

PDFTK

May 11, 2018

NAME

pdftk - A handy tool for manipulating PDF

SYNOPSIS

```
pdftk <input PDF files / - / PROMPT>  
[ input_pw <input PDF owner passwords / PROMPT> ]  
[ <operation> <operation arguments> ]  
[ output <output filename / - / PROMPT> ]  
[ encrypt_40bit | encrypt_128bit ]  
[ allow <permissions> ]  
[ owner_pw <owner password / PROMPT> ]  
[ user_pw <user password / PROMPT> ]  
[ flatten ] [ need_appearances ]  
[ compress | uncompress ]  
[ keep_first_id | keep_final_id ] [ drop_xfa ] [ drop_xmp ]  
[ verbose ] [ dont_ask | do_ask ]
```

Where:

<operation> may be empty, or:

```
[ cat | shuffle | burst | rotate |  
generate_fdf | fill_form |  
background | multibackground |  
stamp | multistamp |  
dump_data | dump_data_utf8 |  
dump_data_fields | dump_data_fields_utf8 |  
dump_data_annots |  
update_info | update_info_utf8 |  
attach_files | unpack_files ]
```

For Complete Help: **pdftk --help**

DESCRIPTION

If PDF is electronic paper, then pdftk is an electronic staple-remover, hole-punch, binder, secret-decoder-ring, and X-Ray-glasses. Pdftk is a simple tool for doing everyday things with PDF documents. Use it to:

- * Merge PDF Documents or Collate PDF Page Scans
- * Split PDF Pages into a New Document
- * Rotate PDF Documents or Pages
- * Decrypt Input as Necessary (Password Required)
- * Encrypt Output as Desired
- * Fill PDF Forms with X/FDF Data and/or Flatten Forms
- * Generate FDF Data Stencils from PDF Forms
- * Apply a Background Watermark or a Foreground Stamp
- * Report PDF Metrics, Bookmarks and Metadata
- * Add/Update PDF Bookmarks or Metadata
- * Attach Files to PDF Pages or the PDF Document
- * Unpack PDF Attachments
- * Burst a PDF Document into Single Pages
- * Uncompress and Re-Compress Page Streams
- * Repair Corrupted PDF (Where Possible)

OPTIONS

A summary of options is included below.

--help, -h Show this summary of options.

<input PDF files | - | PROMPT> A list of the input PDF files. If you plan to combine these PDFs (without using handles) then list files in the order you want them combined. Use - to pass a single PDF into pdftk via stdin. Input files can be associated with handles, where a handle is one or more upper-case letters:

<input PDF handle>=<input PDF filename>

Handles are often omitted. They are useful when specifying PDF passwords or page ranges, later.

For example: A=input1.pdf QT=input2.pdf M=input3.pdf

[**input_pw** <input PDF owner passwords | **PROMPT**>] Input PDF owner passwords, if necessary, are associated with files by using their handles:

<input PDF handle>=<input PDF file owner password>

If handles are not given, then passwords are associated with input files by order.

Most pdftk features require that encrypted input PDF are accompanied by the ~owner~ password. If the input PDF has no owner password, then the user password must be given, instead. If the input PDF has no passwords, then no password should be given.

When running in **do_ask** mode, pdftk will prompt you for a password if the supplied password is incorrect or none was given.

[<operation> <operation arguments>] Available operations are: **cat**, **shuffle**, **burst**, **rotate**, **generate_fdf**, **fill_form**, **background**, **multibackground**, **stamp**, **multistamp**, **dump_data**, **dump_data_utf8**, **dump_data_fields**, **dump_data_fields_utf8**, **dump_data_annots**, **update_info**, **update_info_utf8**, **attach_files**, **unpack_files**. Some operations takes additional arguments, described below.

