PDFTron PDF2Text™
User Manual

Version 4.x

LEGAL STATEMENT AND COPYRIGHT NOTICE

PDFTron PDF2Text™ Command-Line Application User Manual
Part number: PDFTRON-4-PDF2TextCMD
Last Updated: February 1, 2010

TABLE OF CONTENTS

1. Introduction

1.1 An Introduction to PDFTron PDF2Text

PDFTron's PDF2Text is an easy-to-use, multi-platform command-line program for high-quality and efficient text extraction from PDF documents. PDF2Text can be used to convert text from any PDF document as Unicode or as structured XML, while providing a wide range of output styles and configuration options.

PDF2Text is offered as an easy-to-use command-line application and as a software development component that can be used as a building block for other client and server-based applications.

1.1.1 Key Functions

- Extracts text from any PDF document to text or as structured XML.
- Offers different Unicode text encoding (UTF-8 and UTF-16) options.
- Provides positioning, font, and styling information for every Paragraph, Line, Word, or a Glyph on a page.
- Offers options to control the level of detail and the formatting in the output XML.
- Offers advanced options to control ligature expansion, hyphen removal, and to remove duplicate text (e.g. which is sometimes used for drop shadow effects).
- Allows for text extraction from a clip rectangle or to hide text in specific regions on a page.
- Option to remove hidden text or text that is obscured by other page elements (such as images or rectangles).
- Support for all versions of the PDF format (PDF 1.0 to ISO32000).
- Full support for encrypted documents (40 and 128 bit RC4 and 128 bit AES).
- Supports automation and batch operation.

1.1.2 Benefits

Complete Unicode support. PDF2Text can process PDF files from any part of the world (including Asian languages) and represent the extracted text using UTF-8 and UTF-16. To improve Unicode output PDF2Text can recognize vendor-specific Unicode character assignments (in the Private Use Area) and map them to public Unicode area. Similarly Unicode ligatures and PDF specific ligatures can be broken into a sequence of individual Unicode characters. Characters that can't be mapped to Unicode are predictably mapped in the Private Use Area.

Intelligent Text Recognition. Intelligent text recognition and logical structure engine used to recognize words, lines, paragraphs, and the reading order in PDF documents. The engine can remove duplicated text commonly used to drop shadows, or text that is obscured by other page content. The text extractor also works flawlessly with PDF documents that contain rotated text or documents where the information is presented in a random order or is scattered across the page.

Highest Reliability and Robustness. PDF2Text was from ground-up designed to be run in high throughput server-based and multi-threaded applications. A regular and rigorous Q&A process sets high standards for the reliability of all PDFTron products.

Top Performance. Advanced text recognition and content analysis algorithms coupled with low-memory usage and native code efficiency, make PDF2Text the ideal choice for high-traffic servers as well as for interactive applications.

1.1.3   Sample Use Case Scenarios

- Server-based, on-demand conversion of PDF documents to text format files.
- Extract text from a large PDF repository for text indexing or content retrieval purposes (e.g. to implement a PDF search engine).
- Classify or summarize PDF documents based on their content. Find specific words for content editing purposes (such as splitting pages based on keywords, etc).
- Convert PDF pages to text or XML for content repurposing.
- Search PDF pages for specific words or keywords and return their positioning information (e.g. to highlight instances of a given word).

1.1.4   PDF2Text SDK

For developers who are looking for a software development component to integrate into their application, PDFTron also offers PDF2Text SDK, an easy-to-use, yet powerful software component for extracting text from PDF documents. PDF2Text SDK is available as a plain "C DLL" and can be easily accessed from any programming language (including C#, VB.NET, C/C++, Java, VB6, Perl, Python, Ruby, Delphi, etc). PDF2Text is based on PDFNet SDK, PDFTron's own comprehensive PDF library. If you require rasterization or additional PDF functionality than what is provided as part of PDF2Text SDK for embedding in your own applications, please check out PDFNet SDK (http://www.pdftron.com/pdfnet) or contact a PDFTron representative for more information.

1.1.5   Operating Systems Supported

- Windows 7, 2008, Vista, XP, 2003, 2000, NT, 98
- Mac OSX
- Linux

1.1.6   System Requirements

- At least 10 MB of free disk space.
- 2 GB or RAM.

