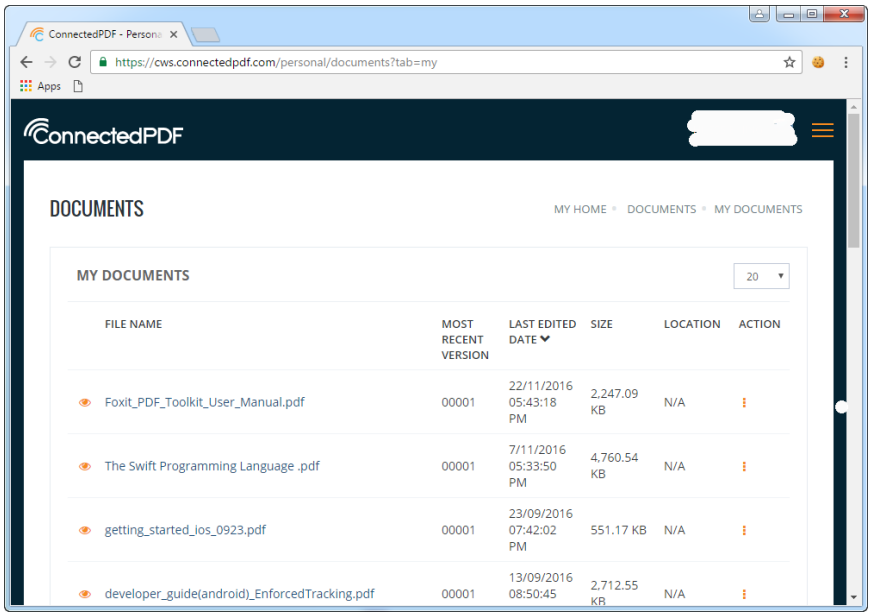
Usage Example

1) Convert a URL that requires login into PDF file (-cookies connectedpdf_cookies.txt)

fhtml2pdf -i "https://cws.connectedpdf.com/personal/documents?tab=my" -o d:\output -cookies connectedpdf_cookies.txt

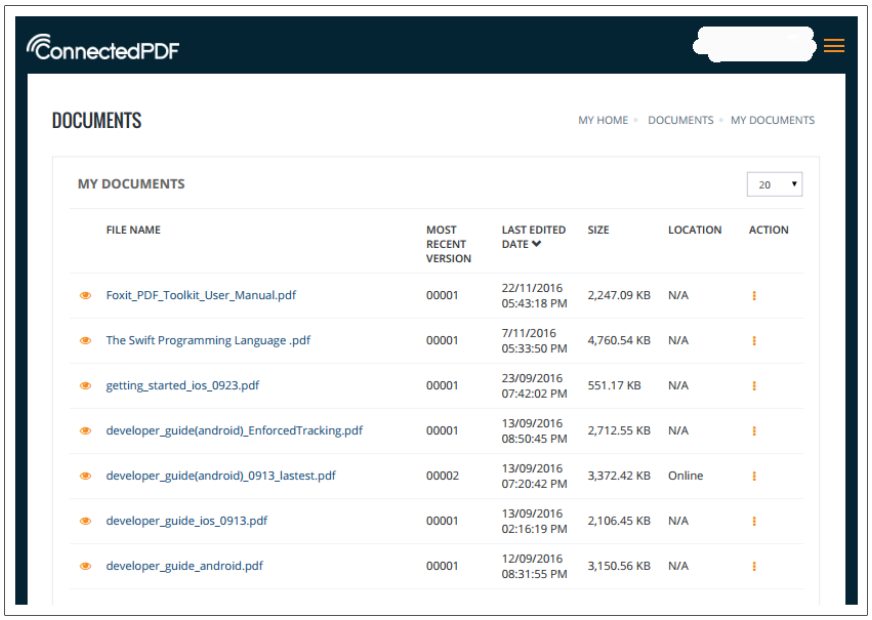
Conversion Result

The Foxit ConnectedPDF personal document page that I want to convert: (only capture a part of the page)



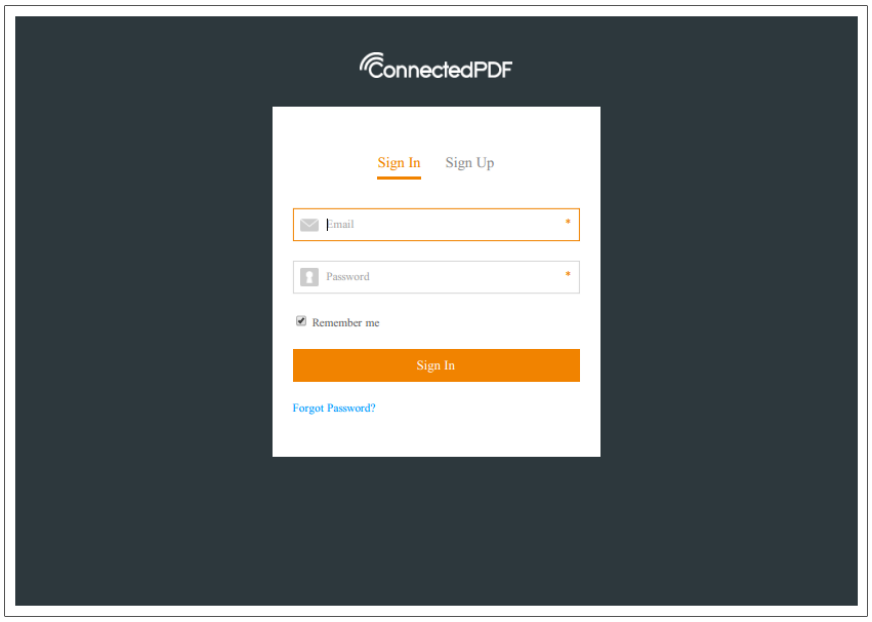
Convert the above URL into PDF file with "-cookies" argument: (only capture the first page of the PDF file)

fhtml2pdf -i "https://cws.connectedpdf.com/personal/documents?tab=my" -o d:\output -cookies connectedpdf_cookies.txt



Convert the above URL into PDF file without "-cookies" argument:

fhtml2pdf -i "https://cws.connectedpdf.com/personal/documents?tab=my" -o d:\output



3.11.3.7 JavaScript File Setting (-js)

- The optional argument (**-js**) is used to specify the path of the JavaScript file which stores some JS scripts that will be applied to webpages to handle some actions, such as login simulation, closing popup window, scrolling to obtain the asynchronous data and more.

  Note

  - *It supports native DOM/BOM API, and currently can only support ECMAScript 5.1.*
  - *For login simulation, it is only valid for the websites that are not related to finance and do not need verification code.*

Usage Example

1) Apply JavaScript to webpages (-js login.js)

```
fhtml2pdf -i c:\input -o d:\output -js login.js
fhtml2pdf -i test\1.html -o d:\output\1.pdf -js login.js
fhtml2pdf -i "c:\input\*.html" -o d:\output -js login.js
fhtml2pdf -i "www.foxitsoftware.com" -o output -js login.js
```

In the "samples\javascript" folder of the installation path, Foxit Html2PDF provides several JavaScript samples which includes the actions about login simulation, closing popup window and scrolling to obtain the asynchronous data. You can refer to them to write the JavaScript that you want.

3.11.3.8 Recursion Depth of Sub-folders (-r)

- The optional argument (**-r**) indicates the recursion depth of sub-folders, which is valid only when the input is a folder or a path that includes wildcard character like "c:\input*.html". By default, the recursion depth is 1, so the sub-folders will not be processed. For more details about this argument, please refer to section 3.11.2 "Command Line Summary".

Usage Examples

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

```
fhtml2pdf -i test -o output -r
fhtml2pdf -i c:\input -o d:\output -r
fhtml2pdf -i "c:\input\*.html" -o d:\output -r
fhtml2pdf -i test -o output -r 0
fhtml2pdf -i c:\input -o d:\output -r 0
fhtml2pdf -i "c:\input\*.html" -o d:\output -r 0
```

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

```
fhtml2pdf -i test -o output -r 1
fhtml2pdf -i c:\input -o d:\output -r 1
fhtml2pdf -i "c:\input\*.html" -o d:\output -r 1
```

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

```
fhtml2pdf -i test -o output
fhtml2pdf -i c:\input -o d:\output
fhtml2pdf -i "c:\input\*.html" -o d:\output
```

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

```
fhtml2pdf -i test -o output -r 2
fhtml2pdf -i c:\input -o d:\output -r 2
fhtml2pdf -i "c:\input\*.html" -o d:\output -r 2
```

3.11.3.9 Multi-thread Support (-t)

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

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

Usage Example

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

```
fhtml2pdf -i test -o output -t 3
fhtml2pdf -i c:\input -o d:\output -t 3
fhtml2pdf -i "c:\input\*.html" -o d:\output -t 3
```

3.11.3.10 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 argument (**-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 3.11.2 "Command Line Summary".

Usage Example

1) Save the log file to "d:\output\html2pdf.log" and set the log level to 3 (-log d:\output\html2pdf.log -l 3)

```
fhtml2pdf -i c:\input -o d:\output -log d:\output\html2pdf.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 the users.

Usage Example

1) Register the html2pdf tool with the code "xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx" and the licensee "Foxit" (-register xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx Foixt)

```
fhtml2pdf -register xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx Foixt
```

c) License agreement (-license)

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

Usage Example

1) Print the license agreement (-license)

fhtml2pdf -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)

fhtml2pdf -version
fhtml2pdf -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)

fhtml2pdf -help
fhtml2pdf –h

f) Copyright information (-copyright)

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

Usage Example

1) Print the copyright information (-copyright)

fhtml2pdf -copyright

3.12 PDF Page Organizer

3.12.1 Basic Syntax

fpdfppo *<-i <srcfile/srcfolder>[,pagelist]> <-o <destfile/destfolder>>*
[-mode <mode type>] [-maxpage <max page>] [-deletebm]
[-op <password>] [-r [recursion]] [-t <thread>] [-log <logfile>] [-l <level>]
fpdfppo *-register <code> <licensee>*
fpdfppo *-license*
fpdfppo *-version/-v*
fpdfppo *-help/-h*
fpdfppo *-copyright*

Note:

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

Only the *<-i <srcfile/srcfolder>[,pagelist*]> and <*-o <destfile/destfolder>>* 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. Full details on each are explained in the following section.

3.12.2 Command Line Summary

Note *For some arguments whose values are strings, users can choose whether to add quotation marks (" ") to the strings. In the manual we have added notes where adding quotation marks (" ") is required.*

Option	Parameter	Description
-i	*<-i <string>>* *e.g.* *-i c:\input\1.pdf* *-i c:\input* *-i "1.pdf" "2.pdf"* *-i "1.pdf,2,5,9"* *-i "1.pdf,odd" "2.pdf,even"* *-i "test,1-5,20-"* *-i "c:\input*.pdf"* *-i "c:\input*.pdf,2,5,odd"*	Specifies the input file to be processed. ▪ The input string can be the name of a single PDF file, multiple PDF files or a folder with a page list or not. ▪ The file name can contain the wildcard character (*). For example, use *.pdf to include all PDF files in a given folder. **Note** Wildcard character (*.*) is currently not supported.
-o	*<-o <string>>* *e.g.* *-o d:\output\merge.pdf* *-o d:\output* *-o 3.pdf 4.pdf (when the input has two PDF files)*	Specifies the path of the output PDF file or folder. ▪ In Merge mode, the output should be a PDF file. ▪ In Split and Delete modes, the output can be PDF file(s) or a folder. For more details, please refer to "Input and Output" and "Modes of Operation". **Note** The specified output path must already exist.
-mode	*[-mode <integer>]* *e.g.* *-mode 1* *-mode 2* *-mode 3*	Specifies the mode of operation. The default value is 1. • **-mode 1**: Merge mode. Merge multiple PDF files into one PDF file. • **-mode 2**: Split mode. Split PDF file(s) into several PDF files. • **-mode 3**: Delete mode. Delete specific pages from existing PDF file(s).
-maxpage	*[-maxpage <integer>]* *e.g.* *-maxpage 5* *-maxpage 10*	Sets the maximum number of pages for each split PDF file. The number of pages for the last PDF file will be smaller than the maximum number if the total number of the PDF pages is not a multiple of the value specified by "-maxpage". **Note** The argument (**-maxpage**) is valid only when (-**mode**) is set to 2 and the input is not followed by a

Option	Parameter	Description
		page list.
-deletebm	*[-deletebm]* *e.g.* *-deletebm*	Delete the bookmarks of the merged PDF file. **Note** The argument (**-deletebm**) is valid only when the (-mode) is set to 1.
-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 password protected. **Note** The output PDF file will retain the open password from the input file.
-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. • **-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** ▪ If no integer value is specified, or if the integer value is 0, then full folders will be searched. By default, the number of layers to recurse is 1, which means that only the current folder will be searched and not sub-folders. ▪ The input PDF file or folder will be skipped if it is secured and the messages will be displayed.
-t	*[-t <integer>]* *e.g.* *-t 1* *-t 2* …	Specifies the number of CPU threads to use. The default value is 1.
-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.*	Sets the log level. The default is 4. • **-l 1**: logs messages only concerning out of

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

3.12.3 Basic Usage

3.12.3.1 Page List Format

Foxit PDF Page Organizer offers a variety of options for selecting specific pages or page ranges while performing the operations.

- <page number>, selects the specific pages, such as 1,3,5.
- all, selects all pages of PDF file(s).
- even, selects all even pages of PDF file(s).
- odd, select all odd pages of PDF file(s).
- <start page number>-<end page number>, select all the pages in the range from <start page number> to <end page number> of PDF file(s), such as 1-5, 3-10.
- <start page number>-, select all the pages from <start page number> to the last page of PDF file(s), such as 50-, 100-.

Note *The above-mentioned options can be used in a combined way. For example, select pages 2,3, all odd pages, and the pages from 15 to 20, you can specify the page list like "2,3,odd,15-20". Assume the input PDF file has a total of 20 pages and specify the page list "2,3,odd,15-20" in Merge mode, then the output PDF file will have 14 pages including pages 1,2,3,5,7,9,11,13,15,16,17,18,19 and 20.*

3.12.3.2 Input and Output (-i, -o)

a) Input (-i)

- The input should be a single PDF file, multiple PDF files, or a folder. Users are not able to input multiple folders, as well as a mixture composed of folders and PDF files. For example:

-i c:\input\1.pdf	(a single PDF file)
-i "c:\input\1.pdf" "c:\input\2.pdf"	(multiple PDF files)
-i c:\input	(a single folder)

Note *If the input files are multiple PDF files, it is recommended to enclose each input path with quotation marks (" "). In this manual, we add (" ") whenever the input files are multiple image files.*

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

-i test\1.pdf	("test" folder is in the current working folder)
-i "test\1.pdf" "test\2.pdf"	("test" folder 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, it is recommended to enclose the input files with quotation marks (" "). In this manual, we add (" ") whenever the input files contain wildcard characters.*

- The input can be followed by a page list which is used for selecting specific pages or page ranges. The page list can directly follow the input PDF filename or input folder name (use a comma to separate the elements). For example:

Select pages 1, 5, and 9 of "1.pdf"

-i "c:\input\1.pdf,1,5,9"
-i "test\1.pdf,1,5,9"

Select pages 1 and 6 of "1.pdf" and the pages in the range from 2 to 5 of "2.pdf"

-i "c:\input\1.pdf,1,6" "c:\input\2.pdf,2-5"
-i "test\1.pdf,1,6" "test\2.pdf,2-5"

Select all the even pages of the PDF files in a given folder

-i "c:\input,even"
-i "test,even"
-i "c:\input*.pdf,even"
-i "test*.pdf,even"

Select all the odd pages and the pages in the range from 20 to the last page of the PDF files in a given folder

-i "c:\input,odd,20-"
-i "test,odd,20-"
-i "c:\input*.pdf,odd,20-"
-i "test*.pdf,odd,20-"

For more details about how to specify the page list as part of the input file, please refer to the section "Page List Format".

==========

Note *If the input paths or file names contain spaces, you must enclose the input files with quotation marks (" "), such as -i "c:\input file\1.pdf", -i "test\user manual.pdf". This rule is also used for some other arguments whose values are strings.*

b) Output (-o)

- The output should be a single PDF file, multiple PDF files, or a folder. For example:

-o d:\output\3.pdf	(a single PDF file)
-o "d:\output\3.pdf" "d:\output\4.pdf"	(multiple PDF files)
-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. Users can input just the name of the output PDF file or folder, instead of an absolute path. For example:

-o output\3.pdf	("output" folder is in the current working folder)
-o "output\3.pdf" "output\4.pdf"	("output" folder is in the current working folder)
-o output	("output" folder is in the current working folder)

Note *There are some usage examples for every mode in the following section "Modes of Operation".*

3.12.3.3 Modes of Operation (-mode)

- The optional argument (**-mode**) is used to specify the mode of operation. Foxit PDF Page Organizer supports three modes of operation: Merge mode, Split mode, and Delete mode. If this argument is not set, the default mode is Merge mode.

Merge Mode (-mode 1)

In this mode, usually the **input** should be multiple PDF files or a folder. If the input is a PDF file, you'd better specify a page list to extract some pages of the PDF file and merge them to a new PDF file. If you just input one PDF file or a folder that only contains one PDF file without a page list, the generated PDF file will be the same as the original input PDF file without any changes.

There are two input formats available: specifying an input file without a page list, or with a page list. The format of specifying an input file without a page list means selecting all pages of the input PDF file(s) to merge them to a new PDF file. The input should be like the following examples:

-i "c:\input\1.pdf" "c:\input\2.pdf" (multiple PDF files)
-i "test\1.pdf" "test\2.pdf" ("test" folder is in the current working folder)
-i c:\input (a single folder)
-i test ("test" folder is in the current working folder)

The other format of specifying an input file with a page list means extracting some pages from the input PDF file(s) to merge them to a new PDF file. The input should be like the following examples:

-i "c:\input\1.pdf,1,5,9"
-i "test\1.pdf,2-8,20,30"
-i "c:\input\1.pdf,odd" "c:\input\2.pdf,even"
-i "test\1.pdf,all" "test\2.pdf,10,20-"
-i "c:\input,odd"
-i "test,even,1-10"

For more details about how to specify the page list as part of the input file, please refer to "Page List Format".

The **output** should be a PDF file. If you just type like "-o output", the generated PDF file will be automatically named "output.pdf", with the extension of ".pdf" added.

The output format should be like the following examples:

-o d:\output\3.pdf (a single PDF file)
-o output\3.pdf ("output" folder is in the current working folder)

Usage Example

1) Merge "c:\1.pdf" and "c:\2.pdf" into one PDF file (-mode 1)

fpdfppo -i "c:\1.pdf" "c:\2.pdf" -o d:\merge.pdf -mode 1
fpdfppo -i "c:\1.pdf" "c:\2.pdf" -o d:\merge.pdf (the default mode is "-mode 1")

2) Merge the PDF files under "c:\input" folder into one PDF file (-mode 1)

fpdfppo -i c:\input -o d:\merge.pdf -mode 1
fpdfppo -i "c:\input*.pdf" -o d:\merge.pdf -mode 1 or
fpdfppo -i c:\input -o d:\merge.pdf (the default mode is "-mode 1")
fpdfppo -i c:\input*.pdf -o d:\merge.pdf

3) Extract all the even pages from "1.pdf" to merge them to a new PDF file (-mode 1)

```
fpdfppo -i "c:\1.pdf,even" -o d:\merge.pdf -mode 1
fpdfppo -i "test\1.pdf,even" -o d:\merge.pdf -mode 1                    or
fpdfppo -i "c:\1.pdf,even" -o d:\merge.pdf        (the default mode is "-mode 1")
fpdfppo -i "c:\1.pdf,even" -o d:\merge.pdf
```

4) Merge pages 1,5 and 9 of "1.pdf" and all the pages of "2.pdf" into one PDF file (-mode 1)

```
fpdfppo -i "c:\1.pdf,1,5,9" "c:\2.pdf,all" -o d:\merge.pdf -mode 1
fpdfppo -i "test\1.pdf,1,5,9" "test\2.pdf,all" -o d:\merge.pdf -mode 1      or
fpdfppo -i "c:\1.pdf,1,5,9" "c:\2.pdf,all" -o d:\merge.pdf   (the default mode is "-mode 1")
fpdfppo -i "test\1.pdf,1,5,9" "test\2.pdf,all" -o d:\merge.pdf
```

5) Merge the first five pages of all the PDF files under "c:\input" folder into one PDF file (-mode 1)