If this optional argument is omitted, then pdftk runs in 'filter' mode. Filter mode takes only one PDF input and creates a new PDF after applying all of the output options, like encryption and compression.

cat [<page ranges>] Assembles (catenates) pages from input PDFs to create a new PDF. Use **cat** to merge PDF pages or to split PDF pages from documents. You can also use it to rotate PDF pages. Page order in the new PDF is specified by the order of the given page ranges. Page ranges are described like this:

<input PDF handle>[<begin page number>[-<end page number>[<qualifier>]]][<page rotation>]

Where the handle identifies one of the input PDF files, and the beginning and ending page numbers are one-based references to pages in the PDF file. The qualifier can be **even** or **odd**, and the page rotation can be **north**, **south**, **east**, **west**, **left**, **right**, or **down**.

If a PDF handle is given but no pages are specified, then the entire PDF is used. If no pages are specified for any of the input PDFs, then the input PDFs' bookmarks are also merged and included in the output.

If the handle is omitted from the page range, then the pages are taken from the first input PDF.

The **even** qualifier causes pdftk to use only the even-numbered PDF pages, so **1-6even** yields pages 2, 4 and 6 in that order. **6-1even** yields pages 6, 4 and 2 in that order.

The **odd** qualifier works similarly to the **even**.

The page rotation setting can cause pdftk to rotate pages and documents. Each option sets the page rotation as follows (in degrees): **north**: 0, **east**: 90, **south**: 180, **west**: 270, **left**: -90, **right**: +90, **down**: +180. **left**, **right**, and **down** make relative adjustments to a page's rotation.

If no arguments are passed to cat, then pdftk combines all input PDFs in the order they were given to create the output.

NOTES:

- * *<end page number>* may be less than *<begin page number>*.
- * The keyword **end** may be used to reference the final page of a document instead of a page number.
- * Reference a single page by omitting the ending page number.
- * The handle may be used alone to represent the entire PDF document, e.g., B1-end is the same as B.
- * You can reference page numbers in reverse order by prefixing them with the letter **r**. For example, page r1 is the last page of the document, r2 is the next-to-last page of the document, and rend is the first page of the document. You can use this prefix in ranges, too, for example r3-r1 is the last three pages of a PDF.

Page Range Examples without Handles:

1-endeast - rotate entire document 90 degrees

5 11 20 - take single pages from input PDF

5-25oddwest - take odd pages in range, rotate 90 degrees

6-1 - reverse pages in range from input PDF

Page Range Examples Using Handles:

Say **A=in1.pdf B=in2.pdf**, then:

A1-21 - take range from in1.pdf

Bend-1odd - take all odd pages from in2.pdf in reverse order

A72 - take a single page from in1.pdf

A1-21 Beven A72 - assemble pages from both in1.pdf and in2.pdf

Awest - rotate entire in1.pdf document 90 degrees

B - use all of in2.pdf

A2-30evenleft - take the even pages from the range, remove 90 degrees from each page's rotation

A A - catenate in1.pdf with in1.pdf

Aevenwest Aoddeast - apply rotations to even pages, odd pages from in1.pdf

Awest Bwest Bdown - catenate rotated documents

shuffle [**<page ranges>**] Collates pages from input PDFs to create a new PDF. Works like the **cat** operation except that it takes one page at a time from each page range to assemble the output PDF. If one range runs out of pages, it continues with the remaining ranges. Ranges can use all of the features described above for **cat**, like reverse page ranges, multiple ranges from a single PDF, and page rotation. This feature was designed to help collate PDF pages after scanning paper documents.

burst Splits a single input PDF document into individual pages. Also creates a report named **doc_data.txt** which is the same as the output from **dump_data**. If the **output** section is omitted, then PDF pages are named: pg_%04d.pdf, e.g.: pg_0001.pdf, pg_0002.pdf, etc. To name these pages yourself, supply a printf-styled format string via the **output** section. For example, if you want pages named: page_01.pdf, page_02.pdf, etc., pass **output page_%02d.pdf** to pdftk. Encryption can be applied to the output by appending output options such as **owner_pw**, e.g.:

```
pdftk in.pdf burst owner_pw foopass
```

rotate [**<page ranges>**] Takes a single input PDF and rotates just the specified pages. All other pages remain unchanged. The page order remains unchanged. Specify the pages to rotate using the same notation as you would with **cat**, except you omit the pages that you aren't rotating:

```
[<begin page number>[-<end page number>[<qualifier>]]][<page rotation>]
```

The qualifier can be **even** or **odd**, and the page rotation can be **north**, **south**, **east**, **west**, **left**, **right**, or **down**.