1.2   About This Manual

This manual is intended as a guide to the installation and use of PDF2Text. It is intended for programmers and other users who are familiar with PDF documents, text file layouts, as well as general computer processes.

- Section 1 introduces PDF2Text and describes the manual.
- Section 2 explains how to install and uninstall PDF2Text.
- Section 3 covers basic use of PDF2Text.
- Section 4 is where you will find all the support information you may require, such as how to report a problem with the software.

2.   Installing and Uninstalling PDF2Text

2.1   PDF2Text Installation

PDF2Text Command-line Application is supplied as a download from a distributor or directly from www.pdftron.com. The release is packaged as a .zip file. (pdf2text.zip). To install the software, simply unzip the archive in the desired location and make sure to preserve the directory/folder structure during this process. To register the software, copy the license file provided to you into the "pdf2text" folder.
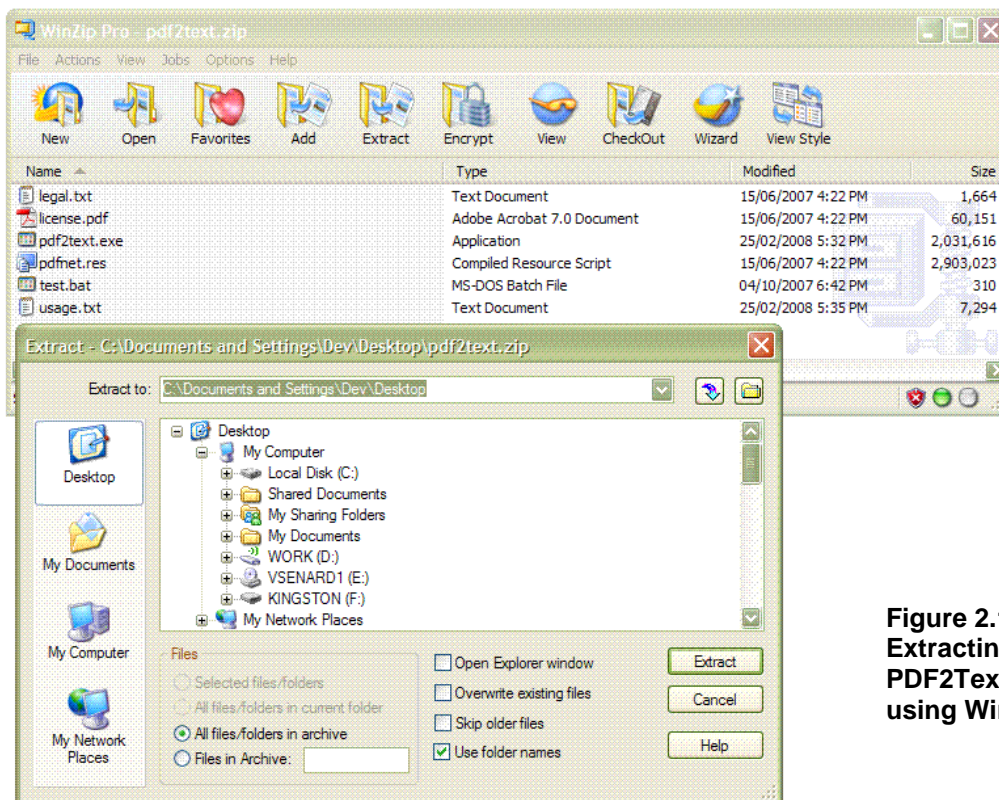
Figure 2.1 - Extracting PDF2Text Archive using WinZip

2.2   Demo Version Installation

If you wish to evaluate the product, you can download the demo version of the product without any serial number or license key.

To do this, go to PDFTron's **Downloads** page at www.pdftron.com/downloads.html. Click on the appropriate product name/version. This will bring you to the link to the page to download the demo. Download the zip file (pdf2text.zip) and extract the archive in the desired location, while making sure to preserve the directory (folder) structure when extracting the archive. Download the zip file *pdf2text.zip*. Extract the archive in the desired location (making sure to preserve the folder structure). This will provide you a working copy of the application along with various examples. The limitation of the evaluation version is that all output pages will have demo stamp.