```
fpdfppo -i "c:\input,1-5" -o d:\merge.pdf -mode 1
fpdfppo -i "c:\input\*.pdf,1-5" -o d:\merge.pdf -mode 1                 or
fpdfppo -i "c:\input,1-5" -o d:\merge.pdf                  (the default mode is "-mode 1")
fpdfppo -i "c:\input\*.pdf,1-5" -o d:\merge.pdf
```

Split Mode (-mode 2)

In this mode, the **input** should be a single PDF file, multiple PDF files, or a folder.

There are two input formats available: specifying an input file without a page list, or with a page list. The format of specifying an input file without a page list means splitting all pages of the input PDF file(s) into individual PDF files, while each page will be extracted as a separate PDF file. The input should be like the following examples:

```
-i c:\input\1.pdf                    (a single PDF file)
-i test\1.pdf                        ("test" folder is in the current working folder)
-i "c:\input\1.pdf" "c:\input\2.pdf" (multiple PDF files)
-i "test\1.pdf" "test\2.pdf"         ("test" folder is in the current working folder)
-i c:\input                          (a single folder)
-i test                              ("test" folder is in the current working folder)
```

Note *If you want to split the pages of the input PDF file(s) evenly, you can specify the value through "-maxpage" argument. For more details about "-maxpage", please refer to "Maximum Pages Setting".*

The other format of specifying an input file with a page list means extracting some pages from the input PDF file(s) to create some individual PDF files, while each page will be extracted as a separate PDF file. The input should be like the following examples:

-i "c:\input\1.pdf,1,5,9"
-i "test\1.pdf,2-8,20,30"
-i "c:\input\1.pdf,odd" "c:\input\2.pdf,even"
-i "test\1.pdf,all" "test\2.pdf,10,20-"
-i "c:\input,odd"
-i "test,even,1-10"

For more details about how to specify the page list as part of the input file, please refer to "Page List Format".

Note *If you specify a page list and the "-maxpage" argument at the same time, the "-maxpage" argument will be ignored automatically.*

The **output** should be a single PDF file, multiple PDF files, or a folder. The specified PDF names are used for naming the generated PDF files.

The output format should be like the following examples:

-o d:\output\3.pdf	(a PDF file)
-o output\3.pdf	("output" folder is in the current working folder)
-o "d:\output\3.pdf" "d:\output\4.pdf"	(multiple PDF files)
-o output\3.pdf	("output" folder is in the current working folder)
-o d:\output	(a single folder)
-o output	("output" folder is in the current working folder)

File naming for Split mode

- **If the input is a single PDF file**, you can specify an output path of a PDF file or a folder. For example:

  -i c:\input\1.pdf -o d:\output\Foxit.pdf -mode 2

  When you specify an output path of a PDF file, the generated PDFs will be named with the specified PDF name: "*<specified_PDF_name>_split_<page_index>.pdf*".

  So in the example above, the names of the output PDF files are "*Foxit_split_1.pdf*", "*Foxit_split_2.pdf*"…

  -i c:\input\1.pdf -o d:\output -mode 2

When you specify an output path of a folder, the generated PDFs will inherit the name of the input file and be named as "*<inputfile_name>_split_<page_index>.pdf*".

So in the example above, the names of the output PDF files are "*1_split_1.pdf*", "*1_split_2.pdf*"…

- **If the input is multiple PDF files**, you can specify an output path of PDF files with the same number of input files, or an output path of a folder. For example:

  -i "c:\input\1.pdf" "c:\input\2.pdf" -o "d:\Foxit1.pdf" "d:\Foxit2.pdf" -mode 2

  When you specify an output path of PDF files with the same number of input files, the generated PDFs will be named with the corresponding specified PDF names: "*<specified_PDF_name1>_split_<page_index>.pdf*", and "*<specified_PDF_name2>_split_<page_index>.pdf*".

  So in the example above, the names of the output PDF files are "*Foxit1_split_1.pdf*", "*Foxit1_split_2.pdf*"…, and "*Foxit2_split_1.pdf*", "*Foxit2_split_2.pdf*"…

  -i "c:\input\1.pdf" "c:\input\2.pdf" -o d:\output -mode 2

  When you specify an output path of a folder, the generated PDFs will inherit the names of the input files and be named as "*<inputfile_name1>_split_<page_index>.pdf*", and "*<inputfile_name2>_split_<page_index>.pdf*".

  So in the example above, the names of the output PDF files are "*1_split_1.pdf*", "*1_split_2.pdf*"…, and "*2_split_1.pdf*", "*2_split_2.pdf*"…

- **If the input is a single folder**, you can only specify an output path of a folder. For example:

  -i c:\input -o d:\output -mode 2

  If the input is a single folder, the PDF files in the given folder that need to be processed will have the same operations, and the generated PDFs will inherit the names of the input files, and be named as "*<inputfile_name>_split_<page_index>.pdf*".

  So in the example above, all of the PDF files under "c:\input" folder (not including sub-folders) will be split. The output PDF files will be saved to "d:\output", and inherit the names of the input files with the suffix "*_split_<page_index>.pdf*".

  If you want to split the PDF files in the sub-folders of "c:\output", please specify the "-r" argument. For more details about "-r", please refer to "Recursion Depth of Sub-folders".

Usage Example

1) Split every page of "1.pdf" into individual PDF files (-mode 2)

```
fpdfppo -i c:\input\1.pdf -o d:\output\split.pdf -mode 2
fpdfppo -i c:\input\1.pdf -o d:\output -mode 2
fpdfppo -i test\1.pdf -o d:\output\split.pdf -mode 2
fpdfppo -i test\1.pdf -o d:\output -mode 2
```

2) Split three PDF files into individual PDF files (-mode 2)

```
fpdfppo -i "c:\1.pdf" "c:\2.pdf" "c:\3.pdf" -o d:\output -mode 2
fpdfppo -i "1.pdf" "2.pdf" "3.pdf" -o d:\output -mode 2
fpdfppo -i "c:\1.pdf" "c:\2.pdf" "c:\3.pdf" -o "d:\4.pdf" "d:\5.pdf" "d:\6.pdf" -mode 2
fpdfppo -i "1.pdf" "2.pdf" "3.pdf" -o "4.pdf" "5.pdf" "6.pdf" -mode 2
```

3) Split the PDF files in a given folder into individual PDF files (-mode 2)

```
fpdfppo -i c:\input -o d:\output -mode 2
fpdfppo -i test -o d:\output -mode 2
fpdfppo -i c:\input\*.pdf -o d:\output –mode 2
```

4) Extract pages 2,5 and 10 from one PDF file to create three individual PDF files (-mode 2)

```
fpdfppo -i "c:\input\1.pdf,2,5,10" -o d:\output\split.pdf -mode 2
fpdfppo -i "c:\input\1.pdf,2,5,10" -o d:\output -mode 2
fpdfppo -i "test\1.pdf,2,5,10" -o d:\output\split.pdf -mode 2
fpdfppo -i "test\1.pdf,2,5,10" -o d:\output -mode 2
```

5) Extract all odd pages from "1.pdf" and all even pages of "2.pdf" to create several individual PDF files (-mode 2)

```
fpdfppo -i "c:\1.pdf,odd" "c:\2.pdf,even" -o d:\output -mode 2
fpdfppo -i "1.pdf,odd" "2.pdf,even" -o d:\output -mode 2
fpdfppo -i "c:\1.pdf,odd" "c:\2.pdf,even" -o "d:\3.pdf" "d:\4.pdf" -mode 2
fpdfppo -i "1.pdf,odd" "2.pdf,even" -o "3.pdf" "4.pdf" -mode 2
```

6) Extract the first five pages from all the PDF files in a given folder to create several individual PDF files (-mode 2)

```
fpdfppo -i "c:\input,1-5" -o d:\output -mode 2
fpdfppo -i "test,1-5" -o d:\output -mode 2
fpdfppo -i "c:\input\*.pdf,1-5" -o d:\output -mode 2
```

Delete Mode (-mode 3)

Note you cannot delete all of the pages in a PDF file, because PDF files must contain at least one page.

In this mode, the **input** should be a single PDF file, multiple PDF files, or a folder. You should specify a page list that you want to delete, or a warning message will prompt you that you cannot delete all of the pages in a PDF file and you should specify a valid page list.

The input format should be like the following examples:

-i "c:\input\1.pdf,1,5,9"	(a single PDF file)
-i "test\1.pdf,2-10,even"	("test" folder is in the current working folder)
-i "c:\input\1.pdf,odd" "c:\input\2.pdf,even"	(multiple PDF files)
-i "test\1.pdf,20-" "test\2.pdf,1-5,20-"	("test" folder is in the current working folder)
-i "c:\input,2,5,8"	(a single folder)
-i "test,odd,50-"	("test" folder is in the current working folder)

For more details about how to specify the page list as part of the input file, please refer to "Page List Format".

The **output** should be a single PDF file, multiple PDF files, or a folder. The specified PDF names are used for naming the generated PDF files.

The output format should be like the following examples:

-o d:\output\3.pdf	(a PDF file)
-o output\3.pdf	("output" folder is in the current working folder)
-o "d:\output\3.pdf" "d:\output\4.pdf"	(multiple PDF files)
-o output\3.pdf	("output" folder is in the current working folder)
-o d:\output	(a single folder)
-o output	("output" folder is in the current working folder)

File naming for Delete mode

- **If the input is a single PDF file**, you can specify an output path of a PDF file or a folder. For example:

  -i "c:\input\1.pdf,1,5,9" -o d:\output\Foxit.pdf -mode 3

  When you specify an output path of a PDF file, the generated PDF will be named with the specified PDF name.

  So in the example above, pages 1, 5, and 9 of "1.pdf" under "c:\input" will be deleted, and the output PDF file will be saved to "d:\output" and named "*Foxit.pdf*".

-i "c:\input\1.pdf,1,5,9" -o d:\output -mode 3

When you specify an output path of a folder, the generated PDF will inherit the name of the input file and be named as "*<inputfile_name>_del.pdf*".

So in the example above, pages 1, 5, and 9 of "1.pdf" under "c:\input" will be deleted, and the output PDF file will be saved to "d:\output" and named "*1_del.pdf*".

- **If the input is multiple PDF files**, you can specify an output path of PDF files with the same number of input files, or an output path of a folder. For example:

  -i "c:\input\1.pdf,odd" "c:\input\2.pdf,even" -o "d:\Foxit1.pdf" "d:\Foxit2.pdf" -mode 3

  When you specify an output path of PDF files with the same number of input files, the generated PDFs will be named with the corresponding specified PDF names.

  So in the example above, in "c:\input" folder, the odd pages of "1.pdf" and the even pages of "2.pdf" will be deleted, and then the output PDF files will be saved to "d:\" and named "Foxit1.pdf" and "Foxit2.pdf" respectively.

  -i "c:\input\1.pdf,odd" "c:\input\2.pdf,even" -o d:\output -mode 3

  When you specify an output path of a folder, the generated PDFs will inherit the names of the input files and be named as "*<inputfile_name1>_del.pdf*", and "*<inputfile_name2>_del.pdf*".

  So in the example above, in "c:\input" folder, the odd pages of "1.pdf" and the even pages of "2.pdf" will be deleted, and then the output PDF files will be saved to "d:\output" and named "*1_del.pdf*" and "*2_del.pdf*" respectively.

- **If the input is a single folder**, you can only specify an output path of a folder. For example:

  -i "c:\input,2-5,100-" -o d:\output -mode 3

  If the input is a single folder, the PDF files in the given folder that need to be processed will have the same operations, and the generated PDFs will inherit the names of the input files, and be named as "*<inputfile_name>_del.pdf*".

  So in the example above, all of the PDF files under "c:\input" folder (not including sub-folder) will delete the pages in the range from 2 to 5 and from 100 to the last page. The output PDF files will be saved to "d:\output", and inherit the names of the input files with the suffix "*_del.pdf*".

  If you want to delete the pages of PDF files in the sub-folders of "c:\output", please specify the "-r" argument. For more details about "-r", please refer to "Recursion Depth of Sub-folders".

Usage Example

1) Delete page 1 of a PDF file (-mode 3)

```
fpdfppo -i "c:\input\1.pdf,1" -o d:\output\out.pdf -mode 3
fpdfppo -i "c:\input\1.pdf,1" -o d:\output -mode 3
fpdfppo -i "test\1.pdf,1" -o d:\output\out.pdf -mode 3
fpdfppo -i "test\1.pdf,1" -o d:\output -mode 3
```

2) Delete the pages in the range from 4-10 and all odd pages of a PDF file (-mode 3)

```
fpdfppo -i "c:\input\1.pdf,4-10,odd" -o d:\output\out.pdf -mode 3
fpdfppo -i "c:\input\1.pdf,4-10,odd" -o d:\output -mode 3
fpdfppo -i "test\1.pdf,4-10,odd" -o d:\output\out.pdf -mode 3
fpdfppo -i "test\1.pdf,4-10,odd" -o d:\output -mode 3
```

3) Delete pages 2 and 5 of "1.pdf" and pages 3 and 8 of "2.pdf" (-mode 3)

```
fpdfppo -i "c:\1.pdf,2,5" "c:\2.pdf,3,8" -o d:\output -mode 3
fpdfppo -i "1.pdf,2,5" "2.pdf,3,8" -o d:\output -mode 3
fpdfppo -i "c:\1.pdf,2,5" "c:\2.pdf,3,8"  -o "d:\out1.pdf" "d:\out2.pdf" -mode 3
fpdfppo -i "1.pdf,2,5" "2.pdf,3,8"  -o "out1.pdf" "out2.pdf" -mode 3
```

4) Delete all even pages of the PDF files in a given folder (-mode 3)

```
fpdfppo -i "c:\input,even"-o d:\output -mode 3
fpdfppo -i "test,even" -o d:\output -mode 3
fpdfppo -i "c:\input\*.pdf,even" -o d:\output -mode 3
```

3.12.3.4 Maximum Pages Setting (-maxpage)

- The optional argument (**-maxpage**) is used to set the maximum number of pages for each split PDF file. The number of pages for the last PDF file will be smaller than the maximum number if the total number of the PDF pages is not a multiple of the value specified by "-maxpage".

  Note *It is valid only for Split mode and the input is not followed by a page list.*

Usage Example

1) Split "1.pdf" into a set of PDF files containing 5 pages each (-mode 2 -maxpage 5)

```
fpdfppo -i c:\input\1.pdf -o d:\output\split.pdf -mode 2 -maxpage 5
```

```
fpdfppo -i c:\input\1.pdf -o d:\output -mode 2 -maxpage 5
fpdfppo -i test\1.pdf -o d:\output\split.pdf -mode 2 -maxpage 5
fpdfppo -i test\1.pdf -o d:\output -mode 2 -maxpage 5
```

2) Split "1.pdf" and "2.pdf" into a set of PDF files containing 3 pages each (-mode 2 -maxpage 3)

```
fpdfppo -i "c:\1.pdf" "c:\2.pdf" -o d:\output -mode 2 -maxpage 3
fpdfppo -i "1.pdf" "2.pdf" -o d:\output -mode 2 -maxpage 3
fpdfppo -i "c:\1.pdf" "c:\2.pdf" -o "d:\3.pdf" "d:\4.pdf" -mode 2 -maxpage 3
fpdfppo -i "1.pdf" "2.pdf" -o "3.pdf" "4.pdf" -mode 2 -maxpage 3
```

3) Split the PDF files in a given folder into a set of PDF files containing 2 pages each (-mode 2 -maxpage 2)

```
fpdfppo -i c:\input -o d:\output -mode 2 -maxpage 2
fpdfppo -i test -o d:\output -mode 2 -maxpage 2
fpdfppo -i c:\input\*.pdf -o d:\output –mode 2 -maxpage 2
```

3.12.3.5 Delete Bookmarks (-deletebm)

- The optional argument (**-deletebm**) is used to delete the bookmarks of the merged PDF file. It is valid only for Merge mode.

  Note *The bookmarks will be saved only when the output PDF file is merged from the whole input files rather than extracted pages thereof. In the Split and Delete modes, the bookmarks will not be retained in the output PDF files, and in Merge Mode, if you specify an input file with a page list, the bookmarks will also not be retained in the output PDF file.*

Usage Example

1) Merge "c:\1.pdf" and "c:\2.pdf" into one PDF file containing no bookmarks (-deletebm)

```
fpdfppo -i "c:\1.pdf" "c:\2.pdf" -o d:\merge.pdf -mode 1 -deletebm          or
fpdfppo -i "c:\1.pdf" "c:\2.pdf" -o d:\merge.pdf -deletebm          (the default mode is "-mode 1")
```

2) Merge the PDF files under "c:\input" folder into one PDF file containing no bookmarks (-deletebm)

```
fpdfppo -i c:\input -o d:\merge.pdf -mode 1 -deletebm
fpdfppo -i "c:\input\*.pdf" -o d:\merge.pdf -mode 1 -deletebm          or
fpdfppo -i c:\input -o d:\merge.pdf -deletebm          (the default mode is "-mode 1")
```

fpdfppo -i c:\input*.pdf -o d:\merge.pdf -deletebm

3.12.3.6 Password for the Input File (-op)

- The optional argument (**-op**) indicates the password for a password-protected input PDF file. It is not required if the input file is not password protected. There are two kinds of passwords, one is user password, and the other is owner password.

  User password is also known as "open password", which protects the document against unauthorized opening and reading.

  Owner password is also known as "master password", which protects the document against unauthorized editing, printing, changing, and copying. It grants users full access to the document. With the owner password, the document can not only be opened and read, but also be allowed to change the document's security settings. This means that even if the permission of modifying the PDF file was restricted, you can still modify it with the owner password.

  Note

  - *The output PDF file will not retain the password from the input file.*
  - *If the input PDF file is protected by a user password, you need to provide the password using the option "-op" to process the PDF file.*
  - *If the input PDF file is protected by an owner password, there are two circumstances. If the permissions of changing the PDF file were not restricted, you can process the PDF file without providing the owner password; otherwise, if the permissions of changing the PDF file were restricted, you need to provide the password using the option "-op" to process the PDF file.*
  - *If the input PDF file is protected by a user password and an owner password, there are also two circumstances. If the permissions of changing the PDF file were not restricted, you can provide either of the passwords using the option "-op" to process the PDF file; otherwise, if the permissions of changing the PDF file were restricted, you need to provide the owner password using "-op" option to process the PDF file.*

Usage Example

1) Specify the password for a password-protected input PDF file (-op 123)

```
fpdfppo -i c:\input\1.pdf -o d:\output -mode 2 -op 123
fpdfppo -i "c:\input\1.pdf,2,5,9" -o d:\output -mode 3 -op 123
fpdfppo -i "test\2.pdf,even" -o output\merge.pdf -mode 1 -op 123
fpdfppo -i "test\2.pdf,2,5,9,odd" -o output -mode 2 -op 123
```

```
fpdfppo -i "test\2.pdf,1-5" -o output -mode 3 -op 123
```

2) Specify the password for multiple password-protected input PDF files that share the same password (-op 123)

```
fpdfppo -i "c:\1.pdf" "c:\2.pdf" -o d:\output\merge.pdf -mode 1 -op 123
fpdfppo -i "c:\1.pdf" "c:\2.pdf" -o d:\output -mode 2 -op 123
fpdfppo -i "c:\1.pdf,2,3" "c:\2.pdf,5,8" -o d:\output -mode 3 -op 123
fpdfppo -i "1.pdf,even" "2.pdf,odd" -o output\merge.pdf -mode 1 -op 123
fpdfppo -i "1.pdf,all" "2.pdf,2,5,10-" -o output\split.pdf -mode 2 -op 123
fpdfppo -i "1.pdf,1-5" "2.pdf,2-6" -o output\delete.pdf -mode 3 -op 123
```

3) Specify the password for all password-protected input PDF files in a given folder that share the same password (-op welcome)

```
fpdfppo -i c:\input -o d:\output\merge.pdf -mode 1 -op welcome
fpdfppo -i c:\input -o d:\output -mode 2 -op welcome
fpdfppo -i "c:\input,2,5,9" -o d:\output -mode 3 -op welcome
fpdfppo -i "test,even" -o output\merge.pdf -mode 1 -op welcome
fpdfppo -i "test,1-5,20-" -o output\split.pdf -mode 2 -op welcome
fpdfppo -i "test,2,5-9" -o output\delete.pdf -mode 3 -op welcome
fpdfppo -i "c:\input\*.pdf" -o d:\output\merge.pdf -mode 1 -op welcome
fpdfppo -i "c:\input\*.pdf,2,5,even" -o d:\output -mode 2 -op welcome
fpdfppo -i "c:\input\*.pdf,1-3" -o d:\output -mode 3 -op welcome
```

Note *It only supports typing one value for the argument (**-op**). Only files with the same password can be processed together and files with different password need to be processed separately.*

3.12.3.7 Recursion Depth of Sub-folders (-r)

- The optional argument (**-r**) indicates the recursion depth of sub-folders, which is valid only when the input is a folder or a path that includes wildcard characters like "c:\input*.pdf". By default, the recursion depth is 1, so the sub-folders will not be processed. For more details about this argument, please refer to section 3.12.2 "Command Line Summary".

Usage Examples

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

