PDFlib GmbH

pCOS Command-Line Tool

pCOS Interface 12

PDF Information Retrieval Tool

PDFlib GmbH
Franziska-Bilek-Weg 9, 80339 München, Germany
www.pdflib.com
phone +49 • 89 • 452 33 84-0

If you have questions check the PDFlib mailing list at
groups.yahoo.com/neo/groups/pdflib/info

Licensing contact: sales@pdflib.com
Support for commercial PDFlib licensees: support@pdflib.com (please include your license number)

Contents

1 pCOS Command-Line Examples

The pCOS command-line tool allows you to query information from one or more PDF documents without the need for programming. In addition, it can be used as a frontend to the pCOS interface. The pCOS command-line tool is built on top of the pCOS interface. In this chapter we present sample calls of the pCOS tool. We start with simple examples and proceed to more complex applications. A detailed list of all command-line options can be found in Chapter 2, »pCOS Command-Line Reference«, page 13.

The pCOS programming interface can be used to integrate pCOS queries in applications written in a variety of programming languages. The pCOS interface uses pCOS paths as its workhorse; these are discussed in a separate manual, the pCOS Path Reference.

Applying the license key. The pCOS command-line tool is activated via a valid PLOP or PLOP DS license key. Please refer to the PLOP manual which discusses several methods for supplying the license key in a license file or the Windows registry. If no valid PLOP or PLOP DS license key is found, the pCOS command-line tool will operate in evaluation mode.

The default file search paths described in the PLOP manual also apply to the pCOS command-line tool.

1.1 For Starters: simple Mode

The first command does not use any options, which means that general information plus all document info entries are listed:

```
pcos file.pdf
```

The following command lists all fonts used in the document along with their type and embedding status:

```
pcos --font file.pdf
```

The following command creates a hierarchical list of all form fields in the document along with their field type and the field value:

```
pcos --field file.pdf
```

The following command creates a hierarchical list of all bookmarks in the document:

```
pcos --bookmark file.pdf
```

The following command lists the width and height of all pages as well as relevant Box entries (e.g. CropBox) and rotation:

```
pcos --pagesize file.pdf
```

The following command emits information about the PDF/X and PDF/A status of the document:

```
pcos --pdfx --pdfa file.pdf
```

The following command emits information about the PDF/UA status of the document:

```
pcos --pcospath pdfua file.pdf
```

The following command lists all web links on the first two pages:

```
pcos --firstpage 1 --lastpage 2 --weblink file.pdf
```

The following command lists all digital signature fields along with relevant details:

```
pcos --signature file.pdf
```

Understanding pCOS paths in the generated output. In many cases pCOS creates output which not only includes text and numbers found in the PDF document, but also emits pCOS paths which designate an object within the PDF object hierarchy. While the pCOS path syntax is discussed in detail in the pCOS Path Reference, here are a few important notes based on sample output.

The *--weblink* option creates output similar to the following line. The first column contains the pCOS path, while the second column contains the URL. It is important to note that in pCOS syntax page numbering starts at 0, i.e. the first page is designated as *pages[0]*. Similarly, annotations are numbered starting from 0:

```
pages[0]/annots[0]/A/URI: http://www.pdflib.com
```

1.2 Extracting Data from PDF

Note *Our product TET (Text Extraction Toolkit) can be used to extract text and image contents from PDF pages. Text and images can not be extracted with pCOS.*

The pCOS command-line tool can be used to extract various data items from PDF documents. The extracted data items are written to disk files with unique names (based on the name of the input PDF, the data type, and increasing numbers). This section lists several examples for PDF data extraction; see Section 2.4, »Options for Retrieving PDF Elements«, page 17, for more detailed option descriptions.

The following command extracts all file attachments (on page level) in the document:

```
pcos --extract attachment file.pdf
```

The following command extracts all file attachments (on document level) in the document:

```
pcos --extract embeddedfile file.pdf
```

The following command extracts all JavaScripts in the document. Note that a particular script can be used in more than one places (e.g. validation scripts for form fields). In this case the script is extracted more than once:

```
pcos --extract javascript file.pdf
```

The following command extracts the output intent ICC profile of a PDF/X or PDF/A file:

```
pcos --extract outputintent file.pdf
```

The following command extracts document-level XMP metadata to a file:

```
pcos --extract metadata file.pdf
```

1.3 For advanced Applications: extended Mode

In this section we will present commands which use the extended output mode of pCOS and options for advanced formatting control.