2.3   Uninstalling PDF2Text

To remove PDF2Text from a computer, simply delete the "pdf2text" folder.

3. Overview

PDFTron PDF2Text is a command-line application designed to convert PDF documents to text or XML. This section covers the basic usage of PDF2Text explaining all of the available options.
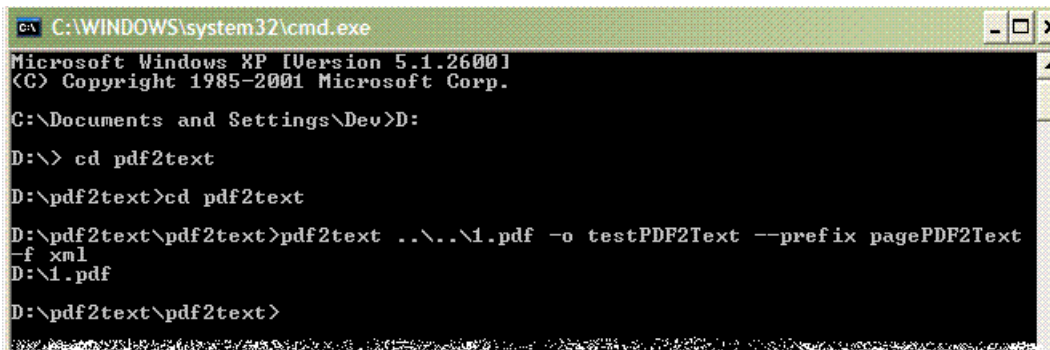

Figure 3.1   PDF2Text Command-line Application.

3.1   Basic Syntax

The basic command-line syntax is:

```
pdf2text [options] file1 file2 folder1 file3 ...
```

3.2   Command-Line Summary

The following is a list of available command-line options for PDF2Text:

Option	Parameter	Description
-o or --output	e.g. -o testfolder	The output folder used to store converted files. By default, the output will be sent to console window.
-a or --pages	Convert page 1,3, and 10: -a 1,3,10 Convert all even pages: -a even Convert pages in the range from 3-11 and page 50: --pages 3-11,50 Convert all odd pages and all pages in the range from 100 to the last page: -a odd,100-	Specifies the list of pages to convert. By default, all pages are converted.
-e or--encoding	e.g. -e UTF8	Output text encoding: ■ **UTF8** ■ **UTF16** The default output encoding is UTF8.
-f or --format	e.g. -f plain	Output text formatting: ■ **plain** ■ **wordlist** ■ **textruns** ■ **xml** The default output format is **plain**.
--noligatures	e.g. --noligatures	Disables expanding of ligatures using a predefined mapping. Default ligatures are: fi, ff, fl, ffi, ffl, ch, cl, ct, ll, ss, fs, st, oe, OE.
--nodehyphen	e.g. --nodehyphen	Disables finding and removing hyphens that split words across two lines. Hyphens are often used at the end of lines as an indicator that a word spans two lines. Hyphen detection enables removal of hyphen character and merging of text runs to form a single word. This option has no effect on Tagged PDF files.
--no_dup_remove	e.g. –no_dup_remove	Disables removing duplicated text that is frequently used to achieve visual effects of drop shadow and fake bold.
--punct_break	e.g. –punct_break	Treat punctuation (e.g. full stop, comma, semicolon, etc.) as word break characters.
--remove_hidden_text	e.g. –remove_hidden_text	Enables removal of text that is obscured by images or rectangles. Since this option has small