```
fpdfppo -i c:\input -o d:\output\merge.pdf -mode 1 -r 0
fpdfppo -i c:\input -o d:\output -mode 2 -r 0
fpdfppo -i "c:\input,2,5,9" -o d:\output -mode 3 -r 0
fpdfppo -i "test,even" -o output\merge.pdf -mode 1 -r 0
fpdfppo -i "test,1-5,20-" -o output\split.pdf -mode 2 -r 0
fpdfppo -i "test,2,5-9" -o output\delete.pdf -mode 3 -r 0
fpdfppo -i "c:\input\*.pdf" -o d:\output\merge.pdf -mode 1 -r
fpdfppo -i "c:\input\*.pdf,2,5,even" -o d:\output -mode 2 -r
fpdfppo -i "c:\input\*.pdf,1-3" -o d:\output -mode 3 -r
```

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

```
fpdfppo -i c:\input -o d:\output\merge.pdf -mode 1 -r 1
fpdfppo -i c:\input -o d:\output -mode 2 -r 1
fpdfppo -i "c:\input,2,5,9" -o d:\output -mode 3 -r 1
fpdfppo -i "test,even" -o output\merge.pdf -mode 1 -r 1
fpdfppo -i "test,1-5,20-" -o output\split.pdf -mode 2 -r 1
fpdfppo -i "test,2,5-9" -o output\delete.pdf -mode 3 -r 1
fpdfppo -i "c:\input\*.pdf" -o d:\output\merge.pdf -mode 1 -r 1
fpdfppo -i "c:\input\*.pdf,2,5,even" -o d:\output -mode 2 -r 1
fpdfppo -i "c:\input\*.pdf,1-3" -o d:\output -mode 3 -r 1
```

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

```
fpdfppo -i c:\input -o d:\output\merge.pdf -mode 1
fpdfppo -i c:\input -o d:\output -mode 2
fpdfppo -i "c:\input,2,5,9" -o d:\output -mode 3
fpdfppo -i "test,even" -o output\merge.pdf -mode 1
fpdfppo -i "test,1-5,20-" -o output\split.pdf -mode 2
fpdfppo -i "test,2,5-9" -o output\delete.pdf -mode 3
fpdfppo -i "c:\input\*.pdf" -o d:\output\merge.pdf -mode 1
fpdfppo -i "c:\input\*.pdf,2,5,even" -o d:\output -mode 2
fpdfppo -i "c:\input\*.pdf,1-3" -o d:\output -mode 3
```

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

```
fpdfppo -i c:\input -o d:\output\merge.pdf -mode 1 -r 2
fpdfppo -i c:\input -o d:\output -mode 2 -r 2
fpdfppo -i "c:\input,2,5,9" -o d:\output -mode 3 -r 2
```

```
fpdfppo -i "test,even" -o output\merge.pdf -mode 1 -r 2
fpdfppo -i "test,1-5,20-" -o output\split.pdf -mode 2 -r 2
fpdfppo -i "test,2,5-9" -o output\delete.pdf -mode 3 -r 2
fpdfppo -i "c:\input\*.pdf" -o d:\output\merge.pdf -mode 1 -r 2
fpdfppo -i "c:\input\*.pdf,2,5,even" -o d:\output -mode 2 -r 2
fpdfppo -i "c:\input\*.pdf,1-3" -o d:\output -mode 3 -r 2
```

3.12.3.8 Multi-thread Support (-t)

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

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

Usage Example

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

```
fpdfppo -i c:\input -o d:\output\merge.pdf -mode 1 -t 3
fpdfppo -i c:\input -o d:\output -mode 2 -t 3
fpdfppo -i "c:\input,2,5,9" -o d:\output -mode 3 -t 3
fpdfppo -i "test,even" -o output\merge.pdf -mode 1 -t 3
fpdfppo -i "test,1-5,20-" -o output\split.pdf -mode 2 -t 3
fpdfppo -i "test,2,5-9" -o output\delete.pdf -mode 3 -t 3
fpdfppo -i "c:\input\*.pdf" -o d:\output\merge.pdf -mode 1 -t 3
fpdfppo -i "c:\input\*.pdf,2,5,even" -o d:\output -mode 2 -t 3
fpdfppo -i "c:\input\*.pdf,1-3" -o d:\output -mode 3 -t 3
```

3.12.3.9 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 argument (**-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 3.12.2 "Command Line Summary".

Usage Example

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

```
fpdfppo -i c:\input -o d:\output\merge.pdf -mode 1 -log d:\output\pdfppo.log -l 3
fpdfppo -i c:\input -o d:\output -mode 2 -log d:\output\pdfppo.log -l 3
fpdfppo -i "c:\input,2,5,9" -o d:\output -mode 3 -log d:\output\pdfppo.log -l 3
fpdfppo -i "test,even" -o output\merge.pdf -mode 1 -log d:\output\pdfppo.log -l 3
fpdfppo -i "test,1-5,20-" -o output\split.pdf -mode 2 -log d:\output\pdfppo.log -l 3
fpdfppo -i "test,2,5-9" -o output\delete.pdf -mode 3 -log d:\output\pdfppo.log -l 3
fpdfppo -i "c:\input\*.pdf" -o d:\output\merge.pdf -mode 1 -log d:\output\pdfppo.log -l 3
fpdfppo -i "c:\input\*.pdf,2,5,even" -o d:\output -mode 2 -log d:\output\pdfppo.log -l 3
fpdfppo -i "c:\input\*.pdf,1-3" -o d:\output -mode 3 -log d:\output\pdfppo.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 the **<licensee>** is the licensee name designated by the users.

Usage Example

1) Register the pdf page organizer tool with the code "xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx" and the licensee "Foxit" (-register xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx Foixt)

```
fpdfppo -register xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx Foixt
```

c) License agreement (-license)

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

Usage Example

1) Print the license agreement (-license)

```
fpdfppo -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)

fpdfppo -version
fpdfppo -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)

fpdfppo -help
fpdfppo -h

f) Copyright information (-copyright)

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

Usage Example

1) Print the copyright information (-copyright)

fpdfppo -copyright

3.13 PDFSecure

3.13.1 Basic Syntax

fpdfsecure *<-i <srcfile/srcfolder>> <-o <destfile/destfolder>> <-mode <mode type>>*
[-sup <user password>] [-sop <owner password>]
[-cert <certificate>] [-certpwd <password of private key>]
[-certname <certificate name>] [-certstore <certificate store>]
[-p <permission flags>] [-ea <security handler>] [-em]
[-title <title>] [-subject <subject>] [-keywords <keywords>] [-author <author>] [-creator <creator>]
[-op <password>] [-r [recursion]] [-t <thread>] [-log <logfile>] [-l <level>]
fpdfsecure *<-i <srcfile>> <-mode <0>> <-lp> [-op <password>] [-log <logfile>] [-l <level>]*
fpdfsecure *-register <code> <licensee>*
fpdfsecure *-license*
fpdfsecure *-version/-v*
fpdfsecure *-help/-h*
fpdfsecure *-copyright*

Note:

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

If you want to encrypt PDF files, you should set not only *<-i <srcfile/srcfolder>>*, *<-o <destfile/destfolder>>* and *<-mode <1/2>>* arguments, but also at least one of the following arguments (-sup, -sop, -cert, and -certname) according to the setting value of the mode. If you just want to print the permission flags of a PDF file, the *<-i <srcfile>>*, *<-mode <0>> and <-lp>* arguments are required.
All others are optional, which are available for controlling the process as desired. The arguments could be given in any order. Full details on each are explained in the following section.

3.13.2 Command Line Summary

Note *For some arguments whose values are strings, users can choose whether to add quotation marks (" ") to the strings. In the manual we have added notes where adding quotation marks (" ") is required.*

Option	Parameter	Description
-i	*<-i <string>>* *<-i <srcfile/srcfolder>>* *e.g.* *-i c:\input\1.pdf* *-i c:\input* *-i "c:\input*.pdf"*	Specifies the input file to be processed. ▪ The input string can be the name of a single PDF file or a folder. ▪ The file name can contain the wildcard character (*). For example, use *.pdf to include all PDF files in a given folder. **Note** Wildcard character (*.*) is currently not supported.
-o	*<-o <string>>* *<-o <destfile/destfolder>>* *e.g.* *-o d:\output\1_secure.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_secure.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>>* *<-mode <mode type>>* *e.g.* *-mode 0* *-mode 1* *-mode 2*	Specifies the mode of operation. The default value is 1. • **-mode 0**: lists the user's permissions to a PDF file. • **-mode 1**: password encryption. • **-mode 2**: certificate encryption.
-sup	*[-sup <string>]* *[-sup <user password>]* *e.g.* *-sup 12345* *-sup welcome* *-sup welcome123*	Sets a user password for the output PDF file. User password protects the PDF files against unauthorized opening and reading. **Note** The argument (-**sup**) is valid only when (-**mode**) is set to 1.
-sop	*[-sop <string>]* *[-sop <owner password>]* *e.g.* *-sup 12345* *-sup welcome* *-sup welcome123*	Sets an owner password for the output PDF file. Owner password protects the PDF files against unauthorized editing, printing, changing and copying. **Note** The argument (-**sop**) is valid only when (-**mode**) is set to 1.

Option	Parameter	Description
-cert	*[-cert <string>]* *[-cert <certificate>]* *e.g.* *-cert public_cert.cer* *-cert private_cert.pfx*	Encrypts the output PDF file with a certificate file. It supports ".cer", and ".pfx" file formats. **Note** The argument (-**cert**) is valid only when (-**mode**) is set to 2.
-certpwd	*[-certpwd <string>]* *[-certpwd <password of private key>]* *e.g.* *-certpwd "123456"*	Specifies a password for the private key of the certificate. **Note** The private key of the certificate is protected by a password, which is only known to its owner. When you use a ".pfx" file to encrypt PDF files, the (-**certpwd**) is required.
-certname	*[-certname <string>]* *[-certname <certificate name>]* *e.g.* *-certname "Foxit"*	Specifies the name of the certificate in Windows certificate store used to encrypt the output PDF file. **Note** In Windows certificate store, it corresponds to "Issued to".
-certstore	*[-certstore <integer>]* *[-certstore <certificate store>]* *e.g.* *-certstore 1* *-certstore 2* *-certstore 3*	Sets the system certificate store where the signing certificate should be taken. The default value is 1 (MY). • **-certstore 1**: MY. Personal certificates. • **-certstore 2**: ROOT. Trusted Root Certification Authorities. • **-certstore 3**: CA. Intermediate Certification Authorities.
-p	*[-p <string>]* *[-p <permission flags>]* *e.g.* *-p 0* *-p 1* *-p 234* *-p 3456789*	Sets the permission flags for the output PDF file. The default value is 1. • **0**: allows nothing (no permissions are granted) • **1**: allows everything (all permissions are granted) • **2**: allows printing with low resolution. • **3**: allows printing with high resolution. • **4**: allows filling in a form. • **5**: allows commenting in the document. • **6**: allows managing pages and bookmarks.

Option	Parameter	Description
		• **7**: allows modifying document. • **8**: allows content copying for accessibility. • **9**: allows extracting the contents of document. **Note** ▪ The argument (**-p**) is valid only when (**-sop**), (**-cert**) or (-**certname**) is used. ▪ The < permission flags> should be one integer from 0 to 9, or a combination of the integers from 2 to 9 without any separators. If the combination string contains 0 or 1, other permission flags will not work.
-ea	*[-ea <integer>]* *[-ea <security handler>]* *e.g.* *-ea 1* *-ea 2* *-ea 3*	Sets a security handler to encrypt the output PDF file. The value should be 1, 2 or 3, and the default value is 1. • **-ea 1**: 128-bit AES encryption. • **-ea 2**: 256-bit AES encryption. • **-ea 3**: 128-bit RC4 encryption.
-em	*[-em]* *e.g.* *-em*	Encrypts the metadata of the output PDF file.
-lp	*[-lp]* *e.g.* *-lp*	Lists the user's permissions to the input PDF designated by the **-i** option. **Note** The argument (**-lp**) can only be used with **-i**, **-mode**, **-op**, **-log** and **-l** options. Other arguments will be ignored if set. The value of **-i** should be a PDF file, and the value of **-mode** should be 0.
-title	*<-title <string>>* *e.g.* *-title "Foxit PDF Toolkit"*	Sets title of PDF files.
-subject	*<-subject <string>>* *e.g.* *-subject "PDF Toolkit"*	Sets subject of PDF files.

Option	Parameter	Description
-keywords	*[-keywords <string>]* *e.g.* *-keywords "Foxit"*	Sets keywords of PDF files.
-author	*[-author <string>]* *e.g.* *-author "Jessie"*	Sets author of PDF files.
-creator	*[-creator <string>]* *e.g.* *-creator "Foxit PhantomPDF"* *-creator "Foxit Reader"* *-creator "Microsoft® Word 2013"*	Sets creator of PDF files. **Note** It indicates the name of the application that created the source document from which a PDF is generated. For example, if the source document was created from Microsoft® *Word 2013, then you can set the creator to* "Microsoft® *Word 2013".*
-op	*[-op<string>]* *e.g.* *-op "123"* *-op "welcome"*	Specifies the password for the input file. Not required if the input file is not password protected. **Note** ▪ *If the input PDF file is protected by a user password or an owner password, you need to provide the corresponding password to change the document's security settings.* ▪ *If the input PDF file is protected by a user password and an owner password, the owner password is required to change the document's security settings.*
-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. • **-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** ▪ If no integer value is specified, or if the integer value is 0, then full folders will be searched. By

Option	Parameter	Description
		default, the number of layers to recurse is 1, which means that only the current folder will be searched and not sub-folders. ▪ The input PDF file or folder will be skipped if it is secured and the messages will be displayed.
-t	*[-t <integer>]* *e.g.* *-t 1* *-t 2* *…*	Specifies the number of CPU threads to use. The default value is 1.
-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 messages only concerning out of memory errors. • **-l 2**: logs failure messages concerning the errors caused during execution or those returned from underlying libraries, as well as those for level 1. • **-l 3**: logs warning messages concerning the PDF files that are overwritten, as well as those for level 2. • **-l 4**: logs informational messages, as well as those for level 3. **Note** The argument (**-l**) is valid only when (**-log**) is used.
-register	*[-register <String> <String>]* *-register <code> <licensee>* *e.g.* *-register* xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxxx Foxit	Registers the command line tool. ▪ **<code>**: the activation code from Foxit. ▪ **<licensee>**: the Licensee name designated by the users.
-help/-h	*[-help]/[-h]* *e.g.* *-help*	Prints the usage information.

Option	Parameter	Description
	-h	
-version/-v	*[-version]/[-v]* *e.g.* *-version* *-v*	Prints the version information.
-license	*[-license]* *e.g.* *-license*	Prints the license agreement.
-copyright	*[-copyright]* *e.g.* *-copyright*	Prints the copyright information.

3.13.3 Basic Usage

3.13.3.1 Input and Output (-i, -o)

a) Input (-i)

- The input file should be a single PDF file or a folder. Users are not able to input multiple PDF files or folders, as well as a mixture composed 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 if the input file is in the current working folder. Users can input just the name of the PDF file or folder, instead of an absolute path. For example:

  -i test\1.pdf ("test" folder 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 encrypt PDF files under "c:\input" folder)
  -i "test*.pdf" (Only encrypt PDF files under "test" folder)

  Note *When using wildcard characters in the input files, it is recommended to enclose the input files with quotation marks (" "). In this manual, we add (" ") whenever the input files contain wildcard characters.*

Note *If the input paths or file names contain spaces, you must enclose the input files with quotation marks (" "), such as -i "c:\input file\1.pdf", -i "test\user manual.pdf". This rule is also used for some other arguments whose values are strings.*

b) Output (-o)

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

  -o d:\output\1_secure.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. Users can input just the name of the output PDF file or folder, instead of an absolute path. For example:

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

3.13.3.2 Modes of Operation (-mode)

- The optional argument (**-mode**) is required in the command line, which is used to specify the mode of operation. Foxit PDFSecure supports the following three modes:

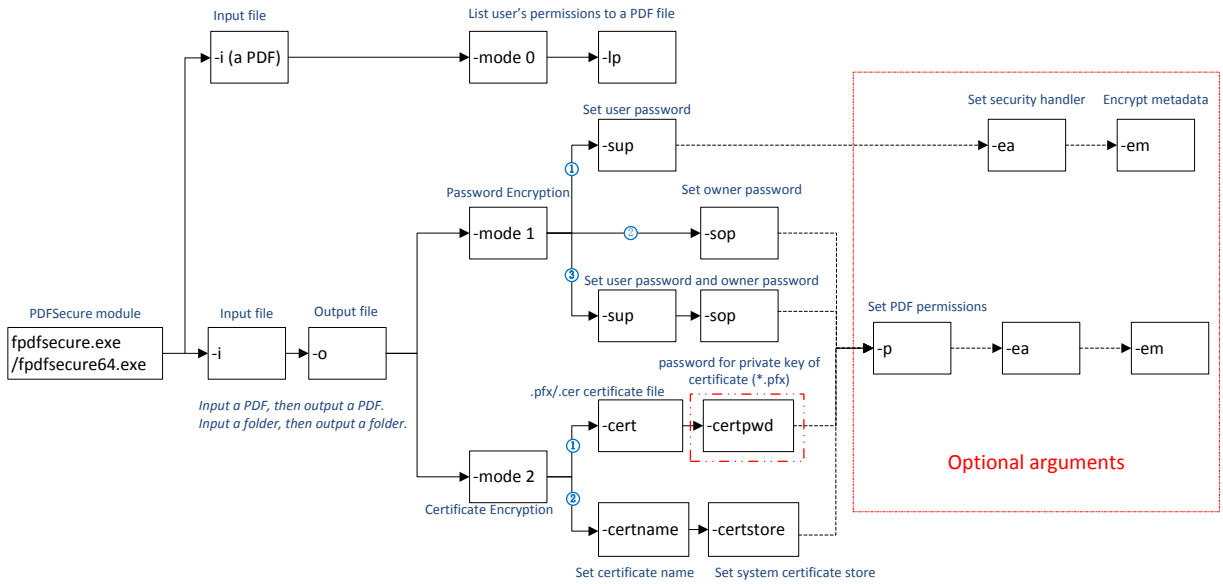
  - **-mode 0**: lists the user's permissions to a PDF file.
  - **-mode 1**: adds password encryption.
  - **-mode 2**: adds certificate encryption.

  In "-**mode 0**", list the user's permission to the PDF file designated by "-i" option using "-**lp**" option.

  In "-**mode 1**", set a password for the output PDF file using "-**sup**" option (user password), "-**sop**" option (owner password), or both options.

  In "-**mode 2**", use a certificate to encrypt the output PDF file using "-**cert**" option (.cer or .pfx file), or "-**certname**" option (the name of the certificate in Windows certificate store). If you use a ".pfx" file to encrypt the PDF file, the "-certpwd" option is required to specify the password of the private key.

  The following picture shows a brief summary for the usage of PDFSecure tool.



A brief summary for the usage of PDFSecure tool

3.13.3.3 User Password and Owner Password Settings (-sup, -sop)

a) User Password (-sup)

- The optional argument (**-sup**) is used to set a user password for the output PDF file. User password is also known as "open password", which protects the document against unauthorized opening and reading. If you want to read a user-password-protected PDF file, the user password is required. It is valid only when (-mode) is set to 1.

  Note

  - *If the input PDF file is protected by a user password, you need to provide the user password using the option "-op" to change the document's security settings.*
  - *Opening the PDF file with the correct user password (or opening a PDF file that does not have a user password) allows additional operations to be performed according to the user access permissions that were specified when the PDF file was created.*

Usage Example

1) Set the user password to "welcome" for one PDF file (-mode 1 -sup "welcome")

```
fpdfsecure -i c:\input\1.pdf -o d:\output\1_secure.pdf -mode 1 -sup "welcome"
fpdfsecure -i test\2.pdf -o output\2_secure.pdf -mode 1 -sup "welcome"
```

2) Set the user password to "123" for all the PDF files in a given folder (-mode 1 -sup "123")

```
fpdfsecure -i c:\input -o d:\output -mode 1 -sup "welcome"
fpdfsecure -i test -o output -mode 1 -sup "welcome"
fpdfsecure -i "c:\input\*.pdf" -o d:\output -mode 1 -sup "welcome"
```

b) Owner Password (-sop)

- The optional argument (**-sop**) is used to set an owner password for the output PDF file. Owner password is also known as master password, which protects the document against unauthorized editing, printing, changing, and copying. It grants the users full access to the document. With owner password, the document can not only be opened and read, but also be allowed to change the document's security settings. This means that even if the printing of the PDF file was restricted, you can still print the PDF file with the owner password. It is valid only when (-mode) is set to 1.

  The following table shows the four possible combinations of passwords and the corresponding processing behaviors.

User password and Owner password

User Password	Owner Password	Behavior
none	none	Everyone can open and read the document. Everyone can change the security settings.
none	set	Everyone can open and read the document. Owner password is required to change the security settings.
set	none	User password is required to open and read the document, and to change the security settings.
set	set	User password or owner password is required to open and read the document. Owner password is required to change the security settings.

Note

- *If the input PDF file is protected by an owner password, you need to provide the owner password using the option "-op" to change the document's security settings.*
- *If the input PDF file is protected by a user password and an owner password, the owner password specified by "-op" option is required to change the document's security settings.*

Usage Example

1) Set the owner password to "foxit" for one PDF file (-mode 1 -sop "foxit")