Text output. The following command lists all annotations (links and other types) with their Subtype, destination within the document, the target URL, and the link rectangle coordinates on the page. Double quotes must surround the list of annotation keys since they must be supplied as a single argument to the program:

```
pcos --extended annotation "Subtype destpage A/URI Rect" file.pdf
```

If you have a file with comments from a review process you can list the text in the comments along with the reviewers' name with the following command. The *PP* variable at the start of the formatting string will create the corresponding pCOS path which includes the page number and the annotation number (both starting at 0). The *KEY* variable denotes the key (name) of a dictionary entry, which usually is a PDF name object; the *VAL* variable refers to the corresponding value which may have any type. The parenthesis around the key/value pair mean that this expression is repeated for all entries in the annotation dictionary:

```
pcos --format "PP (KEY=VAL )\n" --extended annotation "Subtype Contents T" file.pdf
```

The following command lists all file attachments (embedded files):

```
pcos --format "(KEY=VAL )\n" --extended attach "Subtype Contents T Name" file.pdf
```

The following command lists the file name and Author for multiple files. The default headline is disabled since we included the name of the input file (variable *IF)* in the format string:

```
pcos --headline "" --format "IF:(VAL\n)" --extended docinfo Author *.pdf
```

The following command lists important properties of PDFlib blocks. Double quotes are used to avoid problems with space characters in block names:

```
pcos --bracket dquot --format "(KEY=VAL\n)\n" --extended block "Name Subtype Description"
        file.pdf
```

The following command creates a table of contents from the bookmark titles and corresponding page numbers; this only works if the bookmarks actually point to a page:

```
pcos --indent 4 --format "(VAL )\n" --extended bookmark "Title destpage" file.pdf
```

The following command lists the names of all named destinations along with the corresponding target page. The pCOS path (variable *PP)* contains the destination name:

```
pcos --format "PP: page VAL\n" --extended destination destpage file.pdf
```

Tabular output for use in spreadsheet applications. Using the formatting options of pCOS it is easy to create output which can be processed in applications such as Microsoft Excel. The following commands create comma-separated lists of various pieces of information retrieved from an arbitrary number of PDF documents. The required comma and newline characters are created using suitable format strings. The output can be imported in Microsoft Excel and similar spreadsheet applications which support the CSV (comma-separated values) format.

The following command creates a table with the pCOS path (variable *PP)* containing the page number (starting at 0) in the first column, and the width and height of each page in subsequent columns:

```
pcos --outfile table.csv --format "PP,(VAL,)\n" --extended pagesize "width height"
        file.pdf
```

The following command extends the previous example for use with many files; it creates a table with the file names of all input files (variable *IF)* along with the pCOS path (variable *PP)* and the size of all pages. It suppresses the default headline since the input file name is already printed in the first column of each output line:

```
pcos --outfile table.csv --headline "" --format "IF,PP,(VAL,)\n"
        --extended pagesize "width height" file.pdf
```

The following command creates a table of PDFlib block names, types, and position:

```
pcos --outfile table.csv --bracket dquot --format "(VAL,)\n"
        --extended block "Name Subtype fontname Rect[0] Rect[1] Rect[2] Rect[3]" file.pdf
```

The following command creates a table containing the file names (created by the *IF* variable) and various document info entries:

```
pcos --outfile table.csv --replace missing "" --bracket dquot --headline ""
        --format "IF,(VAL,)\n" --extended docinfo "Title Author Creator Subject" *.pdf
```

The following command creates a table with type, name, and value of form fields. In order to avoid unwanted whitespace we set the indentation to 0. A headline with the names of the extracted field keys is placed at the top. Missing entries are designate with a custom string:

```
pcos --outfile table.csv --indent 0 --headline "FT,fullname,V\n"
        --replace missing "(unavailable)" --format "(VAL,)\n"
        --extended field "FT fullname V" file.pdf
```

The following command creates a table of file names along with all fonts and their embedding status. We place the input file name (variable *IF)* in the first column of each line, and disable the default heading (which would place the input file name on a separate line) by specifying an empty headline:

```
pcos --outfile table.csv --headline "" --bracket dquot --format "IF,(VAL,)\n"
        --extended font "name type embedded" file.pdf
```

The following command creates a table of all Web links (URL and position). The pCOS path in the first column (variable *PP)* contains the page and annotation numbers (0-based):

```
pcos --outfile table.csv --format "PP,(VAL,)\n"
        --extended weblink "A/URI Rect[0] Rect[1] Rect[2] Rect[3]" file.pdf
```

Querying all keys in a dictionary object. Using the »*xx*« special key you can list all keys which are contained in a dictionary without having to know in advance the name of the keys.