		performance penalty on performance of text extraction, by default it is not enabled.
--no_invisible_text	e.g. --no_invisible_text	Enables removing text that uses rendering mode 3 (i.e. invisible text). Invisible text is usually used in 'PDF Searchable Images' (i.e. scanned pages with a corresponding OCR text). As a result, invisible text will be extracted by default.
--output_bbox	e.g. –output_bbox	Include bounding box information for each text element. If the output format is 'XML' the bounding box information will bestored in 'bbox' attribute. If the output format is 'wordlist' the coordinates of the bounding box will precede the word.
--xml_words_as_elements	e.g. –xml_words_as_elements	Output words as XML elements instead of inline text.
--xml_output_styles	e.g. --xml_output_styles	Include font and styling information.
--wordcount	e.g. -wordcount	Get the number of words on each page.
--charcount	e.g. --charcount	Get total number of characters on each page.
--pageinfo	e.g. --pageinfo	Get the width, height, media box, crop box, and page rotation for every page.
--prefix	e.g. –prefix testPage	The prefix for output text files. The output filename will be constructed by appending the prefix string, the page number, and the appropriate file extension (e.g. myprefix1.txt, myprefix2.xml, etc). The prefix option should be used only for processing of individual documents. By default, PDF filename will be used as a prefix.
--digits	e.g. –digits 5 Note the above will get the following format: 00001	The number of digits used in the page counter portion of the output filename. By default, new digits are added as needed; however this parameter could be used to format the page counter field to a uniform width (e.g. myfile0001.txt, myfile0002.txt, etc).
--subfolders	e.g. --subfolders	Process all sub-directory for every directory specified in the argument list. By default, sub-directories are not processed.
-c or --clip	e.g. -c 216, 522, 330, 600	User definable clip box. The default clip region is crop box of the page.
--noprompt	e.g. --noprompt	Disables any user input. By default, the application will ask for a valid password if the password is incorrect.
-p	e.g. –p 111	The password for secured PDF files. Not required if the input document is not secured using the 'open' password.

--extension	e.g. –extension .jpg	The default file extension used to process PDF documents. The default extension is ".pdf".
--verb	e.g. –verb 2	Set the opt.m_verbosity level to 'arg' (0-2). The default is 1.
-v or --version	e.g. --version	Print the version information.
-h or --help	e.g. -h	Print a listing of available options.

3.3   Basic Usage

3.3.1   How do I save exported text file to a given folder?

By default, PDF2Text outputs extracted text in the console window. To save the result in a certain folder instead, use the -o (or --output) parameter. For example:

```
pdf2text -o "..\..\My Output" 1.pdf
```

Note: If the specified path does not exist, PDF2Text will attempt to create the necessary folders.

3.3.2   How can I control the output name for converted files?

By default, PDF2Text creates a separate text file for every page in the document. The output filename is constructed using the name of the input PDF file, page counter, and appropriate file extension. For example, the following command-line generates a sequence of text files in "MyFolder", starting with mydoc_1.txt, mydoc_2.txt, etc.:

```
pdf2text –o MyFolder mydoc.pdf
```

PDF2Text allows output filename customizations using the '--prefix' and '--digits' options. For example, the following command-line generates a sequence of text files in "MyFolder", starting with newname_0001.jpg, newname_0002.jpg, etc.:

```
pdf2text –o MyFolder --prefix newname --digits 4 mydoc.pdf
```

The '--digits' parameter specifies the number of digits used in the page counter portion of the output filename. By default, new digits are added as needed; however this parameter could be used to format the page counter field to a uniform width (e.g. myfile0001.jpg, myfile0010.jpg, instead of myfile_1.jpg, myfile_10.jpg, etc).

To avoid any ambiguities in file naming, the prefix option should be used only for conversion of individual documents.

3.3.3   How do I convert PDF to XML, a list of words or list of text-runs?

By default, PDF2Text automatically converts PDF to a plain .txt file without any extra metadata. The output image format can be modified using the '-f' (or --format) option. For example,

```
pdf2text –f xml in.pdf
```

Will convert PDF to XML and will include number of additional properties such as positioning and styling information for each word.

The '--format' parameter accepts any of the following text formats:

- **plain**
- **wordlist**
- **textruns**
- **xml**

3.3.4   How do I specify the output text encoding format?

By default, PDF2Text is using UTF8 encoding. To modify output encoding use -e (or --encoding) option. For example,

```
pdf2text --encoding UTF16 in.pdf
```

The '--encoding' parameter supports two encoding formats:

- **UTF8**
- **UTF16**

3.3.5   How do I open a password protected PDF?

PDF2Text will, without user intervention, decrypt and convert documents secured with a master/owner password. If the document is secured using a user (i.e. 'file open') password, PDF2Text will, by default, prompt the user to enter the password. If '--noprompt' option is used, the program will not ask for a password, and an error message will be displayed instead.