```
fpdfsecure -i c:\input\1.pdf -o d:\output\1_secure.pdf -mode 1 -sop "foxit"
fpdfsecure -i test\2.pdf -o output\2_secure.pdf -mode 1 -sop "foxit"
```

2) Set the owner password to "456" for all the PDF files in a given folder (-mode 1 -sop "456")

```
fpdfsecure -i c:\input -o d:\output -mode 1 -sop "456"
fpdfsecure -i test -o output -mode 1 -sop "456"
fpdfsecure -i "c:\input\*.pdf" -o d:\output -mode 1 -sop "456"
```

3) Set the owner password to "foxit" and user password to "123" for one PDF file (-mode 1 -sop "foxit" -sup "123")

```
fpdfsecure -i c:\input\1.pdf -o d:\output\1_secure.pdf -mode 1 -sop "foxit" -sup "123"
fpdfsecure -i test\2.pdf -o output\2_secure.pdf -mode 1 -sop "foxit" -sup "123"
```

4) Set the owner password to "456" and user password to "123" for all the PDF files in a given folder (-mode 1 -sop "456" -sup "123")

```
fpdfsecure -i c:\input -o d:\output -mode 1 -sop "456" -sup "123"
fpdfsecure -i test -o output -mode 1 -sop "456" -sup "123"
fpdfsecure -i "c:\input\*.pdf" -o d:\output -mode 1 -sop "456" -sup "123"
```

3.13.3.4 Certificate encryption (-cert, -certpwd, -certname, -certstore)

a) Specify a certificate file (-cert)

- The optional argument (**-cert**) is used to encrypt the output PDF file with a certificate file. It supports ".cer" and ".pfx" file formats. It is valid only when (-mode) is set to 2.

b) Password for the private key of the certificate (-certpwd)

- The optional argument (**-certpwd**) is used to specify a password for the private key of the certificate.

  Note *If you use a ".pfx" file to encrypt the output PDF file, the "-certpwd" is required to specify the password of the private key.*

Usage Example

1) Use the "public_cert.cer" file to encrypt the PDF file(s) (-mode 2 -cert "public_cert.cer")

```
fpdfsecure -i c:\input\1.pdf -o d:\output\1_secure.pdf -mode 2 -cert "public_cert.cer"
fpdfsecure -i test\2.pdf -o output\2_secure.pdf -mode 2 -cert "public_cert.cer"
fpdfsecure -i c:\input -o d:\output -mode 2 -cert "public_cert.cer"
fpdfsecure -i test -o output -mode 2 -cert "public_cert.cer"
fpdfsecure -i "c:\input\*.pdf" -o d:\output -mode 2 -cert "public_cert.cer"
```

2) Use the "private_cert.pfx" file to encrypt the PDF file(s) and specify the password of its private key to "123456" (-mode 2 -cert "private_cert.pfx" -certpwd "123456")

```
fpdfsecure -i c:\input\1.pdf -o d:\output\1_secure.pdf -mode 2 -cert "private_cert.pfx" -certpwd "123456"
fpdfsecure -i test\2.pdf -o output\2_secure.pdf -mode 2 -cert "private_cert.pfx" -certpwd "123456"
fpdfsecure -i c:\input -o d:\output -mode 2 -cert "private_cert.pfx" -certpwd "123456"
fpdfsecure -i test -o output -mode 2 -cert "private_cert.pfx" -certpwd "123456"
fpdfsecure -i "c:\input\*.pdf" -o d:\output -mode 2 -cert "private_cert.pfx" -certpwd "123456"
```

c) The name of the certificate in Windows certificate store (-certname)

- The optional argument (**-certname**) is used to specify the name of the certificate in Windows certificate store used to encrypt the output PDF file. This argument is helpful to the users who want to use an installed certificate to encrypt PDF files, without requiring the *.cer or *.pfx files. It is valid only when (-mode) is set to 2.

  Note *In Windows certificate store, it corresponds to "Issued to".*

Usage Example

1) Use the certificate named "Foxit" in Windows certificate store to encrypt the PDF file(s) (-mode 2 -certname "Foxit")

```
fpdfsecure -i c:\input\1.pdf -o d:\output\1_secure.pdf -mode 2 -certname "Foxit"
fpdfsecure -i test\2.pdf -o output\2_secure.pdf -mode 2 -certname "Foxit"
fpdfsecure -i c:\input -o d:\output -mode 2 -certname "Foxit"
fpdfsecure -i test -o output -mode 2 -certname "Foxit"
fpdfsecure -i "c:\input\*.pdf" -o d:\output -mode 2 -certname "Foxit"
```

d) System certificate store (-certstore)

- The optional argument (**-certstore**) is used to set the system certificate store where the signing certificate should be taken. The default is 1 ("MY"). It is valid only when (-mode) is set to 2 and (-certname) is used.

  - **-certstore 1**: MY. Personal certificates.
  - **-certstore 2**: ROOT. Trusted Root Certification Authorities.
  - **-certstore 3**: CA. Intermediate Certification Authorities.

Usage Example

1) Set the system certificate store to "CA" (-mode 2 -certname "Foxit" -certstore 3)

```
fpdfsecure -i c:\input\1.pdf -o d:\output\1_secure.pdf -mode 2 -certname "Foxit" -certstore 3
fpdfsecure -i test\2.pdf -o output\2_secure.pdf -mode 2 -certname "Foxit" -certstore 3
fpdfsecure -i c:\input -o d:\output -mode 2 -certname "Foxit" -certstore 3
fpdfsecure -i test -o output -mode 2 -certname "Foxit" -certstore 3
fpdfsecure -i "c:\input\*.pdf" -o d:\output -mode 2 -certname "Foxit" -certstore 3
```

3.13.3.5 Permission flags (-p)

- The optional argument (**-p**) is used to set the permission flags for the output PDF file. The argument (-**p**) is valid only when (-**sop**), (-**cert**), or (-**certname**) is used, because only when the document is protected with an owner password or a certificate, it is possible to adjust the PDF permissions. By default, all permissions are granted. The permissions that can be granted are listed as follows.

  - **0**: allows nothing (no permissions are granted)
  - **1**: allows everything (all permissions are granted)
  - **2**: allows printing with low resolution.
  - **3**: allows printing with high resolution.
  - **4**: allows filling in a form.
  - **5**: allows commenting in the document.
  - **6**: allows managing pages and bookmarks.
  - **7**: allows modifying document.
  - **8**: allows content copying for accessibility.
  - **9**: allows extracting the contents of document.

Note *The value of the argument (-p) should be one integer from 0 to 9, or a combination of the integers from 2 to 9 without any separators. If the combination string contains 0 or 1, other permission flags will not work.*

Usage Example

1) Set the owner password to "foxit" and permission flags to "allow printing with high resolution" and "allow commenting in the document" (-mode 1 -sop "foxit" -p 35)

```
fpdfsecure -i c:\input\1.pdf -o d:\output\1_secure.pdf -mode 1 -sop "foxit" -p 35
fpdfsecure -i test\2.pdf -o output\2_secure.pdf -mode 1 -sop "foxit" -p 35
fpdfsecure -i c:\input -o d:\output -mode 1 -sop "foxit" -p 35
fpdfsecure -i test -o output -mode 1 -sop "foxit" -p 35
fpdfsecure -i "c:\input\*.pdf" -o d:\output -mode 1 -sop "foxit" -p 35
```

2) Set the user password to "123", owner password to "456" and permission flags to "allow filling in a form" and "allow modifying document." (-mode 1 -sup "123" -sop "456" -p 47)

```
fpdfsecure -i c:\input\1.pdf -o d:\output\1_secure.pdf -mode 1 -sup "123" -sop "456" -p 47
fpdfsecure -i test\2.pdf -o output\2_secure.pdf -mode 1 -sup "123" -sop "456" -p 47
fpdfsecure -i c:\input -o d:\output -mode 1 -sup "123" -sop "456" -p 47
fpdfsecure -i test -o output -mode 1 -sup "123" -sop "456" -p 47
fpdfsecure -i "c:\input\*.pdf" -o d:\output -mode 1 -sup "123" -sop "456" -p 47
```

3) Use the "public.cer" file to encrypt PDF file(s), and set the permission flag to "allow nothing" (-mode 2 -cert "public.cer" -p 0)

```
fpdfsecure -i c:\input\1.pdf -o d:\output\1_secure.pdf -mode 2 -cert "public.cer" -p 0
fpdfsecure -i test\2.pdf -o output\2_secure.pdf -mode 2 -cert "public.cer" -p 0
fpdfsecure -i c:\input -o d:\output -mode 2 -cert "public.cer" -p 0
fpdfsecure -i test -o output -mode 2 -cert "public.cer" -p 0
fpdfsecure -i "c:\input\*.pdf" -o d:\output -mode 2 -cert "public.cer" -p 0
```

4) Use the "private.pfx" file to encrypt PDF file(s), set the password of its private key to "123", and set the permission flag to "allow everything" (-mode 2 -cert "private.pfx" -certpwd "123" -p 1)

```
fpdfsecure -i c:\input\1.pdf -o d:\output\1_secure.pdf -mode 2 -cert "private.pfx" -certpwd "123" -p
1
fpdfsecure -i test\2.pdf -o output\2_secure.pdf -mode 2 -cert "private.pfx" -certpwd "123" -p 1
```

```
fpdfsecure -i c:\input -o d:\output -mode 2 -cert "private.pfx" -certpwd "123" -p 1
fpdfsecure -i test -o output -mode 2 -cert "private.pfx" -certpwd "123" -p 1
fpdfsecure -i "c:\input\*.pdf" -o d:\output -mode 2 -cert "private.pfx" -certpwd "123" -p 1
```

5) Use the certificate named "Foxit" in Windows certificate store to encrypt PDF file(s), set the system certificate store to "CA", and set the permission flags to "allow printing with low resolution" and "allow printing with high resolution". (-mode 2 -certname "Foxit" -certstore 3 -p 23)

```
fpdfsecure -i c:\input\1.pdf -o d:\output\1_secure.pdf -mode 2 -certname "Foxit" -certstore 3 -p 23
fpdfsecure -i test\2.pdf -o output\2_secure.pdf -mode 2 -certname "Foxit" -certstore 3 -p 23
fpdfsecure -i c:\input -o d:\output -mode 2 -certname "Foxit" -certstore 3 -p 23
fpdfsecure -i test -o output -mode 2 -certname "Foxit" -certstore 3 -p 23
fpdfsecure -i "c:\input\*.pdf" -o d:\output -mode 2 -certname "Foxit" -certstore 3 -p 23
```

3.13.3.6 Security handler setting (-ea)

- The optional argument (**-ea**) is used to set a security handler to encrypt the output PDF file. The value should be 1, 2 or 3, and the default is 1. Foxit PDFSecure supports the following three security handler.

  - **-ea 1**: 128-bit AES encryption.
  - **-ea 2**: 256-bit AES encryption.
  - **-ea 3**: 128-bit RC4 encryption.

Usage Example

1) Use 256-bit AES to encrypt the output PDF file(s) (-ea 2)

```
fpdfsecure -i c:\input\1.pdf -o d:\output\1_secure.pdf -mode 1 -sup "123" -ea 2
fpdfsecure -i test\2.pdf -o output\2_secure.pdf -mode 1 -sop "456" -ea 2
fpdfsecure -i c:\input -o d:\output -mode 1 -sup "123" -sop "456" -ea 2
fpdfsecure -i test -o output -mode 2 -cert "test.cer" -ea 2
fpdfsecure -i "c:\input\*.pdf" -o d:\output -mode 2 -certname "Foxit" -ea 2
```

3.13.3.7 Metadata encryption (-em)

- The optional argument (**-em**) is used to encrypt the metadata of the output PDF file.

Usage Example

1) Encrypt the metadata of the output PDF file(s) (-em)

```
fpdfsecure -i c:\input\1.pdf -o d:\output\1_secure.pdf -mode 1 -sup "123" -em
fpdfsecure -i test\2.pdf -o output\2_secure.pdf -mode 1 -sop "456" -em
fpdfsecure -i c:\input -o d:\output -mode 1 -sup "123" -sop "456" -em
fpdfsecure -i test -o output -mode 2 -cert "test.cer" -em
fpdfsecure -i "c:\input\*.pdf" -o d:\output -mode 2 -certname "Foxit" -em
```

3.13.3.8 Read an Encrypted PDF file (-op)

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

Note

- *If the input PDF file is protected by a user password or an owner password, you need to provide the corresponding password using the option "-op" to change the document's security settings.*
- *If the input PDF file is protected by a user password and an owner password, the owner password specified by "-op" option is required to change the document's security settings.*

Usage Example

1) Specify the password for an input PDF file protected by a user password or an owner password (-op "123")

```
fpdfsecure -i c:\input\1.pdf -o d:\output\1_secure.pdf -mode 1 -sup "111" -op "123"
fpdfsecure -i test\2.pdf -o output\2_secure.pdf -mode 1 -sop "222" -op "123"
fpdfsecure -i c:\input\1.pdf -o d:\output\1_secure.pdf -mode 1 -sup "111" -sop "222" -op "123"
fpdfsecure -i test\2.pdf -o output\2_secure.pdf -mode 2 -cert "test.cer" -op "123"
fpdfsecure -i test\2.pdf -o output\2_secure.pdf -mode 2 -certname "Foxit" -op "123"
```

2) Specify the owner password for all input PDF files that have been protected by the same user password "123" and same owner password "welcome" (-op "welcome")

```
fpdfsecure -i c:\input -o d:\output -mode 1 -sup "111" -op "welcome"
fpdfsecure -i test -o output -mode 1 -sop "222" -op "welcome"
fpdfsecure -i "c:\input\*.pdf" -o d:\output -mode 1 -sup "123" -sop "456" -op "welcome"
fpdfsecure -i test -o output -mode 2 -cert "test.pfx" -certpwd "123456" -op "welcome"
```

fpdfsecure -i test\2.pdf -o output\2_secure.pdf -mode 2 -certname "Foxit" -op "welcome"

Note *It only supports typing one value for the argument (**-op**). Only files with the same password can be processed together and files with different password need to be processed separately.*

3.13.3.9 List the permission flags (-lp)

- The optional argument (**-lp**) is used to list the user's permissions to the input PDF designated by the **-i** option. The user's permissions to a PDF can be listed with the option "-lp" as long as you can open the PDF document, so if the input PDF file is protected by a user password, you need to provide the user password using the option "-op", or if it is protected by a certificate, the permissions can be listed only if the certificate is installed.

  Note *The argument (**-lp**) can only be used with **-i**, **-mode**, **-op**, **-log** and **-l** options. Other arguments will be ignored if set. The value of **-i** should be a PDF file, and the value of **-mode** should be 0.*

Usage Example

1) List the user's permissions to the document (foxit.pdf) (-i c:\foxit.pdf -mode 0 -lp)

   fpdfsecure -i c:\foxit.pdf -mode 0 -lp

2) List the user's permissions to the document (user_manual.pdf) and specify the user password (-i c:\user_manual.pdf -mode 0 -op "123" -lp)

   fpdfsecure -i c:\user_manual.pdf -mode 0 -op "123" -lp

3.13.3.10 Document Metadata Settings (-title, -subject, -keywords, -author, -creator)

a) Title (-title)

- The optional argument (**-title**) is used to set title of PDF files.

Usage Example

1) Set document title to "Foxit PDF Toolkit" (-title "Foxit PDF Toolkit")

   fpdfsecure -i c:\input\1.pdf -o d:\output\1_secure.pdf -mode 1 -sup "123" -title "Foxit PDF Toolkit"
   fpdfsecure -i test\2.pdf -o output\2_secure.pdf -mode 1 -sop "456" -title "Foxit PDF Toolkit"
   fpdfsecure -i c:\input -o d:\output -mode 1 -sup "123" -sop "456" -title "Foxit PDF Toolkit"

```
fpdfsecure -i test -o output -mode 2 -cert "test.cer" -title "Foxit PDF Toolkit"
fpdfsecure -i "c:\input\*.pdf" -o d:\output -mode 2 -certname "Foxit" -title "Foxit PDF Toolkit"
```

b) Subject (-subject)

- The optional argument (**-subject**) is used to set subject of PDF files.

Usage Example

1) Set document subject to "PDF Toolkit" (-subject "PDF Toolkit")

```
fpdfsecure -i c:\input\1.pdf -o d:\output\1_secure.pdf -mode 1 -sup "123" -subject "PDF Toolkit"
fpdfsecure -i test\2.pdf -o output\2_secure.pdf -mode 1 -sop "456" -subject "PDF Toolkit"
fpdfsecure -i c:\input -o d:\output -mode 1 -sup "123" -sop "456" -subject "PDF Toolkit"
fpdfsecure -i test -o output -mode 2 -cert "test.cer" -subject "PDF Toolkit"
fpdfsecure -i "c:\input\*.pdf" -o d:\output -mode 2 -certname "Foxit" -subject "PDF Toolkit"
```

c) Keywords (-keywords)

- The optional argument (**-keywords**) is used to set keywords of PDF files.

Usage Example

1) Set document keywords to "toolkit" (-keywords "toolkit")

```
fpdfsecure -i c:\input\1.pdf -o d:\output\1_secure.pdf -mode 1 -sup "123" -keywords "toolkit"
fpdfsecure -i test\2.pdf -o output\2_secure.pdf -mode 1 -sop "456" -keywords "toolkit"
fpdfsecure -i c:\input -o d:\output -mode 1 -sup "123" -sop "456" -keywords "toolkit"
fpdfsecure -i test -o output -mode 2 -cert "test.cer" -keywords "toolkit"
fpdfsecure -i "c:\input\*.pdf" -o d:\output -mode 2 -certname "Foxit" -keywords "toolkit"
```

d) Author (-author)

- The optional argument (**-author**) is used to set author of PDF files.

Usage Example

1) Set document author to "Jessie" (-author "Jessie")

```
fpdfsecure -i c:\input\1.pdf -o d:\output\1_secure.pdf -mode 1 -sup "123" -author "Jessie"
```

```
fpdfsecure -i test\2.pdf -o output\2_secure.pdf -mode 1 -sop "456" -author "Jessie"
fpdfsecure -i c:\input -o d:\output -mode 1 -sup "123" -sop "456" -author "Jessie"
fpdfsecure -i test -o output -mode 2 -cert "test.cer" -author "Jessie"
fpdfsecure -i "c:\input\*.pdf" -o d:\output -mode 2 -certname "Foxit" -author "Jessie"
```

e) Creator (-creator)

- The optional argument (**-creator**) is used to set file creation application information of PDF files.

Usage Example

1) Set document creator to "Foxit Reader" (-creator "Foxit Reader")

```
fpdfsecure -i c:\input\1.pdf -o d:\output\1_secure.pdf -mode 1 -sup "123" -creator "Foxit Reader"
fpdfsecure -i test\2.pdf -o output\2_secure.pdf -mode 1 -sop "456" -creator "Foxit Reader"
fpdfsecure -i c:\input -o d:\output -mode 1 -sup "123" -sop "456" -creator "Foxit Reader"
fpdfsecure -i test -o output -mode 2 -cert "test.cer" -creator "Foxit Reader"
fpdfsecure -i "c:\input\*.pdf" -o d:\output -mode 2 -certname "Foxit" -creator "Foxit Reader"
```

3.13.3.11 Recursion Depth of Sub-folders (-r)

- The optional argument (**-r**) indicates the recursion depth of sub-folders, which is valid only when the input is a folder or a path that includes wildcard characters like "c:\input*.pdf". By default, the recursion depth is 1, so the sub-folders will not be processed. For more details about this argument, please refer to section 3.13.2 "Command Line Summary".

Usage Examples

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

```
fpdfsecure -i c:\input -o d:\output -mode 1 -sup "111" -r
fpdfsecure -i test -o output -mode 1 -sop "222" -r
fpdfsecure -i "c:\input\*.pdf" -o d:\output -mode 1 -sup "123" -sop "456" -r 0
fpdfsecure -i test -o output -mode 2 -cert "test.cer" -r 0
fpdfsecure -i c:\input -o d:\output -mode 2 -certname "Foxit" -r 0
```

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

```
fpdfsecure -i c:\input -o d:\output -mode 1 -sup "111" -r 1
fpdfsecure -i test -o output -mode 1 -sop "222" -r 1
```

```
fpdfsecure -i "c:\input\*.pdf" -o d:\output -mode 1 -sup "123" -sop "456" -r 1
fpdfsecure -i test -o output -mode 2 -cert "test.cer" -r 1
fpdfsecure -i c:\input -o d:\output -mode 2 -certname "Foxit" -r 1
```

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

```
fpdfsecure -i c:\input -o d:\output -mode 1 -sup "111"
fpdfsecure -i test -o output -mode 1 -sop "222"
fpdfsecure -i "c:\input\*.pdf" -o d:\output -mode 1 -sup "123" -sop "456"
fpdfsecure -i test -o output -mode 2 -cert "test.cer"
fpdfsecure -i c:\input -o d:\output -mode 2 -certname "Foxit"
```

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

```
fpdfsecure -i c:\input -o d:\output -mode 1 -sup "111" -r 2
fpdfsecure -i test -o output -mode 1 -sop "222" -r 2
fpdfsecure -i "c:\input\*.pdf" -o d:\output -mode 1 -sup "123" -sop "456" -r 2
fpdfsecure -i test -o output -mode 2 -cert "test.cer" -r 2
fpdfsecure -i c:\input -o d:\output -mode 2 -certname "Foxit" -r 2
```

3.13.3.12 Multi-thread Support (-t)

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

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

Usage Example

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

```
fpdfsecure -i c:\input -o d:\output -mode 1 -sup "111" -t 3
fpdfsecure -i test -o output -mode 1 -sop "222" -t 3
fpdfsecure -i "c:\input\*.pdf" -o d:\output -mode 1 -sup "123" -sop "456" -t 3
fpdfsecure -i test -o output -mode 2 -cert "test.cer" -t 3
fpdfsecure -i c:\input -o d:\output -mode 2 -certname "Foxit" -t 3
```

3.13.3.13 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 argument (**-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 3.13.2 "Command Line Summary".

Usage Example

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

```
fpdfsecure -i c:\input -o d:\output -mode 1 -sup "111" -log d:\output\pdfsecure.log -l 3
fpdfsecure -i test -o output -mode 1 -sop "222" -log d:\output\pdfsecure.log -l 3
fpdfsecure -i test -o output -mode 1 -sup "123" -sop "456" -log d:\output\pdfsecure.log -l 3
fpdfsecure -i "c:\input\*.pdf" -o d:\output -mode 2 -cert "test.cer" -log d:\output\pdfsecure.log -l 3
fpdfsecure -i c:\input -o d:\output -mode 2 -certname "Foxit" -log d:\output\pdfsecure.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 the **<licensee>** is the licensee name designated by the users.