The following command lists all entries in the PDFlib block dictionaries (generally this is all required entries and those with a non-default value, since the PDFlib Block plugin omits properties which have their default value):

```
pcos --format "(KEY=VAL\n)\n" --extended block xx file.pdf
```

The following command lists all entries in all font dictionaries:

```
pcos --bracket round --format "(KEY=VAL\n)\n" --extended font xx file.pdf
```

1.4 For Experts: raw pCOS Paths

The following command prints the total number of fonts in the document; using the pCOS paths *length:bookmarks, length:pages,* or *length:fields* you can check the number of bookmarks, pages, or form fields, respectively:

```
pcos --pcospath "length:fonts" file.pdf
```

The following command extracts an embedded Distiller job options file:

```
pcos --outfile embedded.joboptions --pcospath "names/EmbeddedFiles[0]/EF/F" file.pdf
```

The following command dumps information about the version of PDFlib blocks on the first page, and the version of the Block plugin used to create the blocks:

```
pcos --format "PP=VAL\n" --pcospath "pages[0]/PieceInfo/PDFlib/Private/Version"
        --pcospath "pages[0]/PieceInfo/PDFlib/Private/PluginVersion" file.pdf
```

The following command prints the number of annotations on the first page:

```
pcos --pcospath "length:pages[0]/Annots" file.pdf
```

The following command extracts the first file attachment on the first page:

```
pcos --outfile attachment.txt --pcospath "pages[0]/Annots[0]/FS/EF" file.pdf
```

The following command extracts the CMS object with cryptographic details in DER format from a signed document:

```
pcos --binary --pcospath signaturefields[0]/V/Contents[0]/V/Contents ↵
        --outfile signature.der input.pdf
```

The extracted DER-encoded CMS object can further be analyzed, e.g. with the OpenSSL command-line tool:

```
openssl cms -in signature.der -inform DER -cmsout -print
```

2 pCOS Command-Line Reference

2.1 Option Processing and Exit Codes

The pCOS program can be controlled via a number of command-line options. It is called as follows for one or more input PDF files:

```
pcos [<options>] <filename>...
```

Constructing pCOS command lines. The following rules must be observed for constructing pCOS command lines:

- Input files are searched in all directories specified as *searchpath*.
- Short forms are available for some options, and can be mixed with long options.
- Long options can be abbreviated provided the abbreviation is unique (e.g. *--last* instead of *--lastpage)*
- Depending on encryption status of the input file, a user or master password may be required. This can be supplied with the *--password* option. pCOS will check whether this password is sufficient for the requested operation.

pCOS checks the full command line before processing any file. If an error is encountered in the options anywhere on the command line, no files are processed at all.

File names. File names which contain blank characters require some special handling when used with command-line tools like pCOS. In order to process a file name with blank characters you should enclose the complete file name with double quote " characters. Wildcards can be used according to standard practice. For example, **.pdf* denotes all files in a given directory which have a *.pdf* file name suffix. Note that on some systems case is significant, while on others it isn't (i.e., **.pdf* may be different from **.PDF).* Also note that on Windows systems wildcards do not work for file names containing blank characters. Wildcards are evaluated in the current directory, not any *searchpath* directory.

On Windows all file name options accept Unicode strings, e.g. as a result of dragging files from the Explorer to a command prompt window.

Response files. In addition to options supplied directly on the command-line, options can also be supplied in a response file. The contents of a response file will be inserted in the command-line at the location where the *@filename* option was found.

A response file is a simple text file with options and parameters. It must adhere to the following syntax rules:

- Option values must be separated with whitespace, i.e. space, linefeed, return, or tab.
- Values which contain whitespace must be enclosed with double quotation marks: "
- Double quotation marks at the beginning and end of a value will be omitted.
- A double quotation mark must be masked with a backslash to use it literally: \"
- A backslash character must be masked with another backslash to use it literally: \\

Response files can be nested, i.e. the *@filename* syntax can be used in another response file.

Exit codes. The pCOS command-line tool returns with an exit code which can be used to check whether or not the requested operations could be successfully carried out:

- Exit code 0: all command-line options could be successfully and fully processed.
- Exit code 1 (parser warning): the parser detected a problem in the command-line options, but continued after issuing a warning (e.g. wrong verbosity number)
- Exit code 2 (parser error): the parser detected a fatal problem in the command-line options, and stopped.
- Exit code 3: a warning was issued while processing the input, but processing continues.
- Exit code 4: an error was found while processing the input, processing stopped.