Each option sets the page rotation as follows (in degrees): **north**: 0, **east**: 90, **south**: 180, **west**: 270, **left**: -90, **right**: +90, **down**: +180. **left**, **right**, and **down** make relative adjustments to a page's rotation.

The given order of the pages doesn't change the page order in the output.

generate_fdf Reads a single input PDF file and generates an FDF file suitable for **fill_form** out of it to the given output filename or (if no output is given) to stdout. Does not create a new PDF.

fill_form <**FDF data filename** | **XFDF data filename** | - | **PROMPT**>

Fills the single input PDF's form fields with the data from an FDF file, XFDF file or stdin. Enter the data filename after **fill_form**, or use - to pass the data via stdin, like so:

```
pdftk form.pdf fill_form data.fdf output form.filled.pdf
```

If the input FDF file includes Rich Text formatted data in addition to plain text, then the Rich Text data is packed into the form fields *as well as* the plain text. Pdftk also sets a flag that cues Reader/Acrobat to generate new field appearances based on the Rich Text data. So when the user opens the PDF, the viewer will create the Rich Text appearance on the spot. If the user's PDF viewer does not support Rich Text, then the user will see the plain text data instead. If you flatten this form before Acrobat has a chance to create (and save) new field appearances, then the plain text field data is what you'll see.

Also see the **flatten** and **need_appearances** options.

background <**background PDF filename** | - | **PROMPT**>

Applies a PDF watermark to the background of a single input PDF. Pass the background PDF's filename after **background** like so:

```
pdftk in.pdf background back.pdf output out.pdf
```

Pdftk uses only the first page from the background PDF and applies it to every page of the input PDF. This page is scaled and rotated as needed to fit the input page. You can use - to pass a background PDF into pdftk via stdin.

If the input PDF does not have a transparent background (such as a PDF created from page scans) then the resulting background won't be visible -- use the **stamp** operation instead.

multibackground <background PDF filename | - | PROMPT>

Same as the **background** operation, but applies each page of the background PDF to the corresponding page of the input PDF. If the input PDF has more pages than the stamp PDF, then the final stamp page is repeated across these remaining pages in the input PDF.

stamp <stamp PDF filename | - | PROMPT> This behaves just like the **background** operation except it overlays the stamp PDF page *on top* of the input PDF document's pages. This works best if the stamp PDF page has a transparent background.

multistamp <stamp PDF filename | - | PROMPT>

Same as the **stamp** operation, but applies each page of the background PDF to the corresponding page of the input PDF. If the input PDF has more pages than the stamp PDF, then the final stamp page is repeated across these remaining pages in the input PDF.

dump_data Reads a single input PDF file and reports its metadata, bookmarks (a/k/a outlines), page metrics (media, rotation and labels), data embedded by STAMPtk (see STAMPtk's **embed** option) and other data to the given output filename or (if no output is given) to stdout. Non-ASCII characters are encoded as XML numerical entities. Does not create a new PDF.

dump_data_utf8 Same as **dump_data** except that the output is encoded as UTF-8.

dump_data_fields Reads a single input PDF file and reports form field statistics to the given output filename or (if no output is given) to stdout. Non-ASCII characters are encoded as XML numerical entities. Does not create a new PDF.

dump_data_fields_utf8 Same as **dump_data_fields** except that the output is encoded as UTF-8.

dump_data_annots **This operation currently reports only link annotations.** Reads a single input PDF file

and reports annotation information to the given output filename or (if no output is given) to stdout. Non-ASCII characters are encoded as XML numerical entities. Does not create a new PDF.

update_info <info data filename | - | PROMPT>

Changes the bookmarks and metadata in a single PDF's Info dictionary to match the input data file. The input data file uses the same syntax as the output from **dump_data**. Non-ASCII characters should be encoded as XML numerical entities.