Usage Example

1) Register the PDFSecure tool with the code "xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx" and the licensee "Foxit" (-register xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx Foixt)

```
fpdfsecure -register xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx Foixt
```

c) License agreement (-license)

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

Usage Example

1) Print the license agreement (-license)

```
fpdfsecure -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)

```
fpdfsecure -version
fpdfsecure -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)

```
fpdfsecure -help
fpdfsecure -h
```

f) Copyright information (-copyright)

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

Usage Example

1) Print the copyright information (-copyright)

```
fpdfsecure -copyright
```

3.14 PDF2Word

3.14.1 Basic Syntax

fpdf2word *<-i <srcfile/srcfolder>> <-o <destfile/destfolder>> [-range <page range>]*
[-op <password>] [-r [recursion]] [-t <thread>] [-log <logfile>] [-l <level>]
fpdf2word *-register <code> <licensee>*
fpdf2word *-license*
fpdf2word *-version/-v*
fpdf2word *-help/-h*
fpdf2word *-copyright*

Note:

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

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

3.14.2 Command Line Summary

Note *For some arguments whose values are strings, users can choose whether to add quotation marks (" ") to the strings. In the manual we have added notes where adding quotation marks (" ") is required.*

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

Option	Parameter	Description
-o	*<-o <string>>* *e.g.* *-o d:\output\1.docx* *-o d:\output*	Specifies the path of the output Word file or folder. ▪ If the input is a single PDF file, the output should be a single Word file, (e.g. *-o d:\output\1.docx*). ▪ 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.
-range	*[-rang <String>]* *e.g.* *-range "1,5,9"* *-range "all"* *-range "even"* *-range "odd"* *-range "2-10,30"* *-range "10,50-"* *-range "odd,100-"*	Specifies a page range to convert. By default, all of the pages will be converted. ▪ Converts page 1,5, and 9: **-range "1,5,9"** ▪ Converts all pages: **-range "all"** ▪ Converts all even pages: **-range "even"** ▪ Converts all odd pages: **-range "odd"** ▪ Converts pages in the range from 2-10 and page 30: **-range "2-10,30"** ▪ Converts page 10 and all pages in the range from 50 to the last page: **-range "10,50-"** ▪ Converts all odd pages and all pages in the range from 100 to the last page: **-range "odd,100-"**
-op	*[-op<string>]* *e.g.* *-op 123* *-op welcome*	Specifies the password for the input file. Not required if the input file is not password protected.
-r	*[-r [integer]]* *e.g.* *-r*	Specifies the number of layers to recurse when the input is a folder. • **-r 0 <-r>**: searches the full folders.

Option	Parameter	Description
	-r 0 *-r 1* *-r 2* …	• **-r 1**: searches only the current folder. • **-r 2**: searches the current folder and its sub-folders … **Note** ▪ If no integer value is specified, or if the integer value is 0, then full folders will be searched. By default, the number of layers to recurse is 1, which means that only the current folder will be searched and not sub-folders. ▪ The input PDF file or folder will be skipped if it is secured and the messages will be displayed.
-t	*[-t <integer>]* *e.g.* *-t 1* *-t 2* …	Specifies the number of CPU threads to use. The default value is 1.
-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 messages only concerning out of memory errors. • **-l 2**: logs failure messages concerning the errors caused during execution or those returned from underlying libraries, as well as those for level 1. • **-l 3**: logs warning messages concerning the PDF files that are overwritten, as well as those for level 2. • **-l 4**: logs informational messages, as well as those for level 3. **Note** The argument (**-l**) is valid only when (**-log**) is used.
-register	*[-register <String> <String>]*	Registers the command line tool.

Option	Parameter	Description
	-register <code> <licensee> *e.g.* *-register* xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx Foxit	▪ **<code>**: the activation code from Foxit. ▪ **<licensee>**: the Licensee name designated by the users.
-help/-h	*[-help]/[-h]* *e.g.* *-help* *-h*	Prints the usage information.
-version/-v	*[-version]/[-v]* *e.g.* *-version* *-v*	Prints the version information.
-license	*[-license]* *e.g.* *-license*	Prints the license agreement.

3.14.3 Basic Usage

3.14.3.1 Input and Output (-i, -o)

a) Input (-i)

- The input should be a single PDF file or a folder. Users are not able to input multiple PDF files or folders, as well as a mixture composed 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 if the input file is in the current working folder. Users can input just the name of the PDF file or folder, instead of an absolute path. For example:

  -i test\1.pdf ("test" folder 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, it is recommended to enclose the input files with quotation marks (" "). In this manual, we add (" ") whenever the input files contain wildcard characters.*

Note *If the input paths or file names contain spaces, you must enclose the input files with quotation marks (" "), such as -i "c:\input file\1.pdf", -i "test\user manual.pdf". This rule is also used for some other arguments whose values are strings.*

b) Output (-o)

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

  -o d:\output\output.docx (a single Word 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. Users can input just the name of the output text file or folder, instead of an absolute path. For example:

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

Usage Examples

1) Convert a single PDF file to Word file:

```
fpdf2word -i c:\input\1.pdf -o d:\output\1.docx
fpdf2word -i test\2.pdf -o output\2.docx
```

2) Convert the PDF files in a folder to Word files:

```
fpdf2word -i c:\input -o d:\output
fpdf2word -i test -o output
fpdf2word -i c:\input\*.pdf -o d:\output
fpdf2word -i "test\*.pdf" -o output
```

3.14.3.2 Page range to convert (-range)

- The optional argument (**-range**) is used to specify a page range to convert. If this argument is not set, all of the pages will be converted. For more details about this argument, please refer to section 3.14.2 "Command Line Summary".

Usage Example

1) Convert pages 2,3 and 8 (-range "2,3,8")

```
fpdf2word -i c:\input\1.pdf -o d:\output\1.docx -range "2,3,8"
fpdf2word -i test\2.pdf -o output\2.docx -range "2,3,8"
fpdf2word -i c:\input -o d:\output -range "2,3,8"
fpdf2word -i test -o output -range "2,3,8"
fpdf2word -i "c:\input\*.pdf" -o d:\output -range "2,3,8"
```

2) Convert all even pages (-range "even")

```
fpdf2word -i c:\input\1.pdf -o d:\output\1.docx -range "even"
fpdf2word -i test\2.pdf -o output\2.docx -range "even"
fpdf2word -i c:\input -o d:\output -range "even"
fpdf2word -i test -o output -range "even"
fpdf2word -i "c:\input\*.pdf" -o d:\output -range "even"
```

3) Convert pages in the range from 2-10 and page 30 (-range "2-10,30")

```
fpdf2word -i c:\input\1.pdf -o d:\output\1.docx -range "2-10,30"
fpdf2word -i test\2.pdf -o output\2.docx -range "2-10,30"
fpdf2word -i c:\input -o d:\output -range "2-10,30"
fpdf2word -i test -o output -range "2-10,30"
fpdf2word -i "c:\input\*.pdf" -o d:\output -range "2-10,30"
```

4) Convert all odd pages and all pages in the range from 100 to the last page (-range "odd,100-")

```
fpdf2word -i c:\input\1.pdf -o d:\output\1.docx -range "odd,100-"
fpdf2word -i test\2.pdf -o output\2.docx -range "odd,100-"
fpdf2word -i c:\input -o d:\output -range "odd,100-"
fpdf2word -i test -o output -range "odd,100-"
fpdf2word -i "c:\input\*.pdf" -o d:\output -range "odd,100-"
```

3.14.3.3 Password for the input file (-op)

- The optional argument (**-op**) indicates the password for a password-protected input PDF file. It is not required if the input file is not password protected. There are two kinds of passwords, one is user password, and the other is owner password.

  User password is also known as "open password", which protects the document against unauthorized opening and reading.

  Owner password is also known as "master password", which protects the document against unauthorized editing, printing, changing, and copying. It grants users full access to the document. With the owner password, the document can not only be opened and read, but also be allowed to change the document's security settings. This means that even if the permission of modifying the PDF file was restricted, you can still modify it with the owner password.

  Note

  - *If the input PDF file is protected by a user password, you need to provide the password using the option "-op" to process the PDF file.*
  - *If the input PDF file is protected by an owner password, there are two circumstances. If the permissions of changing the PDF file were not restricted, you can process the PDF file without providing the owner password; otherwise, if the permissions of changing the PDF file were restricted, you need to provide the password using the option "-op" to process the PDF file.*
  - *If the input PDF file is protected by a user password and an owner password, there are also two circumstances. If the permissions of changing the PDF file were not restricted, you can provide either of the passwords using the option "-op" to process the PDF file; otherwise, if the permissions of changing the PDF file were restricted, you need to provide the owner password using "-op" option to process the PDF file.*

Usage Example

1) Specify the password for a password-protected input PDF file (-op 123)

```
fpdf2word -i c:\input\1.pdf -o d:\output\1.docx -op 123
fpdf2word -i test\2.pdf -o output\2.docx -op 123
```

2) Specify the password for all password-protected input PDF files in a given folder that share the same password (-op welcome)

```
fpdf2word -i c:\input -o d:\output -op welcome
```

```
fpdf2word -i test -o output -op welcome
fpdf2word -i "c:\input\*.pdf" -o d:\output -op welcome
```

Note *It only supports typing one value for the argument (**-op**). Only files with the same password can be processed together and files with different open password need to be processed separately.*

3.14.3.4 Recursion Depth of Sub-folders (-r)

- The optional argument (**-r**) indicates the recursion depth of sub-folders, which is valid only when the input is a folder or a path that includes wildcard characters like "c:\input*.pdf". By default, the recursion depth is 1, so the sub-folders will not be processed. For more details about this argument, please refer to section 3.14.2 "Command Line Summary".

Usage Examples

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

```
fpdf2word -i c:\input -o d:\output -r
fpdf2word -i test -o output -r
fpdf2word -i "c:\input\*.pdf" -o d:\output -r
fpdf2word -i c:\input -o d:\output -r 0
fpdf2word -i test -o output -r 0
fpdf2word -i "c:\input\*.pdf" -o d:\output -r 0
```

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

```
fpdf2word -i c:\input -o d:\output -r 1
fpdf2word -i test -o output -r 1
fpdf2word -i "c:\input\*.pdf" -o d:\output -r 1
```

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

```
fpdf2word -i c:\input -o d:\output
fpdf2word -i test -o output
fpdf2word -i "c:\input\*.pdf" -o d:\output
```

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

```
fpdf2word -i c:\input -o d:\output -r 2
fpdf2word -i test -o output -r 2
```

fpdf2word -i "c:\input*.pdf" -o d:\output -r 2

3.14.3.5 Multi-thread Support (-t)

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

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

Usage Example

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

   fpdf2word -i c:\input -o d:\output -t 3
   fpdf2word -i test -o output -t 3
   fpdf2word -i "c:\input*.pdf" -o d:\output -t 3

3.14.3.6 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 argument (**-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 3.14.2 "Command Line Summary".

Usage Example

1) Save the log file to "d:\output\pdf2word.log" and set the log level to 3 (-log d:\output\pdf2word.log -l 3)

   fpdf2word -i c:\input -o d:\output -log d:\output\pdf2word.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 the **<licensee>** is the licensee name designated by the users.

Usage Example

1) Register the pdf2word tool with the code "xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx" and the licensee "Foxit" (-register xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx Foixt)

fpdf2word -register xxxxx-xxxxx-xxxxx-xxxxx-xxxxx-xxxxx Foixt

c) License agreement (-license)

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

Usage Example

1) Print the license agreement (-license)

fpdf2word -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)

fpdf2word -version
fpdf2word -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)

fpdf2word -help
fpdf2word -h

f) Copyright information (-copyright)

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

Usage Example

1) Print the copyright information (-copyright)

```
fpdf2word -copyright
```

3.15 RMS

3.15.1 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

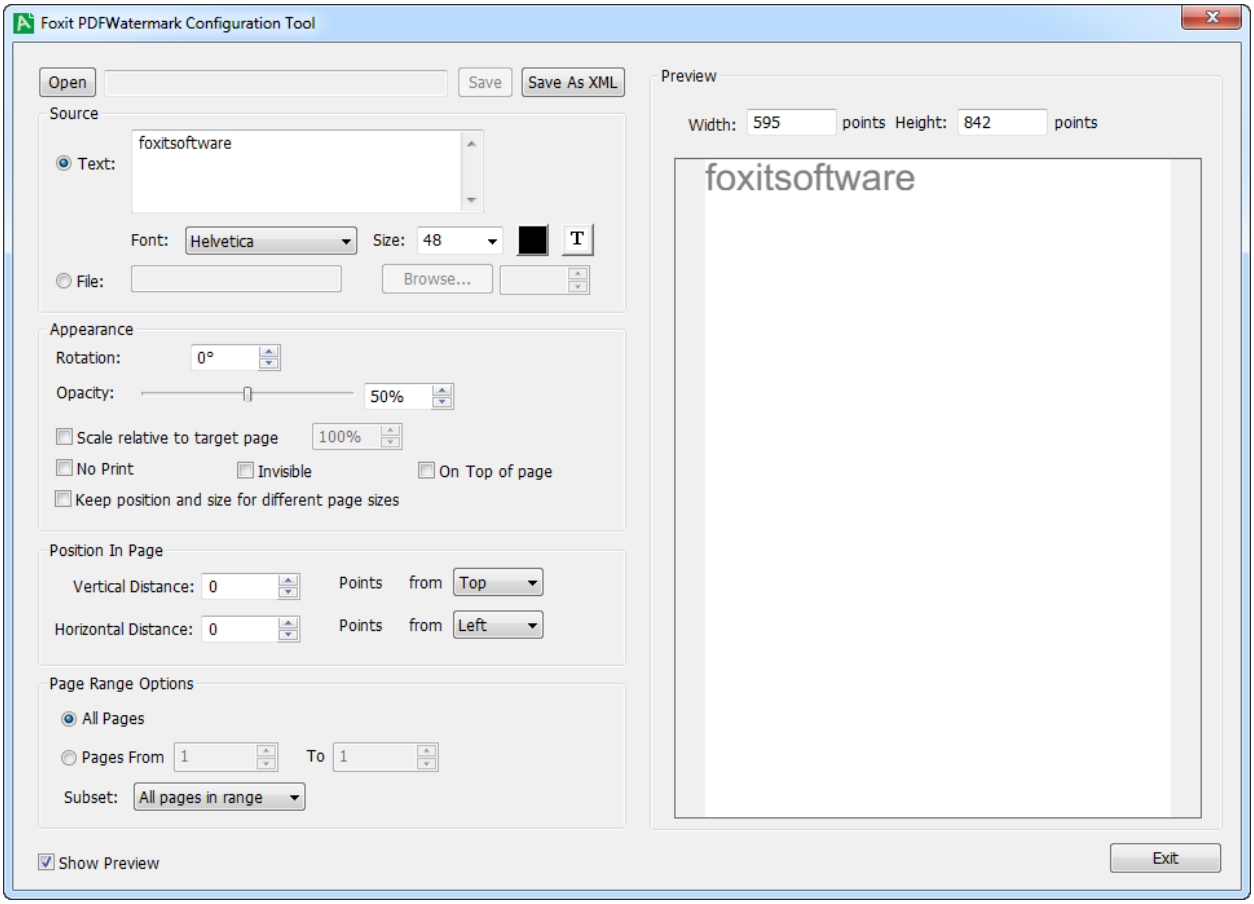
For the RMS tool, please go to the "rms" folder in the installation package for a more detailed introduction.

4 Foxit Configuration Tool

4.1 Watermark Configuration Tool

Foxit PDFWatermark Configuration Tool is used to set the properties of watermarks and then save them as a configuration file with the extension ".xml", which is required in the command line. The watermark settings contain four main parts: source, appearance, position 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 the Foxit PDFWatermark Configuration Tool, and then you can see the configuration screen as follows.



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/PDF 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 three types of watermarks: text, image and PDF.

Text Watermark

- **Text**

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

PDF Watermark

- **File**

Users can browse their computer and choose a PDF file. The preview window will display the first page of the PDF file, you can click the selection box on the right side of the "**Browse…**" button to preview other pages and decide which PDF page will be used to set as PDF watermark.

4.1.1.2 Appearance

This section introduces how to set the appearance for text/image/PDF watermarks.

- **Rotation**

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

- **Opacity**

  This option is used to set the transparence for text/image/PDF watermarks. The value is from 0 to 100 (0 meaning invisible, and 100 meaning visibly 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/PDF watermark in relation to the PDF page's 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/PDF watermarks 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/PDF watermarks 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/PDF 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** positioning. The distance unit is points. Users can set the relative benchmark for the distance, such as Top, Bottom, Left, Right and Center.