For unattended conversion, the password can also be specified directly on the command-line using the '-p' (or --password) option. For example:

```
pdf2text -p secret -f xml secured.pdf
```

The above command line will convert PDF to xml format and will use the provided password ('secret') to open the secured PDF document.

Note: PDF2Text supports all standard security options available in PDF, including 40 and 128 bit RC4 encryption, Crypt filters, and 128 AES (Advanced Encryption Standard) encryption.

3.3.6   How do I specify which pages to convert?

By default, PDF2Text will convert all PDF pages to text. You can specify a subset of pages to convert using the '-a' or '--pages' options. For example:

```
pdf2text -a 1,3,10 in.pdf
```

will convert only pages 1, 3, and 10. Please note that PDF2Text assumes that all pages are numbered sequentially starting from page 1.

To specify a range of pages, use dash character between numbers. For example:

```
pdf2text -a 1,10-20,50- in.pdf
```

will render the first page, pages in the range from 10 to 20 and all pages starting with page 50 to the last page in the document.

All even pages can be selected using the 'e' (or 'even') string. For example, the following line converts all even pages:

```
pdf2text --pages even in.pdf
```

Similarly odd pages can be selected using the 'o' (or 'odd') string. The following line converts all odd pages in the document and every page in the range from 100 to the last page:

```
pdf2text --pages odd,100- in.pdf
```

3.3.7 How do I batch convert PDF files?

PDF2Text supports batch conversion of many PDF files in a single pass. To convert all PDF files in a given folder(s) you can use the following syntax:

```
pdf2text myfolder1 myfolder2
```

The '--subfolders' option can be used to recursively process all subfolders. For example, the following line will convert all documents in 'myfolder1' and 'myfolder2' as well as all subfolders:

```
pdf2text --subfolders myfolder1 myfolder2
```

By default, PDF2Text will convert all files with the extension '.pdf'. To select different files based on the extension use the '--extension' parameter. For example, to convert all XPS documents with a custom extension '.blob', you could use the following line:

```
pdf2text --extension .blob --subfolders myfolder1
```

The use of wild characters is also allowed. For example, to convert all PDF files starting with 'x' in the current folder use:

```
pdf2text x*.pdf
```

3.3.8 How do I customize text extraction?

By default, PDF2Text will expand all ligatures in PDF. In writing and typography, a ligature occurs where two or more graphemes are joined as a single glyph. Use '--noligature' to disable ligature expansion. For example:

```
pdf2text --noligature mypdf
```

PDF files sometimes contain duplicated text to achieve visual effects of drop shadow and to fake bold text style. By default PDF2Text deletes duplicated overlapping text. To keep the duplicates, specify '--no_dup_remove' option on the command line. For example:

```
pdf2text --no_dup_remove mypdf
```

PDF2Text automatically remove hyphens in the original PDF file that are used for connecting split words across two lines. Use option '--nodehyphen' to disable word merging across lines. For example:

```
pdf2text --nodehyphen mypdf
```

PDF2Text provides several options related to the layout of text in the input PDF files.

In some cases, PDF documents may be missing spaces between punctuation characters and words may be merged into a single unit. To break words based on punctuation characters use '--punct_break' option. For example:

```
pdf2text --punct_break mypdf
```

In some cases, text in PDF may be obscured by images or rectangles. By default PDF2Text will extract this invisible text, however you can disable this behavior using '--remove_hidden_text' option. For example:

```
pdf2text --remove_hidden_text mypdf
```

Similarly some scanned PDF files or documents that went through OCR (Optical Character Recognition) may contain invisible text to facilitate text selection, highlighting, and text extraction. PDF2Text will automatically extract hidden text. To prevent text extraction of invisible text use '--remove_invisible_text' option. For example:

```
pdf2text --remove_invisible_text mypdf
```

In case you are looking for more flexibility or a more programmatic approach to text extraction, you may want to consider using PDFTron PDFNet SDK, as shown in the following sample code available at: http://www.pdftron.com/pdfnet/samplecode.html#TextExtract. PDFNet SDK offers a fine grained control over text extraction and access to low-level features in PDF documents.