Encrypted PDF. All objects can be queried if the proper master password has been supplied with the *--password* option. If no password or only the user password has been supplied some objects are available, while others are not. Refer to the pCOS Path Reference for details on PDF security and pCOS modes.

2.2 Option Handling

Table 2.2 lists options related to general option handling.

Table 2.1 pCOS command-line options related to option handling

option	*parameters*	*function*
--		*End the list of options; this is useful in case file names start with a »-« character.*
***@filename*[1]**		*Specify a response file with options; for a syntax description see »Response files«, page 13. Response files will only be recognized before the -- option and before the first filename, and can not be used to replace the parameter for another option.*

1. This option can be supplied more than once.

2.3 Input Options

Table 2.2 lists options related to the input or general processing.

Table 2.2 pCOS command-line options related to input or general processing

option	parameters	function
--docopt	<*option list*>	*Additional option list for PLOP_open_document( )*
--firstpage	*1, 2, ..., last*	*The number of the page where page-related processing will start. The keyword* `last` *can be used to specify the last page. Default: 1*
--lastpage	*1, 2, ..., last*	*The number of the page where page-related processing will finish. The keyword* `last` *can be used to specify the last page. Default:* `last`
--password, -p	<*password*>	*User or master password for encrypted documents*
--plopopt	<*option list*>	*Additional option list for PLOP_set_option( ). This can be used to pass the* `license` *or* `licensefile` *options.*
--pcosopt	<*option list*>	*(Unsupported) Same as* `--plopopt`

2.4 Options for Retrieving PDF Elements

Table 2.3 lists options for simple output retrieval (there are no short option forms nor parameters in this group). Multiple retrieval options can be provided in a single call. In this case output will be created in the following order: first, the *--general* and *--docinfo* options will be processed (if supplied), and then all other retrieval options in Table 2.3 and Table 2.4 in the order in which they have been specified on the command line. If no retrieval option has been provided, the default *--general --docinfo* is used.

All options in Table 2.3 except *--general* require full pCOS mode, i.e. the master password must be provided for encrypted files.

Table 2.3 pCOS command-line options for simple output retrieval

option	function
--annotation[1]	*Contents and type of annotations. This option queries the keys* `Contents` *and* `Subtype` *in* `pages[...]/annots` *for all pages, using the format* `PP/KEY: VAL\n`.
--attachment[1]	*Description and file name of file attachments on the pages (see also* `--embeddedfile`*). This option queries the keys* `Contents`, `FS/F`, *and* `FS/UF` *in* `pages[...]/annots` *for all pages (if* `FS` *is present), using the format* `PP/KEY: VAL\n`. *The actual contents of a file attachment can be retrieved via* `--extract attachment`.
--block[1]	*Name and subtype of PDFlib Blocks for use with the PDFlib Personalization Server (PPS). This option queries the keys* `Name` *and* `Subtype` *in* `pages[...]/PieceInfo/PDFlib/Private/Blocks` *for all pages, using the format* `KEY: VAL\n`.
--bookmark	*Names of bookmarks. This option queries the key* `Title` *in* `bookmarks[...]`*, using the format* `VAL\n`*, and* `bookmarks[...]/level` *for indentation.* *The target page of a bookmark can be retrieved via* `bookmarks[...]/destpage`.
--destination	*Names and destination pages of named destinations. This option queries all keys in* `names[...]/Dest` *(i.e. all named destinations) and the value of the* `destpage` *subkey, using the format* `PP/KEY: VAL\n`.
--docinfo	*Key and value of document info entries. This option queries all keys in* `/Info`*, using the format* `KEY: VAL\n`.
--embeddedfile	*File name and description of named embedded files. This option queries document-level file attachments, while* `--attachment` *will retrieve file attachments on the page level. This option queries the keys* `F`, `UF`, *and* `Desc` *in* `names/EmbeddedFiles/*`*, using the format* `PP/KEY: VAL\n`. *The actual contents of an embedded file can be retrieved via* `--extract embeddedfile`.
--field	*Names, types, and values of form fields. This option queries the keys* `type`, `fullname` *and* `value` *in* `fields[...]`*, using the format* `PP/KEY: VAL\n`*, and* `fields[...]/level` *for indentation.*
--font	*Names, types, and embedding status of fonts. This option queries the keys* `name`, `type`, *and* `embedded` *in* `fonts[...]`*, using the format* `PP/KEY: VAL\n`.
--general	*File name and size, PDF version, encryption status, master/user password, linearization status, PDF/X, PDF/A, XFA, tagged status, signature details, Reader-enabled status, portfolio status, number of pages, number of fonts (page and font count are only available in full pCOS mode), document info fields, presence of XMP metadata, and presence of encrypted attachments. This option queries various real and pseudo objects.*