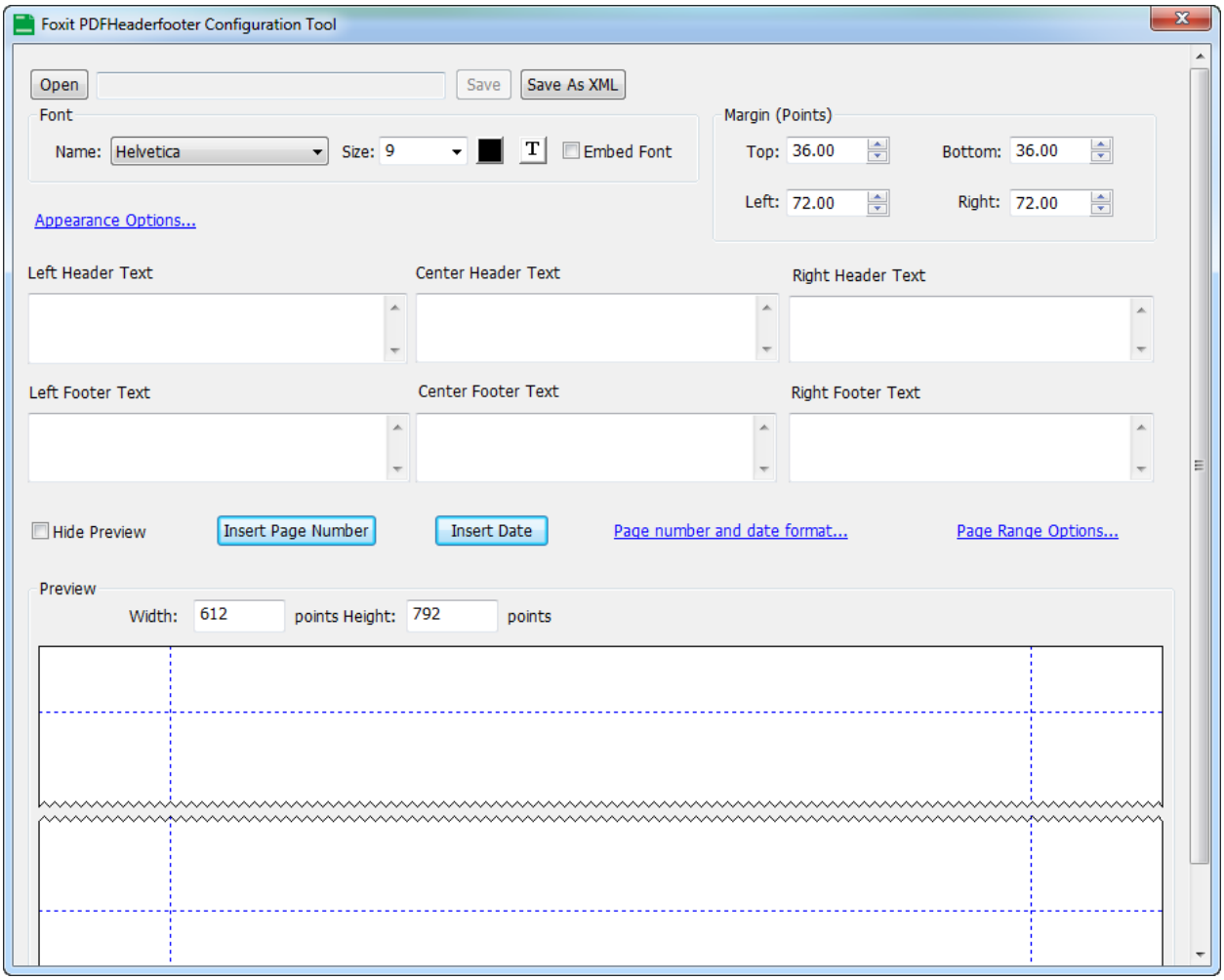
4.1.1.4 Page Range Options

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 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 in 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.



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

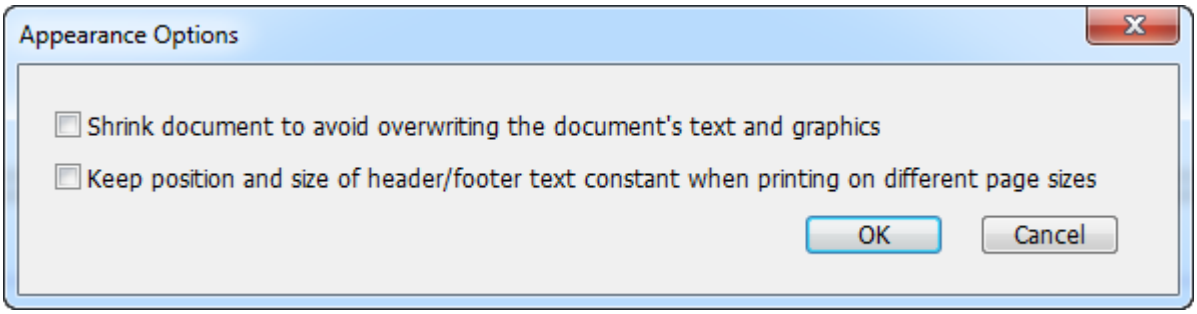
The font option allows users to choose the font name, font size and font color from the drop-down menus, to click the T icon to set underline for text, and to check the "**Embed Font**" to set the text font as embedded font.

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.



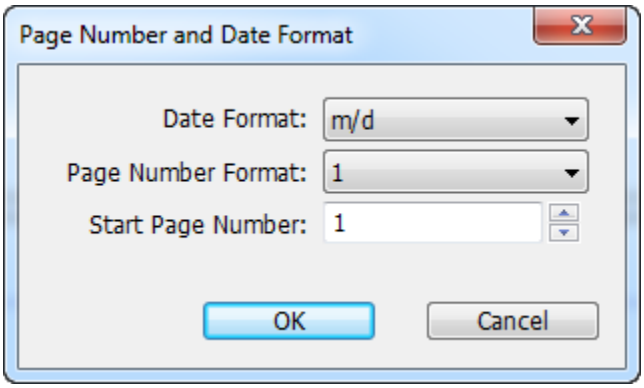
Appearance Options

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 the header/footer.

4.2.1.5 Page Number and Date Format

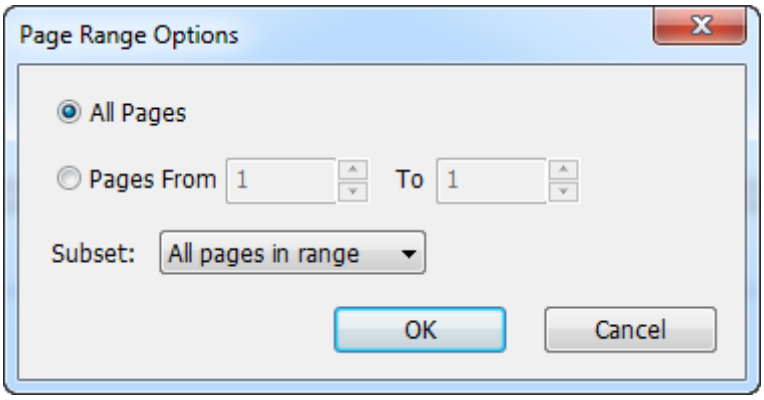
Click "**Page Number and Date Format**" to open the setting window as shown in the following figure. Users can set the Date Format, Page Number Format and Start Page Number. After setting, click the "**Insert Page Number**" button to insert page number to the desired text-input box, and click the "**Insert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



Page Number and Date Format

4.2.1.6 Page Range Options

Click the "**Page Range Options**" to open the settings 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.



Page Range Options

5 Working with API

Foxit PDF Toolkit provides another way for users who want to perform PDF manipulation through API.

Note *To integrate the Foxit PDF Toolkit into your own applications with API, please contact Foxit sales team to purchase Enterprise License.*

Each module offers a simple-to-use API. Four functions are required for developers who want to integrate the Foxit PDF Toolkit into their own applications.

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

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

The parameter "**key**" is the path of the license file ("**ftlkey.txt**", generated in the installation path after registering the module 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 module.

```
int FXT_Image2PDFRun(const wchar_t* commandline, FXT_CallbackFun callback, void* userData = 0);
int FXT_PDF2ImgRun(const wchar_t* commandline, FXT_CallbackFun callback,  void* userData = 0);
int FXT_Office2PDFRun(const wchar_t * commandline, FXT_CallbackFun callback, void* userData = 0);
int FXT_WatermarkRun(const wchar_t * commandline, FXT_CallbackFun callback, void* userData = 0);
int FXT_HeaderFooterRun(const wchar_t * commandline, FXT_CallbackFun callback, void* userData = 0);
int FXT_OptimizerRun(const wchar_t * commandline, FXT_CallbackFun callback, void* userData = 0);
int FXT_RedactorRun(const wchar_t * commandline, FXT_CallbackFun callback, void* userData = 0);
int FXT_MetadataRun(const wchar_t * commandline, FXT_CallbackFun callback, void* userData = 0);
int FXT_PDF2TextRun(const wchar_t * commandline, FXT_CallbackFun callback, void* userData = 0);
int FXT_Text2PDFRun(const wchar_t * commandline, FXT_CallbackFun callback,  void* userData = 0);
int FXT_Html2PDFRun(const wchar_t * commandline, FXT_CallbackFun callback,  void* userData = 0);
int FXT_PDFPpoRun(const wchar_t* commandline, FXT_CallbackFun callback,  void* userData = 0);
int FXT_PDFSecureRun(const wchar_t* commandline, FXT_CallbackFun callback,  void* userData = 0);
int FXT_PDF2WordRun(const wchar_t* commandline, FXT_CallbackFun callback,  void* userData = 0);
```

Select the corresponding function of the module. 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.

Please note that only the FXT_PDF2WordRun function is not thread safe. It means only one thread in a process can call this function while other threads must be blocked.

Third, declare the callback function.

```
typedef wchar_t*(*FXT_CallbackFun)(void* userData, int mode, wchar_t* 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(L"-i d:\\input -o d:\\output\\output.pdf", myCallBack, NULL);

	FXT_DestroyLibrary();
}
```

This application converts all the image files in the "d:\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 the 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::wstring strCommandline = GetCommandLineW();
	std::wstring::size_type pos = strCommandline.find(L".exe");
	strCommandline = strCommandline.substr (pos+6);

	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::wstring input_folder = L"d:\\samples\\input";//or image file: input.jpg...
	std::wstring output_folder = L"d:\\samples\\output";//or PDF file: output.pdf...
	std::wstring set_password = L"secret";
	std::wstring log_file = L"d:\\samples\\output\\image2pdf.log";

	// log level.
	int log_level = 4;

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

	// set title of PDF files.
	std::wstring title = L"Foxit PDF Toolkit User Manual";

	// set subject of PDF files.
	std::wstring subject = L"Foxit PDF Toolkit";

	// set keywords of PDF files.
	std::wstring keywords = L"PDF Toolkit";

	// set author of PDF files.
	std::wstring author = L"Jessie";

	// set file creation application information of PDF files.
```

```
	std::wstring creator = L"Foxit Reader";

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

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

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

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

	if(!title.empty())
		strCommandline.append(L"-title " ).append(title).append(L" ");

	if(!subject.empty())
		strCommandline.append(L"-subject " ).append(subject).append(L" ");

	if(!keywords.empty())
		strCommandline.append(L"-keywords " ).append(keywords).append(L" ");

	if(!author.empty())
		strCommandline.append(L"-author " ).append(author).append(L" ");

	if(!creator.empty())
		strCommandline.append(L"-creator " ).append(creator).append(L" ");

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

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

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

	if (width > 0 && height > 0)
	{
		wmemset(temp, 0 , MAX_LEGHT);
		_itow(width, temp ,DATA_RADIX);
		strCommandline.append(L"-width ").append(temp).append(L" ");

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

```
                }
        }

        if (depth >= 0)
        {
                wmemset(temp, 0 , MAX_LEGHT);
                _itow(depth, temp ,DATA_RADIX);
                strCommandline.append(L"-depth ").append(temp).append(L" ");
        }

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

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

        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 Foxit PDF SDK Library
}
else if (ret == FXT_ERROR_ERROR) {
       // Failed to convert image into pdf.
}
else {
       // Other error
}
```

A non-zero value indicates that an error was encountered. You can find the list for all error codes 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++
wchar_t* MyCallback(void* userData, int mode, wchar_t* msg, bool* isStop) {
       if (mode == CALLBACK_PDFTOOL_RUN_ERROR) {
              wcout << L"Error: " << msg << endl;
       }
       else if (mode == CALLBACK_PDFTOOL_PARAM_ERROR) {
              wcout << msg;
       }
       else if (mode == CALLBACK_PDFTOOL_MSG) {
              wcout << msg;
       }
       else if (mode == CALLBACK_PDFTOOL_PASSWORD) {
              static wstring password;
              wcin >> password;
              return (wchar_t*)password.c_str();
       }
       return 0;
}
```

5.2 PDF2Image

5.2.1 Working with PDF2Image API

The following is the simplest application that can be built using PDF2Image API:

```
// Using C/C++
void main(int argc, char* argv[])
```

```
{
	int ret = FXT_InitLibrary(L"license_key", 0);
	FXT_PDF2ImgRun(L"-i d:\\input -o d:\\output", myCallBack, NULL);

	FXT_DestroyLibrary();
}
```

This application converts all the PDF files in the "d:\input" folder into image files under the "d:\output" folder 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 the command line. Building a command line application using PDF2Image 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::wstring strCommandline = GetCommandLineW();
	std::wstring::size_type pos = strCommandline.find(L".exe");
	strCommandline = strCommandline.substr (pos+6);

	FXT_PDF2ImgRun(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::wstring input_folder = L"d:\\samples\\input"; //or a PDF file: input.pdf
	std::wstring output_folder = L"d:\\samples\\output";
	std::wstring password = L"secret"; //password
	std::wstring log_file = L"d:\\samples\\output\\PDF2Image.log";

	// the page range to convert
	std::wstring range = L"all";

	// specify the output image format and the corresponding compression quality
	std::wstring ext = L"jpg 100";

	// use adam7 algorithm to generate the output image files
	std::wstring interlace = L"adam7";

	// page size of the output image file in pixels
	int width = 612, height = 792;

	// specify a clip region of the PDF page to be converted
	int clip = 100 120 500 600;
```

```
// the resolution (DPI:= Dots Pet Inch) for the output image file
int dpi = 150;

// specify the color space and color depth to render the output image file
std::wstring cs = L"rgb" 24;

// Set the alignment of the PDF page in the output image
std::wstring align = L"tr";

// Set the page rotation of the PDF to be rendered to the output image file
int rotate = 90;

// scale the page of the PDF to fit the page of the image (in either width or height).
bool fitpage = true;

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

// log level
int log_level = 4;

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

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

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

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

if(!range.empty())
       strCommandline.append(L"-range " ).append(range).append(L" ");

if(!ext.empty())
       strCommandline.append(L"-ext " ).append(ext).append(L" ");

if(!interlace.empty())
       strCommandline.append(L"-interlace " ).append(interlace).append(L" ");

if(!cs.empty())
       strCommandline.append(L"-cs " ).append(cs).append(L" ");

if(!align.empty())
       strCommandline.append(L"-align " ).append(align).append(L" ");

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

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

if (log_level > 0)
{
       _itow(log_level,temp,DATA_RADIX);
```

```
            strCommandline.append(L"-l ").append(temp).append(L" ");
        }

        if (depth >= 0)
        {
            wmemset(temp, 0 , MAX_LEGHT);
            _itow(depth, temp ,DATA_RADIX);
            strCommandline.append(L"-depth ").append(temp).append(L" ");
        }

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

        if (width >= 0)
        {
            wmemset(temp, 0, MAX_LEGHT);
            _itow(width, temp ,DATA_RADIX);
            strCommandline.append(L"-width ").append(temp).append(L" ");
        }

        if (height >= 0)
        {
            wmemset(temp, 0, MAX_LEGHT);
            _itow(height, temp ,DATA_RADIX);
            strCommandline.append(L"-height ").append(temp).append(L" ");
        }

        if (lower-left_X >= 0 && lower-left_y >= 0 && upper-right_x > 0 && upper-right_y > 0)
        {
            wmemset(temp, 0, MAX_LEGHT);
            _itow(clip, temp ,DATA_RADIX);
            strCommandline.append(L"-clip ").append(temp).append(L" ");
        }

        if (dpi >= 0)
        {
            wmemset(temp, 0, MAX_LEGHT);
            _itow(dpi, temp ,DATA_RADIX);
            strCommandline.append(L"-dpi ").append(temp).append(L" ");
        }

        if (rotate >= 0)
        {
            wmemset(temp, 0, MAX_LEGHT);
            _itow(rotate, temp ,DATA_RADIX);
            strCommandline.append(L"-rotate ").append(temp).append(L" ");
        }

        if(fitpage)    strCommandline.append(L"-fitpage").append(L" ");

        FXT_PDF2ImgRun(strCommandline.c_str(), myCallBack, NULL);

        FXT_DestroyLibrary() ;
}
```

5.2.2 Reporting Progress Messages and Errors

To find out if PDF2Image processing was successful, the application can query the status code returned by FXT_PDF2ImgRun().

For example,

```
int ret = FXT_PDF2ImgRun(...);
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 Foxit PDF SDK Library
}
else if (ret == FXT_ERROR_ERROR) {
	// Failed to convert PDF file to image file
}
else {
	// Other error
}
```

A non-zero value indicates that an error was encountered. You can find the list for all error codes 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_PDF2ImgRun(). The last parameter in FXT_PDF2ImgRun() 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++
wchar_t* MyCallback(void* userData, int mode, wchar_t* msg, bool* isStop) {
	if (mode == CALLBACK_PDFTOOL_RUN_ERROR) {
		wcout << L"Error: " << msg << endl;
	}
	else if (mode == CALLBACK_PDFTOOL_PARAM_ERROR) {
		wcout << msg;
	}
	else if (mode == CALLBACK_PDFTOOL_MSG) {
		wcout << msg;
	}
	else if (mode == CALLBACK_PDFTOOL_PASSWORD) {
		static wstring password;
		wcin >> password;
		return (wchar_t*)password.c_str();
	}
```

```
        return 0;
}
```

5.3 Office2PDF

5.3.1 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(L"-i d:\\input -o d:\\output", myCallBack, NULL);

        FXT_DestroyLibrary();
}
```

This application converts all the Microsoft Office files in the "d:\input" folder into PDF files under the "d:\output" folder 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 the 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::wstring strCommandline = GetCommandLineW();
        std::wstring::size_type pos = strCommandline.find(L".exe");
        strCommandline = strCommandline.substr (pos+6);

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

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

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

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

        // specify the conversion mode for Microsoft Excel files.
        int scale = 3;

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

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

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

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

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

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

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

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

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

        if(scale >= 0)
        {
                wmemset(temp, 0 , MAX_LEGHT);
                _itow(scale, temp ,DATA_RADIX);
                strCommandline.append(L"-scale ").append(temp).append(L" ");
        }

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

        FXT_DestroyLibrary();
}
```

5.3.2 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 Foxit PDF SDK Library
}
else {
	// Other error
}
```

A non-zero value indicates that an error was encountered. You can find the list for all error codes 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++
wchar_t* MyCallback(void* userData, int mode, wchar_t* msg, bool* isStop) {
	if (mode == CALLBACK_PDFTOOL_RUN_ERROR) {
		wcout << L"Error: " << msg << endl;
	}
	else if (mode == CALLBACK_PDFTOOL_PARAM_ERROR) {
		wcout << msg;
	}
	else if (mode == CALLBACK_PDFTOOL_MSG) {
		wcout << msg;
	}
	else if (mode == CALLBACK_PDFTOOL_PASSWORD) {
		static wstring password;
		wcin >> password;
		return (wchar_t*)password.c_str();
	}
	return 0;
}
```

5.4 PDFWatermark

5.4.1 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(L"-i d:\\input -o d:\\output –conf d:\\conf_wm.xml", myCallBack,
NULL);

	FXT_DestroyLibrary();
}
```

This application adds a watermark into PDF files. All the PDF files in the "d:\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 the 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::wstring strCommandline = GetCommandLineW();
	std::wstring::size_type pos = strCommandline.find(L".exe");
	strCommandline = strCommandline.substr (pos+6);

	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::wstring input_folder = L"d:\\samples\\input";//or PDF file: input.pdf...
	std::wstring output_folder = L"d:\\samples\\output";//or PDF file: output.pdf...
	std::wstring set_password = L"secret";
	std::wstring log_file = L"d:\\samples\\output\\watermark.log";
	std::wstring conf_file = L"d:\\samples\\conf\\conf_wm.xml";//the configuration file
	// set title of PDF files.
	std::wstring title = L"Foxit PDF Toolkit User Manual";
```

```
// set subject of PDF files.
std::wstring subject = L"Foxit PDF Toolkit";

// set keywords of PDF files.
std::wstring keywords = L"PDF Toolkit";

// set author of PDF files.
std::wstring author = L"Jessie";

// set file creation application information of PDF files.
std::wstring creator = L"Foxit Reader";

// log level
int log_level = 4;

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

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

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

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

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

if(!title.empty())
	strCommandline.append(L"-title " ).append(title).append(L" ");

if(!subject.empty())
	strCommandline.append(L"-subject " ).append(subject).append(L" ");

if(!keywords.empty())
	strCommandline.append(L"-keywords " ).append(keywords).append(L" ");

if(!author.empty())
	strCommandline.append(L"-author " ).append(author).append(L" ");

if(!creator.empty())
	strCommandline.append(L"-creator " ).append(creator).append(L" ");

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

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

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

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

        FXT_DestroyLibrary ();
```

5.4.2 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 Foxit PDF SDK Library
}
else if (ret == FXT_ERROR_ERROR) {
        // Failed to add a watermark to PDF file.
}
else {
        // Other error
}
```

A non-zero value indicates that an error was encountered. You can find the list for all error codes 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++
wchar_t* MyCallback(void* userData, int mode, wchar_t* msg, bool* isStop) {
        if (mode == CALLBACK_PDFTOOL_RUN_ERROR) {
                wcout << L"Error: " << msg << endl;
        }
        else if (mode == CALLBACK_PDFTOOL_PARAM_ERROR) {
                wcout << msg;
        }
        else if (mode == CALLBACK_PDFTOOL_MSG) {
                wcout << msg;
        }
```

```
        else if (mode == CALLBACK_PDFTOOL_PASSWORD) {
                static wstring password;
                wcin >> password;
                return (wchar_t*)password.c_str();
        }
        return 0;
}
```

5.5 PDFHeaderFooter

5.5.1 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(L"-i d:\\input -o d:\\output\\output.pdf –mode 1 –overlay –conf
d:\\conf_hf.xml", myCallBack, NULL);

        FXT_DestroyLibrary();
}
```