This operation does not change the metadata stored in the PDF's XMP stream, if it has one. (For this reason you should include a **ModDate** entry in your updated info with a current date/timestamp, format: **D:YYYYMMDDHHmmSS**, e.g. **D:201307241346** -- omitted data after YYYY revert to default values.)

For example:

```
pdftk in.pdf update_info in.info output out.pdf
```

update_info_utf8 <info data filename | - | PROMPT>

Same as **update_info** except that the input is encoded as UTF-8.

attach_files <attachment filenames | PROMPT> [to_page <page

Packs arbitrary files into a PDF using PDF's file attachment features. More than one attachment may be listed after **attach_files**. Attachments are added at the document level unless the optional **to_page** option is given, in which case the files are attached to the given page number (the first page is 1, the final page is **end**). For example:

```
pdftk in.pdf attach_files table1.html table2.html to_page 6  
output out.pdf
```

unpack_files Copies all of the attachments from the input PDF into the current folder or to an output directory given after **output**. For example:

```
pdftk report.pdf unpack_files output ~/atts/
```

or, interactively:

pdftk report.pdf **unpack_files** output PROMPT

[**output** <output filename | - | PROMPT>] The output PDF filename may not be set to the name of an input filename. Use - to output to stdout. When using the **dump_data** operation, use **output** to set the name of the output data file. When using the **unpack_files** operation, use **output** to set the name of an output directory. When using the **burst** operation, you can use **output** to control the resulting PDF page filenames (described above).

[**encrypt_40bit** | **encrypt_128bit**] If an output PDF user or owner password is given, output PDF encryption strength defaults to 128 bits. This can be overridden by specifying **encrypt_40bit**.

[**allow** <permissions>] Permissions are applied to the output PDF only if an encryption strength is specified or an owner or user password is given. If permissions are not specified, they default to 'none,' which means all of the following features are disabled.

The **permissions** section may include one or more of the following features:

Printing Top Quality Printing

DegradedPrinting Lower Quality Printing

ModifyContents Also allows Assembly

Assembly

CopyContents Also allows ScreenReaders

ScreenReaders

ModifyAnnotations Also allows FillIn

FillIn

AllFeatures Allows the user to perform all of the above, and top quality printing.

[**owner_pw** <owner password | PROMPT>]

[**user_pw** <user password | PROMPT>] If an encryption strength is given but no passwords are supplied, then the owner and user passwords remain empty, which means that the resulting PDF may be opened and its security parameters altered by anybody.

[**compress** | **uncompress**] These are only useful when you want to edit PDF code in a text editor like vim or emacs. Remove PDF page stream

compression by applying the **uncompress** filter. Use the **compress** filter to restore compression.

[flatten] Use this option to merge an input PDF's interactive form fields (and their data) with the PDF's pages. Only one input PDF may be given. Sometimes used with the **fill_form** operation.

[need_appearances] Sets a flag that cues Reader/Acrobat to generate new field appearances based on the form field values. Use this when filling a form with non-ASCII text to ensure the best presentation in Adobe Reader or Acrobat. It won't work when combined with the **flatten** option.

[keep_first_id | keep_final_id] When combining pages from multiple PDFs, use one of these options to copy the document ID from either the first or final input document into the new output PDF. Otherwise pdftk creates a new document ID for the output PDF. When no operation is given, pdftk always uses the ID from the (single) input PDF.

[drop_xfa] If your input PDF is a form created using Acrobat 7 or Adobe Designer, then it probably has XFA data. Filling such a form using pdftk yields a PDF with data that fails to display in Acrobat 7 (and 6?). The workaround solution is to remove the form's XFA data, either before you fill the form using pdftk or at the time you fill the form. Using this option causes pdftk to omit the XFA data from the output PDF form.

This option is only useful when running pdftk on a single input PDF. When assembling a PDF from multiple inputs using pdftk, any XFA data in the input is automatically omitted.