3.3.9   How do I include styling information and how do I represent words as XML elements?

By default PDF2Text will expand all words into a single text line when converting to XML. In order to represent each word as a separate XML element with positioning and styling information use '--xml_words_as_elements' option. To include font and styling information for each word or line use 'xml_output_styles' option. For example, the default XML output for a given PDF may look as follows:

```
pdf2text my.pdf
```

```xml
<?xml version="1.0" encoding="utf-8" ?>
<Page num="1" crop_box="0, 0, 612, 792" media_box="0, 0, 612, 792" rotate="0">
  <Flow id="1">
    <Para id="0">
      <Line>PDFTron CPU License Agreement</Line>
    </Para>
    <Para id="1">
      <Line>IMPORTANT NOTICE: PLEASE READ ....</Line>
      <Line>SOFTWARE WHICH ACCOMPANIES…</Line>
    </Para>
  </Flow>
</Page>
```

Using '--xml_words_as_elements' and '--xml_output_styles' option the generated output is richer:

```
pdf2text –format xml  --xml_words_as_elements --xml_output_styles my.pdf
```

12 0 obj
/F2 1 Tf stream
%PDF-1.4 trailer<<
<<
B 0 1   q
%âãÏÓ
/Length 13 0 g

```xml
<?xml version="1.0" encoding="utf-8" ?>
  <Page num="1" crop_box="0, 0, 612, 792" media_box="0, 0, 612, 792" rotate="0">
    <Flow id="1">
      <Para id="0">
        <Line style="font-family:Verdana-Bold; font-size:9; color: #000000;">
         <Word>PDFTron</Word>
         <Word>CPU</Word>
         <Word>License</Word>
         <Word>Agreement</Word>
         </Line>
        </Para>
      <Para id="1">
        <Line style="font-family:Verdana-Bold; font-size:7.5; color: #000000;">
         <Word>IMPORTANT</Word>
```
… etc

3.3.10   How do I retrieve page information?

PDF2Text provides several options for to retrieve page information from existing PDF documents:

Use '--wordcount' option to retrieve number of words for each page. For example:

`pdf2text --wordcount my.pdf`

will retrieve number of words for each page in the specified document.

Use '--charcount' option to number of characters for each page. For example:

`pdf2text --charcount my.pdf`

will retrieve number of characters for each page in the specified document.

Use '--pageinfo' option to retrieve the width, height, media box, crop box, and rotation for every page in the document. For example:

`pdf2text --pageinfo my.pdf`

3.3.11   How do I extract text from a given rectangle on a PDF page?

Using PDF2Text you can extract text from a subset of a page using the '--clip' parameter. The parameter accepts a list of four numbers, separated using commas, giving the coordinates of a pair of diagonally opposite corners. Typically, the list takes the form: *llx, lly, urx, ury* specifying the lower-left *x*, lower-left *y*, upper-right *x*, and upper-right *y* coordinates of the rectangle, in that order. The other two corners of the rectangle are then assumed to have coordinates (*llx*, *ury*) and (*urx*, *lly*). All coordinates need to be expressed in points (a basic unit of PDF 'user' coordinate system). One PDF point is 1/72 of an inch and is approximately the same as a point (unit commonly used in the printing industry).

For example:

`pdf2text -c 150,600,250,700 license.pdf -a 1`

3.3.12   Does PDF2Text have any dependencies on third party components/software?

PDF2Text is a completely stand alone application and does not include any dependencies on third-party components or software.

3.4   General Usage Examples

Example 1.   The simplest command line: Convert PDF to plain text.

Notes:

- This command heavily relies on defaults. The default output image format is plain text.
- The '-o' (or --output) parameter is used to specify the output folder. If this option was not specified, text extracted will show in the console window.

```
pdf2text -o ex1 test/importantdoc.pdf
```

Example 2.   Convert specific PDF pages to XML, including font and styling information, while preserving ligatures and removing hidden text.