This application adds headers/footers into PDF files. All the PDF files in the "d:\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 the 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::wstring strCommandline = GetCommandLineW();
        std::wstring::size_type pos = strCommandline.find(L".exe");
        strCommandline = strCommandline.substr (pos+6);

        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::wstring input_folder = L"d:\\samples\\input";//or PDF file: input.pdf...
	std::wstring output_folder = L"d:\\samples\\output";//or   PDF file: output.pdf...
	std::wstring password = L"secret";
	std::wstring log_file = L"d:\\samples\\output\\headerfooter.log";
	std::wstring conf_file= L"d:\\samples\\conf\\conf_hf.xml";// the configuration tool

	// log level
	int log_level = 4;

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

	// set title of PDF files.
	std::wstring title = L"Foxit PDF Toolkit User Manual";

	// set subject of PDF files.
	std::wstring subject = L"Foxit PDF Toolkit";

	// set keywords of PDF files.
	std::wstring keywords = L"PDF Toolkit";

	// set author of PDF files.
	std::wstring author = L"Jessie";

	// set file creation application information of PDF files.
	std::wstring creator = L"Foxit Reader";

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

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

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

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

	if(!title.empty())
		strCommandline.append(L"-title " ).append(title).append(L" ");

	if(!subject.empty())
		strCommandline.append(L"-subject " ).append(subject).append(L" ");

	if(!keywords.empty())
		strCommandline.append(L"-keywords " ).append(keywords).append(L" ");

	if(!author.empty())
		strCommandline.append(L"-author " ).append(author).append(L" ");

	if(!creator.empty())
		strCommandline.append(L"-creator " ).append(creator).append(L" ");

	if(!log_file.empty())
```

```
                strCommandline.append(L"-log ").append(log_file).append(L" ");

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

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

        if (mode >= 0)
        {
                wmemset(temp, 0 , MAX_LEGHT);
                _itow(mode, temp ,DATA_RADIX);
                strCommandline.append(L"-mode ").append(temp).append(L" ");
        }

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

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

        FXT_DestroyLibrary();
}
```

5.5.2 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 Foxit PDF SDK Library
}
else if (ret == FXT_ERROR_ERROR) {
        // Failed to add header and footer to PDF file.
}
else {
        // Other error
}
```

A non-zero value indicates that an error was encountered. You can find the list for all error codes 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++
wchar_t* MyCallback(void* userData, int mode, wchar_t* msg, bool* isStop) {
	if (mode == CALLBACK_PDFTOOL_RUN_ERROR) {
		wcout << L"Error: " << msg << endl;
	}
	else if (mode == CALLBACK_PDFTOOL_PARAM_ERROR) {
		wcout << msg;
	}
	else if (mode == CALLBACK_PDFTOOL_MSG) {
		wcout << msg;
	}
	else if (mode == CALLBACK_PDFTOOL_PASSWORD) {
		static wstring password;
		wcin >> password;
		return (wchar_t*)password.c_str();
	}
	return 0;
}
```

5.6 PDFOptimizer

5.6.1 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(L"-i d:\\input -o d:\\output", myCallBack, NULL);

	FXT_DestroyLibrary();
}
```

This application optimizes PDF files. All the PDF files in the "d:\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 the 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::wstring strCommandline = GetCommandLineW();
	std::wstring::size_type pos = strCommandline.find(L".exe");
	strCommandline = strCommandline.substr (pos+6);

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

	// set title of PDF files.
	std::wstring title = L"Foxit PDF Toolkit User Manual";

	// set subject of PDF files.
	std::wstring subject = L"Foxit PDF Toolkit";

	// set keywords of PDF files.
	std::wstring keywords = L"PDF Toolkit";

	// set author of PDF files.
	std::wstring author = L"Jessie";

	// set file creation application information of PDF files.
	std::wstring creator = L"Foxit Reader";

	// 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::wstring strCommandline = L"";

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

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

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

```
if(!title.empty())
        strCommandline.append(L"-title " ).append(title).append(L" ");

if(!subject.empty())
        strCommandline.append(L"-subject " ).append(subject).append(L" ");

if(!keywords.empty())
        strCommandline.append(L"-keywords " ).append(keywords).append(L" ");

if(!author.empty())
        strCommandline.append(L"-author " ).append(author).append(L" ");

if(!creator.empty())
        strCommandline.append(L"-creator " ).append(creator).append(L" ");

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

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

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

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

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

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

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

//set monochrome images compress
std::wstring cmAlgorithm = L"jbig2";  //or "ccitt","runlength"
```

```
        std::wstring cmLevel = L"lossless";   //or "lossy"
        strCommandline.append(L"-cm ").append(cmAlgorithm).append(L" ");
        strCommandline.append(cmLevel).append(L" ");

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

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

        strCommandline.append(L"-rd ");

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

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

5.6.2 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 Foxit PDF SDK Library
}
else if (ret == FXT_ERROR_ERROR) {
        // Failed to optimize PDF file
}
else {
        // Other error
}
```

A non-zero value indicates that an error was encountered. You can find the list for all error codes 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++
wchar_t* MyCallback(void* userData, int mode, wchar_t* msg, bool* isStop) {
	if (mode == CALLBACK_PDFTOOL_RUN_ERROR) {
		wcout << L"Error: " << msg << endl;
	}
	else if (mode == CALLBACK_PDFTOOL_PARAM_ERROR) {
		wcout << msg;
	}
	else if (mode == CALLBACK_PDFTOOL_MSG) {
		wcout << msg;
	}
	else if (mode == CALLBACK_PDFTOOL_PASSWORD) {
		static wstring password;
		wcin >> password;
		return (wchar_t*)password.c_str();
	}
	return 0;
}
```

5.7 PDFRedactor

5.7.1 Working with PDFRedactor API

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

```
// Using C/C++
void main(int argc, char* argv[])
{
	int ret = FXT_InitLibrary(L"license_key", 0);
	FXT_RedactorRun(L"-i d:\\input -o d:\\output -mode 1 -keyword "foxit"", myCallBack,
NULL);

	FXT_DestroyLibrary();
}
```

This application redacts (removes) the sensitive content "Foxit" from the PDF files in the "d:\input" folder except for the secured files, and the redacted PDF files will be saved to the "d:\output" folder.

The command string (`"-i d:\\input -o d:\\output -mode 1 -keyword "foxit""`) 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 PDFRedactor 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::wstring strCommandline = GetCommandLineW();
	std::wstring::size_type pos = strCommandline.find(L".exe");
	strCommandline = strCommandline.substr (pos+6);

	FXT_RedactorRun(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::wstring input_folder = L"d:\\samples\\input";//or a PDF file: input.pdf
	std::wstring output_folder = L"d:\\samples\\output";//or a PDF file: output.pdf...
	std::wstring password = L"secret"; //open password
	std::wstring log_file = L"d:\\samples\\output\\pdfredactor.log";

	// the mode of keyword to be searched.
	int mode = 1;

	// the keyword needed to be searched.
	std::wstring keyword = L"Foxit";

	// the characters of the keyword you want to redact.
	int character = 3;

	// the page range to apply redaction.
	std::wstring range = L"1,5,9";

	// fill color for redaction marks.
	int markcolor = 255 255 0;

	// specify the overlay text for redaction marks.
	std::wstring tx = L"secret";

	// the alignment of the overlay text.
	int ta = 1;

	// the font style of the overlay text.
	std::wstring font = L"Calibri";

	// the font size of the overlay text.
	int fs = 11;

	// font color of the overlay text.
	int fontcolor = 200 255 0;

	// Search the specified keyword in both PDF forms and the main body of text.
	bool form = true;
```

```
        // set title of PDF files.
        std::wstring title = L"Foxit PDF Toolkit User Manual";

        // set subject of PDF files.
        std::wstring subject = L"Foxit PDF Toolkit";

        // set keywords of PDF files.
        std::wstring keywords = L"PDF Toolkit";

        // set author of PDF files.
        std::wstring author = L"Jessie";

        // set file creation application information of PDF files.
        std::wstring creator = L"Foxit Reader";

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

        // log level.
        int log_level = 4;

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

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

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

        if(!keyword.empty())
               strCommandline.append(L"-keyword ").append(keyword).append(L" ");

        if(!range.empty())
               strCommandline.append(L"-range ").append(range).append(L" ");

        if(!tx.empty())
               strCommandline.append(L"-tx ").append(tx).append(L" ");

        if(!font.empty())
               strCommandline.append(L"-font " ).append(font).append(L" ");

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

        if(!title.empty())
               strCommandline.append(L"-title " ).append(title).append(L" ");

        if(!subject.empty())
               strCommandline.append(L"-subject " ).append(subject).append(L" ");

        if(!keywords.empty())
               strCommandline.append(L"-keywords " ).append(keywords).append(L" ");

        if(!author.empty())
               strCommandline.append(L"-author " ).append(author).append(L" ");

        if(!creator.empty())
               strCommandline.append(L"-creator " ).append(creator).append(L" ");
```

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

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

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

    if (depth >= 0)
    {
        wmemset(temp, 0 , MAX_LEGHT);
        _itow(depth, temp ,DATA_RADIX);
        strCommandline.append(L"-depth ").append(temp).append(L" ");
    }

    if (mode >= 0)
    {
        wmemset(temp, 0, MAX_LEGHT);
        _itow(mode, temp ,DATA_RADIX);
        strCommandline.append(L"-mode ").append(temp).append(L" ");
    }

    if (character >= 0)
    {
        wmemset(temp, 0, MAX_LEGHT);
        _itow(character, temp ,DATA_RADIX);
        strCommandline.append(L"-character ").append(temp).append(L" ");
    }

    if (R >= 0 && G >= 0 && B >= 0)
    {
        wmemset(temp, 0, MAX_LEGHT);
        _itow(markcolor, temp ,DATA_RADIX);
        strCommandline.append(L"-markcolor ").append(temp).append(L" ");
    }

    if (ta>= 0)
    {
        wmemset(temp, 0, MAX_LEGHT);
        _itow(ta, temp ,DATA_RADIX);
        strCommandline.append(L"-ta ").append(temp).append(L" ");
    }

    if (fs>= 0)
    {
        wmemset(temp, 0, MAX_LEGHT);
        _itow(fs, temp ,DATA_RADIX);
        strCommandline.append(L"-fs ").append(temp).append(L" ");
    }

    if (R >= 0 && G >= 0 && B >= 0)
    {
        wmemset(temp, 0, MAX_LEGHT);
        _itow(fontcolor, temp ,DATA_RADIX);
```

```
                strCommandline.append(L"-fontcolor ").append(temp).append(L" ");
        }

        if(form)        strCommandline.append(L"-form").append(L" ");

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

        FXT_DestroyLibrary() ;
}
```

5.7.2 Reporting Progress Messages and Errors

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

For example,

```
int ret = FXT_RedactorRun(...);
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 Foxit PDF SDK Library
}
else if (ret == FXT_ERROR_ERROR) {
        // Failed to redact PDF file.
}
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_RedactorRun(). The last parameter in FXT_RedactorRun() 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++
wchar_t* MyCallback(void* userData, int mode, wchar_t* msg, bool* isStop) {
        if (mode == CALLBACK_PDFTOOL_RUN_ERROR) {
                wcout << L"Error: " << msg << endl;
        }
        else if (mode == CALLBACK_PDFTOOL_PARAM_ERROR) {
```

```
                wcout << msg;
        }
        else if (mode == CALLBACK_PDFTOOL_MSG) {
                wcout << msg;
        }
        else if (mode == CALLBACK_PDFTOOL_PASSWORD) {
                static wstring password;
                wcin >> password;
                return (wchar_t*)password.c_str();
        }
        return 0;
}
```

5.8 PDFMetadata

5.8.1 Working with PDFMetadata API

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

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

        FXT_DestroyLibrary();
}
```

This application adds document title information to the PDF files in the "d:\input" folder except for the secured files, and the output PDF files will be saved to the "d:\output" folder.

The command string (`"-i d:\\input -o d:\\output -title "Foxit PDF Toolkit User Manual""`) 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 PDFMetadata 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::wstring strCommandline = GetCommandLineW();
        std::wstring::size_type pos = strCommandline.find(L".exe");
        strCommandline = strCommandline.substr (pos+6);

        FXT_MetadataRun(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 = L"d:\\samples\\input";//or a PDF file: input.pdf
	std::string output_folder = L"d:\\samples\\output";//or a PDF file: output.pdf...
	std::string password = L"secret"; //password
	std::string log_file = L"d:\\samples\\output\\pdfmetadata.log";

	// set title of PDF files.
	std::string title = L"Foxit PDF Toolkit User Manual";

	// set subject of PDF files.
	std::string subject = L"Foxit PDF Toolkit";

	// set keywords of PDF files.
	std::string keywords = L"PDF Toolkit";

	// set author of PDF files.
	std::string author = L"Jessie";

	// set file creation application information of PDF files.
	std::string creator = L"Foxit Reader";

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

	// log level.
	int log_level = 4;

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

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

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

	if(!title.empty())
		strCommandline.append(L"-title " ).append(title).append(L" ");

	if(!subject.empty())
		strCommandline.append(L"-subject " ).append(subject).append(L" ");

	if(!keywords.empty())
		strCommandline.append(L"-keywords " ).append(keywords).append(L" ");

	if(!author.empty())
		strCommandline.append(L"-author " ).append(author).append(L" ");

	if(!creator.empty())
		strCommandline.append(L"-creator " ).append(creator).append(L" ");
```

```
        if(!password.empty())
                strCommandline.append(L"-op ").append(password).append(L" ");

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

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

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

        if (depth >= 0)
        {
                wmemset(temp, 0 , MAX_LEGHT);
                _itow(depth, temp ,DATA_RADIX);
                strCommandline.append(L"-depth ").append(temp).append(L" ");
        }

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

        FXT_DestroyLibrary() ;
}
```

5.8.2 Reporting Progress Messages and Errors

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

For example,

```
int ret = FXT_MetadataRun(...);
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 Foxit PDF SDK Library
}
else if (ret == FXT_ERROR_ERROR) {
        // Failed to set metadata information of PDF file
}
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_MetadataRun(). The last parameter in FXT_MetadataRun() 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++
wchar_t* MyCallback(void* userData, int mode, wchar_t* msg, bool* isStop) {
	if (mode == CALLBACK_PDFTOOL_RUN_ERROR) {
		wcout << L"Error: " << msg << endl;
	}
	else if (mode == CALLBACK_PDFTOOL_PARAM_ERROR) {
		wcout << msg;
	}
	else if (mode == CALLBACK_PDFTOOL_MSG) {
		wcout << msg;
	}
	else if (mode == CALLBACK_PDFTOOL_PASSWORD) {
		static wstring password;
		wcin >> password;
		return (wchar_t*)password.c_str();
	}
	return 0;
}
```

5.9 PDF2Text

5.9.1 Working with PDF2Text API

The following is the simplest application that can be built using PDF2Text API:

```
// Using C/C++
void main(int argc, char* argv[])
{
	int ret = FXT_InitLibrary(L"license_key", 0);
	FXT_PDF2TextRun(L"-i d:\\input -o d:\\output", myCallBack, NULL);

	FXT_DestroyLibrary();
}
```

This application converts all the PDF files in the "d:\input" folder into text files under the "d:\output" folder 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 the command line. Building a command line application using PDF2Text 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::wstring strCommandline = GetCommandLineW();
	std::wstring::size_type pos = strCommandline.find(L".exe");
	strCommandline = strCommandline.substr (pos+6);

	FXT_PDF2TextRun(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::wstring input_folder = L"d:\\samples\\input"; //or a PDF file: input.pdf
	std::wstring output_folder = L"d:\\samples\\output"; //or a text file: output.txt...
	std::wstring password = L"secret"; //password
	std::wstring log_file = L"d:\\samples\\output\\PDF2Text.log";

	// the page range to convert.
	std::wstring range = L"all";

	// get the total number of characters on each page.
	bool charcount = false;

	// print the number of characters to the screen.
	bool printcount = false;

	// ignore the PDF page layout.
	bool notype = true;

	// disable displaying the page number in the output text files.
	bool nopagenumber = true;

	// convert each PDF page into individual text files.
	bool singlepage = true;

	// set Unicode text encoding to UTF16.
	std::wstring encoding = L"UTF16";

	// add a page break to represent the end of each PDF page in the output text files.
	bool breakpage = true;

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

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

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

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

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

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

        if(!range.empty())
                strCommandline.append(L"-range " ).append(range).append(L" ");

        if(!encoding.empty())
                strCommandline.append(L"-encoding " ).append(encoding).append(L" ");

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

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

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

        if (depth >= 0)
        {
                wmemset(temp, 0 , MAX_LEGHT);
                _itow(depth, temp ,DATA_RADIX);
                strCommandline.append(L"-depth ").append(temp).append(L" ");
        }

        if(charcount)   strCommandline.append(L"-charcount").append(L" ");

        if(printcount)  strCommandline.append(L"-printcount").append(L" ");

        if(notype)      strCommandline.append(L"-notype").append(L" ");

        if(nopagenumber)      strCommandline.append(L"-nopagenumber").append(L" ");

        if(singlepage)  strCommandline.append(L"-singlepage").append(L" ");

        if(breakpage) strCommandline.append(L"-breakpage").append(L" ");

        FXT_PDF2TextRun(strCommandline.c_str(), myCallBack, NULL);

        FXT_DestroyLibrary() ;
}
```

5.9.2 Reporting Progress Messages and Errors

To find out if PDF2Text processing was successful, the application can query the status code returned by FXT_PDF2TextRun().

For example,

```
int ret = FXT_PDF2TextRun(...);
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 Foxit PDF SDK Library
}
else if (ret == FXT_ERROR_ERROR) {
	// Failed to convert PDF file to text file
}
else {
	// Other error
}
```

A non-zero value indicates that an error was encountered. You can find the list for all error codes 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_PDF2TextRun(). The last parameter in FXT_PDF2TextRun() 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++
wchar_t* MyCallback(void* userData, int mode, wchar_t* msg, bool* isStop) {
	if (mode == CALLBACK_PDFTOOL_RUN_ERROR) {
		wcout << L"Error: " << msg << endl;
	}
	else if (mode == CALLBACK_PDFTOOL_PARAM_ERROR) {
		wcout << msg;
	}
	else if (mode == CALLBACK_PDFTOOL_MSG) {
		wcout << msg;
	}
	else if (mode == CALLBACK_PDFTOOL_PASSWORD) {
		static wstring password;
		wcin >> password;
		return (wchar_t*)password.c_str();
	}
```

```
        return 0;
}
```

5.10 Text2PDF

5.10.1 Working with Text2PDF API

The following is the simplest application that can be built using Text2PDF API:

```
// Using C/C++
void main(int argc, char* argv[])
{
        int ret = FXT_InitLibrary(L"license_key", 0);
        FXT_Text2PDFRun(L"-i d:\\input -o d:\\output", myCallBack, NULL);

        FXT_DestroyLibrary();
}
```

This application converts all the text files in the "d:\input" folder into PDF files under the "d:\output" folder.

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 the command line. Building a command line application using Text2PDF 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::wstring strCommandline = GetCommandLineW();
        std::wstring::size_type pos = strCommandline.find(L".exe");
        strCommandline = strCommandline.substr (pos+6);

        FXT_Text2PDFRun(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::wstring input_folder = L"d:\\samples\\input";//or a text file: input.txt
        std::wstring output_folder = L"d:\\samples\\output";//or a PDF file: output.pdf...
        std::wstring set_password = L"secret"; //open password for office files
        std::wstring log_file = L"d:\\samples\\output\\convert2pdf.log";
```

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

// page margin of the output PDF file.
int margin_top = 20, margin_bottom = 20, margin_left = 20, margin_right = 20;

// font style of the output PDF file
std::wstring font = L"Calibri";

// font size of the output PDF file
int fs = 11;

// font color of the output PDF file
int fontcolor = 255 255 0;

// the output PDF file will be paginated wherever there is a page break.
bool breakpage = true;

// line spacing for text conversion.
int linespace = 12;

// set title of PDF files.
std::wstring title = L"Foxit PDF Toolkit User Manual";

// set subject of PDF files.
std::wstring subject = L"Foxit PDF Toolkit";

// set keywords of PDF files.
std::wstring keywords = L"PDF Toolkit";

// set author of PDF files.
std::wstring author = L"Jessie";

// set file creation application information of PDF files.
std::wstring creator = L"Foxit Reader";

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

// log level.
int log_level = 4;

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

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

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

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

if(!font.empty())
       strCommandline.append(L"-font " ).append(font).append(L" ");

if(!title.empty())
       strCommandline.append(L"-title " ).append(title).append(L" ");
```