[drop_xmp] Many PDFs store document metadata using both an Info dictionary (old school) and an XMP stream (new school). Pdftk's **update_info** operation can update the Info dictionary, but not the XMP stream. The proper remedy for this is to include a **ModDate** entry in your updated info with a current date/timestamp. The date/timestamp format is: **D:YYYYMMDDHHmmSS**, e.g. D:201307241346 -- omitted data after YYYY revert to default values. This newer ModDate should cue PDF viewers that the Info metadata is more current than the XMP data.

Alternatively, you might prefer to remove the XMP stream from the PDF altogether -- that's what this option does. Note that objects inside the PDF might have their own, separate XMP metadata streams, and that

drop_xmp does not remove those. It only removes the PDF's document-level XMP stream.

[**verbose**] By default, pdftk runs quietly. Append **verbose** to the end and it will speak up.

[**dont_ask** | **do_ask**] Depending on the compile-time settings (see `ASK_ABOUT_WARNINGS`), pdftk might prompt you for further input when it encounters a problem, such as a bad password. Override this default behavior by adding **dont_ask** (so pdftk won't ask you what to do) or **do_ask** (so pdftk will ask you what to do).

When running in **dont_ask** mode, pdftk will over-write files with its output without notice.

EXAMPLES

Collate scanned pages `pdftk A=even.pdf B=odd.pdf shuffle A B output collated.pdf`

or if odd.pdf is in reverse order:

`pdftk A=even.pdf B=odd.pdf shuffle A Bend-1 output collated.pdf`

Decrypt a PDF `pdftk secured.pdf input_pw foopass output unsecured.pdf`

Encrypt a PDF using 128-bit strength (the default), withhold all permissions

`pdftk 1.pdf output 1.128.pdf owner_pw foopass`

Same as above, except password 'baz' must also be used to open output

`pdftk 1.pdf output 1.128.pdf owner_pw foo user_pw baz`

Same as above, except printing is allowed (once the PDF is open)

`pdftk 1.pdf output 1.128.pdf owner_pw foo user_pw baz allow printing`

Join in1.pdf and in2.pdf into a new PDF, out1.pdf `pdftk in1.pdf in2.pdf cat output out1.pdf`

or (using handles):

`pdftk A=in1.pdf B=in2.pdf cat A B output out1.pdf`

or (using wildcards):

`pdftk *.pdf cat output combined.pdf`

Remove page 13 from in1.pdf to create out1.pdf `pdftk in.pdf cat 1-12 14-end output out1.pdf`

or:

`pdftk A=in1.pdf cat A1-12 A14-end output out1.pdf`

Apply 40-bit encryption to output, revoking all permissions (the default)

```
pdftk 1.pdf 2.pdf cat output 3.pdf encrypt_40bit owner_pw foopass
```

Join two files, one of which requires the password 'foopass'. The output

```
pdftk A=secured.pdf 2.pdf input_pw A=foopass cat output 3.pdf
```

Uncompress PDF page streams for editing the PDF in a text editor (e.g.

```
pdftk doc.pdf output doc.unc.pdf uncompress
```

Repair a PDF's corrupted XREF table and stream lengths, if possible

```
pdftk broken.pdf output fixed.pdf
```

Burst a single PDF document into pages and dump its data to doc_data.pdf

```
pdftk in.pdf burst
```

Burst a single PDF document into encrypted pages. Allow low-quality printing

```
pdftk in.pdf burst owner_pw foopass allow DegradedPrinting
```

Write a report on PDF document metadata and bookmarks to report.txt

```
pdftk in.pdf dump_data output report.txt
```

Rotate the first PDF page to 90 degrees clockwise

```
pdftk in.pdf cat least 2-end output out.pdf
```

Rotate an entire PDF document to 180 degrees

```
pdftk in.pdf cat 1-endsouth output out.pdf
```

NOTES

This is a port of pdftk to java. See

<https://gitlab.com/marcvinyals/pdftk>

The original program can be found at www.pdftk.com

AUTHOR

Original author of pdftk is Sid Steward (sid.steward at pdflabs dot com).