Notes:

- '-a' or '--pages' option is used to specify the pages to be converted.
- '-f' option specifies output file format.
- '--xml_output_styles' option is used to show font and styling information.
- '--noligatures' option is used to keep ligature setting of the PDF file.
- '--remove_hidden_text' option is used so that hidden text of the PDF file can be removed.
- '--output' is equal to '-o', specifies the output folder.

```
pdf2text --output ex2 -a 3-10 -f xml --xml_output_styles --noligatures --
remove_hidden_text test/impotantdoc.pdf
```

Example 3.   Extract PDF text runs from a given clip region from a password protected PDF.

```
pdf2text -f textruns -o ex3 –c 0,0,595,842 test/blue_secret.pdf
```

3.5 Batch Processing and the Use of Wildcards

PDF2Text supports processing of multiple input documents in the same run. For example, it is possible to specify multiple PDF folders and PDF2Text will automatically process all PDF documents matching a given file extension. For example, the following command-line will process all PDF documents in folders 'test1' and 'test2'

```
c:\>pdf2text -o c:/output_folder c:/test1 c:/test2
```

Wildcard characters can also be used to process multiple input files.

For example, if a directory contains the following PDF documents:

```
C:\test1 >dir
 Directory of C:\test1
01/04/2007  03:35 PM    <DIR>          .
01/04/2007  03:35 PM    <DIR>          ..
05/21/2004  02:27 PM           A1.pdf
05/03/2005  09:38 AM           A2.pdf
05/20/2003  08:46 AM           B1.pdf
05/15/2003  12:50 PM           B2.pdf
```

To process all PDF documents in this folder, you could specify:

```
c:\>pdf2text -o c:/output_folder c:/test1/*.pdf
```

To process all PDF documents staring with 'A', you could specify:

```
pdf2text -o c:/output_folder c:/test1/A*.pdf
```

Or to process all PDF documents ending with '1', you could specify:

```
pdf2text -o c:/output_folder c:/test1/*1.pdf
```

You can use either of the two standard wildcards — the question mark (?) and the asterisk (*) — to specify filename and path arguments on the command line.

The wildcards are expanded in the same manner as operating system commands. (Please refer to your operating system user's guide if you are unfamiliar with wildcards). Enclosing an argument in double quotation marks (" ") suppresses the wildcard expansion. Within quoted arguments, you can represent quotation marks literally by preceding the double-quotation-mark character with a backslash (\). If no matches are found for the wildcard argument, the argument is passed literally.

3.6   Exit Codes

To provide additional feedback, PDF2Text returns exit codes after completing processing. The exit codes can be used to provide user feedback, for logging etc. This is particularly important for applications running in an unattended environment.

The following table lists possible exit codes and their description:

Exit Code	Description
0	All files converted successfully.
1	Document is secured. Need a valid password to open the document.
2	Error opening the input file(s).
3	An unknown exception encountered.

All codes other then '0' indicate that there was an error during the conversion process.

The following illustrates a sample Windows batch script that processes exit codes:


```
@echo off rem convert all PDF files in 'data' folder

pdf2text ./data
if errorlevel 1 goto passwd
if errorlevel 2 goto inputerr
if errorlevel 3 goto othererror
if errorlevel 0 goto exit

:passwd
echo Document is protected. Need a valid password to open the document.
goto exit

:inputerr
echo No input files specified.
goto exit

:othererror
echo An error encountered during processing.
goto exit

:exit
```

4.   Support

4.1   Reporting Problems

If you encounter a problem or question regarding PDFTron PDF2Text, which is not addressed on PDFTron's website, please submit a problem report to PDFTron's Support group at http://www.pdftron.com/reportproblem.html.

When submitting a problem you will be asked to provide the following information:

- Contact details
- Product and Version of the product
- Detailed description of problem
- Problem file(s)
- Whether you have an AMS (Annual Maintenance Subscription)
- Any other information that may be related

4.2   Contact Information

To contact PDFTron directly, please use the contact information below:

Tel: 1-604-730-8989
Fax: 1-604-676-2477

Web site: www.pdftron.com

Email Contacts:

General Business Inquiries: info@pdftron.com
Sales & Licensing: sales@pdftron.com
Product Support: support@pdftron.com
Professional Services: services@pdftron.com
Website related questions: webmaster@pdftron.com
Press & News: press@pdftron.com