```
if(!subject.empty())
        strCommandline.append(L"-subject " ).append(subject).append(L" ");

if(!keywords.empty())
        strCommandline.append(L"-keywords " ).append(keywords).append(L" ");

if(!author.empty())
        strCommandline.append(L"-author " ).append(author).append(L" ");

if(!creator.empty())
        strCommandline.append(L"-creator " ).append(creator).append(L" ");

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

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

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

if (depth >= 0)
{
        wmemset(temp, 0 , MAX_LEGHT);
        _itow(depth, temp ,DATA_RADIX);
        strCommandline.append(L"-depth ").append(temp).append(L" ");
}

if (width > 0 && height > 0)
{
        wmemset(temp, 0 , MAX_LEGHT);
        _itow(width, temp ,DATA_RADIX);
        strCommandline.append(L"-width ").append(temp).append(L" ");

        wmemset(temp, 0 , MAX_LEGHT);
        _itow(height, temp ,DATA_RADIX);
        strCommandline.append(L"-height ").append(temp).append(L" ");
}

if (margin_top >= 0 || margin_right >= 0 || margin_bottom >= 0 || margin_left >= 0)
{
        bool flag = false;
        if (margin_left >= 0)
        {
                wmemset(temp, 0 , MAX_LEGHT);
                _itow(margin_left, temp ,DATA_RADIX);
                strCommandline.append(L"-margin ").append(temp).append(L" ");
                if(margin_top >= 0)
                {
                        wmemset(temp, 0 , MAX_LEGHT);
                        _itow(margin_top, temp ,DATA_RADIX);
                        strCommandline.append(temp).append(L" ");
                        if(margin_right >= 0)
                        {
                                wmemset(temp, 0 , MAX_LEGHT);
```

```
                                        _itow(margin_right, temp ,DATA_RADIX);
                                        strCommandline.append(temp).append(L" ");
                                        if(margin_bottom >= 0)
                                        {
                                                wmemset(temp, 0 , MAX_LEGHT);
                                                _itow(margin_bottom, temp ,DATA_RADIX);
                                                strCommandline.append(temp).append(L" ");
                                        }
                                }
                        }
                }
        }

        if (fs>= 0)
        {
                wmemset(temp, 0, MAX_LEGHT);
                _itow(fs, temp ,DATA_RADIX);
                strCommandline.append(L"-fs ").append(temp).append(L" ");
        }

        if (R >= 0 && G >= 0 && B >= 0)
        {
                wmemset(temp, 0, MAX_LEGHT);
                _itow(fontcolor, temp ,DATA_RADIX);
                strCommandline.append(L"-fontcolor ").append(temp).append(L" ");
        }

        if(breakpage) strCommandline.append(L"-breakpage").append(L" ");

        if (linespace>= 0)
        {
                wmemset(temp, 0, MAX_LEGHT);
                _itow(linespace, temp ,DATA_RADIX);
                strCommandline.append(L"-linespace ").append(temp).append(L" ");
        }

        FXT_Text2PDFRun(strCommandline.c_str(), myCallBack, NULL);

        FXT_DestroyLibrary() ;
}
```

5.10.2 Reporting Progress Messages and Errors

To find out if Text2PDF processing was successful, the application can query the status code returned by FXT_Text2PDFRun().

For example,

```
int ret = FXT_Text2PDFRun(...);
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 Foxit PDF SDK Library
}
else if (ret == FXT_ERROR_ERROR) {
	// Failed to convert text file to PDF file
}
else {
	// Other error
}
```

A non-zero value indicates that an error was encountered. You can find the list for all error codes 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_Text2PDFRun(). The last parameter in FXT_Text2PDFRun() 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++
wchar_t* MyCallback(void* userData, int mode, wchar_t* msg, bool* isStop) {
	if (mode == CALLBACK_PDFTOOL_RUN_ERROR) {
		wcout << L"Error: " << msg << endl;
	}
	else if (mode == CALLBACK_PDFTOOL_PARAM_ERROR) {
		wcout << msg;
	}
	else if (mode == CALLBACK_PDFTOOL_MSG) {
		wcout << msg;
	}
	else if (mode == CALLBACK_PDFTOOL_PASSWORD) {
		static wstring password;
		wcin >> password;
		return (wchar_t*)password.c_str();
	}
	return 0;
}
```

5.11 Html2PDF

5.11.1 Working with Html2PDF API

The following is the simplest application that can be built using Html2PDF API:

```
// Using C/C++
void main(int argc, char* argv[])
{
	int ret = FXT_InitLibrary(L"license_key", 0);
	FXT_Html2PDFRun(L"-i d:\\input -o d:\\output", myCallBack, NULL);
```

```
        FXT_DestroyLibrary();
}
```

This application converts all the HTML files in the "d:\input" folder into PDF files under the "d:\output" folder.

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 Html2PDF 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::wstring strCommandline = GetCommandLineW();
        std::wstring::size_type pos = strCommandline.find(L".exe");
        strCommandline = strCommandline.substr (pos+6);

        FXT_Html2PDFRun(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::wstring input_folder = L"d:\\samples\\input";//or a HTML file or a URL
        std::wstring output_folder = L"d:\\samples\\output";//or a PDF file: output.pdf...
        std::wstring log_file = L"d:\\samples\\output\\html2pdf.log";

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

        // page margin of the output PDF file.
        int margin_top = 20, margin_bottom = 20, margin_left = 20, margin_right = 20;

        // cache address of HTML page resources.
        std::wstring cache = L"d:\resources";

        // timeout to load a webpage.
        int timeout = 60;

        // set all the page contents to one single PDF page.
        bool singlepage = true;

        // disable retaining hyperlinks in PDF files from HTML files or URL.
        bool nolink = true;

        // page rotation for output PDF file.
        int rotate = 90;
```

```
// improves the conversion quality if the webpages include lazy loading elements or if
// the network/hardware performance is not good enough.
bool checklazyload = true;

// specifies the cookies file.
std::wstring cookies = L"d:\cookies.txt";

// specifies the JavaScript file.
std::wstring js = L"d:\login.js";

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

// log level
int log_level = 4;

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

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

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

if(!cache.empty())
	strCommandline.append(L"-cache ").append(cache).append(L" ");

if(!cookies.empty())
	strCommandline.append(L"-cookies ").append(cookies).append(L" ");

if(!js.empty())
	strCommandline.append(L"-js ").append(js).append(L" ");

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

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

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

if (depth >= 0)
{
	wmemset(temp, 0 , MAX_LEGHT);
	_itow(depth, temp ,DATA_RADIX);
	strCommandline.append(L"-depth ").append(temp).append(L" ");
}

if (width > 0 && height > 0)
{
	wmemset(temp, 0 , MAX_LEGHT);
	_itow(width, temp ,DATA_RADIX);
```

```
            strCommandline.append(L"-width ").append(temp).append(L" ");

            wmemset(temp, 0 , MAX_LEGHT);
            _itow(height, temp ,DATA_RADIX);
            strCommandline.append(L"-height ").append(temp).append(L" ");
        }

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

        if (timeout>= 0)
        {
            wmemset(temp, 0, MAX_LEGHT);
            _itow(timeout, temp ,DATA_RADIX);
            strCommandline.append(L"-timeout ").append(temp).append(L" ");
        }

        if(singlepage) strCommandline.append(L"-singlepage").append(L" ");

        if(nolink)     strCommandline.append(L"-nolink").append(L" ");

        if (rotate>= 0)
        {
            wmemset(temp, 0, MAX_LEGHT);
            _itow(rotate, temp ,DATA_RADIX);
            strCommandline.append(L"-rotate ").append(temp).append(L" ");
        }

        if(checklazyload) strCommandline.append(L"-checklazyload").append(L" ");

        FXT_Html2PDFRun(strCommandline.c_str(), myCallBack, NULL);

        FXT_DestroyLibrary() ;
}
```

5.11.2 Reporting Progress Messages and Errors

To find out if Html2PDF processing was successful, the application can query the status code returned by FXT_Html2PDFRun().

For example,

```
int ret = FXT_Html2PDFRun(...);
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_ERROR) {
	// Failed to convert HTML file into PDF file
}
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_Html2PDFRun(). The last parameter in FXT_Html2PDFRun() 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++
wchar_t* MyCallback(void* userData, int mode, wchar_t* msg, bool* isStop) {
	if (mode == CALLBACK_PDFTOOL_RUN_ERROR) {
		wcout << L"Error: " << msg << endl;
	}
	else if (mode == CALLBACK_PDFTOOL_PARAM_ERROR) {
		wcout << msg;
	}
	else if (mode == CALLBACK_PDFTOOL_MSG) {
		wcout << msg;
	}
	else if (mode == CALLBACK_PDFTOOL_PASSWORD) {
		static wstring password;
		wcin >> password;
		return (wchar_t*)password.c_str();
	}
```

```
        return 0;
}
```

5.12 PDF Page Organizer

5.12.1 Working with PDF Page Organizer API

The following is the simplest application that can be built using PDF Page Organizer API:

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

        FXT_DestroyLibrary();
}
```

This application merges all the PDF files in the "d:\input" folder into one PDF files under the "d:\output" folder except for the secured files.

The command string (`"-i d:\\input -o d:\\output\\merge"`) of the above application is set by users directly in advance. Users also can get the command string through the command line. Building a command line application using PDF Page Organizer 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::wstring strCommandline = GetCommandLineW();
        std::wstring::size_type pos = strCommandline.find(L".exe");
        strCommandline = strCommandline.substr (pos+6);

        FXT_PDFPpoRun(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);

        // specify the input with a page list
        std::wstring input_folder = L"d:\\samples\\input,even,50-";

        // specify the output folder
```

```
std::wstring output_folder = L"d:\\samples\\output";

// specify the password for the input secured PDF(s)
std::wstring password = L"secret";

// write log information
std::wstring log_file = L"d:\\samples\\output\\PDFppo.log";

// the mode of operation: split mode
int mode = 2;

// split PDF files into a set of PDF files containing 3 pages each
int maxpage = 5;

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

// log level
int log_level = 4;

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

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

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

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

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

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

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

if (depth >= 0)
{
       wmemset(temp, 0, MAX_LEGHT);
       _itow(depth, temp, DATA_RADIX);
       strCommandline.append(L"-depth ").append(temp).append(L" ");
}

if (mode >= 0)
{
       wmemset(temp, 0, MAX_LEGHT);
       _itow(mode, temp, DATA_RADIX);
       strCommandline.append(L"-mode ").append(temp).append(L" ");
}
```

```
        if (maxpage >= 0)
        {
                wmemset(temp, 0, MAX_LEGHT);
                _itow(maxpage, temp, DATA_RADIX);
                strCommandline.append(L"-maxpage ").append(temp).append(L" ");
        }

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

        FXT_DestroyLibrary() ;
}
```

5.12.2 Reporting Progress Messages and Errors

To find out if PDF Page Organizer processing was successful, the application can query the status code returned by FXT_PDFPpoRun().

For example,

```
int ret = FXT_PDFPpoRun(...);
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 Foxit PDF SDK Library
}
else if (ret == FXT_ERROR_ERROR) {
        // Failed to merge/split PDF files or delete pages of PDF files
}
else {
        // Other error
}
```

A non-zero value indicates that an error was encountered. You can find the list for all error codes 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_PDFPpoRun(). The last parameter in FXT_PDFPpoRun() 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++
wchar_t* MyCallback(void* userData, int mode, wchar_t* msg, bool* isStop) {
        if (mode == CALLBACK_PDFTOOL_RUN_ERROR) {
                wcout << L"Error: " << msg << endl;
```

```
        }
        else if (mode == CALLBACK_PDFTOOL_PARAM_ERROR) {
                wcout << msg;
        }
        else if (mode == CALLBACK_PDFTOOL_MSG) {
                wcout << msg;
        }
        else if (mode == CALLBACK_PDFTOOL_PASSWORD) {
                static wstring password;
                wcin >> password;
                return (wchar_t*)password.c_str();
        }
        return 0;
}
```

5.13 PDFSecure

5.13.1 Working with PDFSecure API

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

```
// Using C/C++
void main(int argc, char* argv[])
{
        int ret = FXT_InitLibrary(L"license_key", 0);
        FXT_PDFSecureRun(L"-i d:\\input -o d:\\output -mode 1 -sup 111", myCallBack, NULL);

        FXT_DestroyLibrary();
}
```

This application sets the user password to "111" for all the unsecured PDF files in the "d:\input" folder, and the output encrypted PDF files will be saved to "d:\output" folder.

The command string (`"-i d:\\input -o d:\\output -mode 1 -sup 111"`) of the above application is set by users directly in advance. Users also can get the command string through the command line. Building a command line application using PDFSecure 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::wstring strCommandline = GetCommandLineW();
        std::wstring::size_type pos = strCommandline.find(L".exe");
        strCommandline = strCommandline.substr (pos+6);

        FXT_PDFSecureRun(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::wstring input_folder = L"d:\\samples\\input"; //or a PDF file: input.pdf
	std::wstring output_folder = L"d:\\samples\\output"; //or a PDF file: output.PDF...
	std::wstring password = L"secret"; //password
	std::wstring log_file = L"d:\\samples\\output\\PDFsecure.log";

	// the mode of operation
	int mode = 1;

	// set user password for the output PDF file
	std::wstring sup = L"welcome";

	// set owner password for the output PDF file
	std::wstring sop = L"foxit";

	// set the permission flags for the output PDF file
	std::wstring permission = L"234";

	// set the security handle to encrypt the output PDF file
	int ea = 2;

	// encrypt the metadata of the output PDF file
	bool em = true;

	// set title of PDF files.
	std::wstring title = L"Foxit PDF Toolkit User Manual";

	// set subject of PDF files.
	std::wstring subject = L"Foxit PDF Toolkit";

	// set keywords of PDF files.
	std::wstring keywords = L"PDF Toolkit";

	// set author of PDF files.
	std::wstring author = L"Jessie";

	// set file creation application information of PDF files.
	std::wstring creator = L"Foxit Reader";

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

	//set thread number
	int thread_number = 4;

	// log level
	int log_level = 4;

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

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

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

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

        if(!sup.empty())
                strCommandline.append(L"-sup " ).append(sup).append(L" ");

        if(!sop.empty())
                strCommandline.append(L"-sop " ).append(sop).append(L" ");

        if(!permission.empty())
                strCommandline.append(L"-p " ).append(permission).append(L" ");

        if(!title.empty())
                strCommandline.append(L"-title " ).append(title).append(L" ");

        if(!subject.empty())
                strCommandline.append(L"-subject " ).append(subject).append(L" ");

        if(!keywords.empty())
                strCommandline.append(L"-keywords " ).append(keywords).append(L" ");

        if(!author.empty())
                strCommandline.append(L"-author " ).append(author).append(L" ");

        if(!creator.empty())
                strCommandline.append(L"-creator " ).append(creator).append(L" ");

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

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

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

        if (depth >= 0)
        {
                wmemset(temp, 0 , MAX_LEGHT);
                _itow(depth, temp ,DATA_RADIX);
                strCommandline.append(L"-depth ").append(temp).append(L" ");
        }

        if (thread_number>= 0)
        {
                wmemset(temp, 0, MAX_LEGHT);
                _itow(thread_number, temp ,DATA_RADIX);
                strCommandline.append(L"-t ").append(temp).append(L" ");
        }
```

```
        if (mode >= 0)
        {
                wmemset(temp, 0, MAX_LEGHT);
                _itow(mode, temp ,DATA_RADIX);
                strCommandline.append(L"-mode ").append(temp).append(L" ");
        }

        if (ea >= 0)
        {
                wmemset(temp, 0, MAX_LEGHT);
                _itow(ea, temp ,DATA_RADIX);
                strCommandline.append(L"-ea ").append(temp).append(L" ");
        }

        if(em)  strCommandline.append(L"-em").append(L" ");

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

        FXT_DestroyLibrary();
}
```

5.13.2 Reporting Progress Messages and Errors

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

For example,

```
int ret = FXT_PDFSecureRun(...);
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 Foxit PDF SDK Library
}
else if (ret == FXT_ERROR_ERROR) {
        // Failed to encrypt the PDF file
}
else {
        // Other error
}
```

A non-zero value indicates that an error was encountered. You can find the list for all error codes 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_PDFSecureRun(). The last parameter in FXT_PDFSecureRun() 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++
wchar_t* MyCallback(void* userData, int mode, wchar_t* msg, bool* isStop) {
	if (mode == CALLBACK_PDFTOOL_RUN_ERROR) {
		wcout << L"Error: " << msg << endl;
	}
	else if (mode == CALLBACK_PDFTOOL_PARAM_ERROR) {
		wcout << msg;
	}
	else if (mode == CALLBACK_PDFTOOL_MSG) {
		wcout << msg;
	}
	else if (mode == CALLBACK_PDFTOOL_PASSWORD) {
		static wstring password;
		wcin >> password;
		return (wchar_t*)password.c_str();
	}
	return 0;
}
```

5.14 PDF2Word

5.14.1 Working with PDF2Word API

The following is the simplest application that can be built using PDF2Word API:

```
// Using C/C++
void main(int argc, char* argv[])
{
	int ret = FXT_InitLibrary(L"license_key", 0);
	FXT_PDF2WordRun(L"-i d:\\input -o d:\\output", myCallBack, NULL);

	FXT_DestroyLibrary();
}
```

This application converts all the PDF files in the "d:\input" folder into Word files under the "d:\output" folder 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 the command line. Building a command line application using PDF2Word 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::wstring strCommandline = GetCommandLineW();
	std::wstring::size_type pos = strCommandline.find(L".exe");
	strCommandline = strCommandline.substr (pos+6);

	FXT_PDF2WordRun(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::wstring input_folder = L"d:\\samples\\input"; //or a PDF file: input.pdf
	std::wstring output_folder = L"d:\\samples\\output"; //or a Word file: output.docx...
	std::wstring password = L"secret"; //password
	std::wstring log_file = L"d:\\samples\\output\\PDF2Word.log";

	// the page range to convert.
	std::wstring range = L"all";

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

	// log level
	int log_level = 4;

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

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

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

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

	if(!range.empty())
		strCommandline.append(L"-range " ).append(range).append(L" ");

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

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

	if (log_level > 0)
```

```
        {
                _itow(log_level,temp,DATA_RADIX);
                strCommandline.append(L"-l ").append(temp).append(L" ");
        }

        if (depth >= 0)
        {
                wmemset(temp, 0 , MAX_LEGHT);
                _itow(depth, temp ,DATA_RADIX);
                strCommandline.append(L"-depth ").append(temp).append(L" ");
        }


        FXT_PDF2WordRun(strCommandline.c_str(), myCallBack, NULL);

        FXT_DestroyLibrary() ;
}
```

5.14.2 Reporting Progress Messages and Errors

To find out if PDF2Word processing was successful, the application can query the status code returned by FXT_PDF2WordRun().

For example,

```
int ret = FXT_PDF2WordRun(...);
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 Foxit PDF SDK Library
}
else if (ret == FXT_ERROR_ERROR) {
        // Failed to convert PDF file to Word file
}
else {
        // Other error
}
```

A non-zero value indicates that an error was encountered. You can find the list for all error codes 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_PDF2WordRun(). The last parameter in FXT_PDF2WordRun() 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++
wchar_t* MyCallback(void* userData, int mode, wchar_t* msg, bool* isStop) {
	if (mode == CALLBACK_PDFTOOL_RUN_ERROR) {
		wcout << L"Error: " << msg << endl;
	}
	else if (mode == CALLBACK_PDFTOOL_PARAM_ERROR) {
		wcout << msg;
	}
	else if (mode == CALLBACK_PDFTOOL_MSG) {
		wcout << msg;
	}
	else if (mode == CALLBACK_PDFTOOL_PASSWORD) {
		static wstring password;
		wcin >> password;
		return (wchar_t*)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 Foxit support team 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
- Your Operating System
- Detailed description of the problem
- Any other related information, such as error screenshot

Note

- *In the unfortunate event that Foxit PDF Toolkit should crash, Foxit PDF Toolkit will generate two files named "CRASHLOG.TXT" and "CRASH.DMP" under the current execution directory. The "CRASHLOG.TXT" file will record the detailed information of the module and the system that are used at the time the crash occurred. To help our engineering team track down the problem and provide a solution, please submit the two files to Foxit support team. Thank you for your cooperation.*
- *For the html2pdf module, if you have installed security software locally and access the webpages embedded with advertising JavaScript, the main program "fhtml2pdf.exe" (or "fhtml2pdf64.exe") and the background program"fxhtml2pdf.exe" may be deleted by the security software. In order to use html2pdf module properly, please add the following lists to the security software's white list, where "[installPath]" is the installation path of Foxit PDF Toolkit.*

  [installPath]\fhtml2pdf.exe
  [installPath]\fhtml2pdf64.exe
  [installPath]\lib\html2pdf\x64\fxhtml2pdf.exe
  [installPath]\lib\html2pdf\x86\fxhtml2pdf.exe

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