Table 2.3 pCOS command-line options for simple output retrieval

option	function
--javascript	*JavaScript at various locations in the document. For each script its length (in Unicode characters) is printed, as well as the total number of scripts found. Depending on the location of the JavaScript in the document, additional information is printed:* *Document open actions: JavaScript which will activated when the document is opened.* *Bookmarks: JavaScript for bookmark activation.* *Document-level JavaScript: additional information for the trigger event* (didprint, didsave, willclose, willprint, willsave) *Page-level JavaScript: additional information for the trigger event* (open, close) *JavaScript for annotation activation. Additional information: page number, annotation type* *Field-level JavaScript. Additional information: form field name, trigger* (activate, keystroke, format, validate, calculate, enter, exit, down, up, focus, blur)
--layer	*Names of all layers in the document. This may include unused layers and layers which are not visible in Acrobat's user interface (e.g. layers which do not require any interaction because they are controlled by JavaScript). This option queries the key* Name *in* /Root/OCProperties/OCGs*, using the format* VAL\n.
--layer-default	*Names of layers which are presented by default in Acrobat's layer pane (not related to the visibility of layer contents on the page). Only layers which are presented to the user is shown, using indentation to visualize the layer hierarchy. Text labels for grouping (which do not directly resemble a layer) will also be printed. Use* --layer *to catch all layers, regardless of their presence in the user interface. This option queries the key* Name *in* /Root/OCProperties/D/Order*, using the format* VAL\n.
--outputintent	*Properties of one or more output intent ICC profilesfor PDF/X and PDF/A. This option queries various keys in the* /Root/OutputIntents[...] *dictionary, using the format* PP/KEY: VAL\n.
--pagefield	*Names, types, and values of form fields listed by page. This option queries the keys* exportvalue, fullname, Rect, type *and* value *in* pages[]/fields[]*, using the format* PP/KEY: VAL\n.
--pagesize[1]	*Width, height, and various boxes describing the page dimensions. This option queries the keys* width, height, MediaBox, CropBox, *and* Rotate *in* pages[...] *for all pages, using the format* PP/KEY: VAL\n.
--pdfa	*PDF/A version and output intent name (no validation). This option queries the part, conformance, and amd (amendment) keys in the* pdfaid *section of the document's XMP metadata* (/Root/Metadata) *if present. If the file conforms to any of the PDF/A-1 standards, the corresponding keys* /Root/OutputIntents[...]/OutputConditionIdentifier *and* /Root/OutputIntents[...]/Info *are queried as well.*
--pdfua	*PDF/UA version (no validation) This option queries the part key in the* pdfuaid *section of the document's XMP metadata* (/Root/Metadata) *if present.*
--pdfvt	*(PDF/VT version (no validation) This option queries the part key in the* pdfvt *section of the document's XMP metadata* (/Root/Metadata) *if present.*
--pdfx	*PDF/X version and output intent name (no validation). This option first queries the key* /Info/GTS_PDFXVersion. *If the file conforms to any of the PDF/X standards, the corresponding keys* /Root/OutputIntents[...]/OutputConditionIdentifier *and* /Root/OutputIntents[...]/Info *are queried as well.*
--signature	*Signature information: name and visibility of all signature fields, signed/unsigned status, and signature details for signed fields. This option queries the keys* fullname, sigtype, visible, permissions, *and* cades *in* fields[...]*as well as the* usagerights *pseudo object.*
--weblink[1]	*Contents and URL of web links. This option queries the keys* Contents *and* A/URI *in* pages[...]/annots *for all pages (if* A/URI *is present), using the format* PP/KEY: VAL\n.
--xfa	*Checks whether the documents contains any XFA information* (eXtensible Forms Architecture). *This option queries the key* /Root/AcroForm/XFA.

1. This option is subject to the --firstpage *and* --lastpage *options.*

2.5 Advanced Retrieval Options

Table 2.4 lists options for advanced output retrieval. If pCOS runs in minimum or restricted mode, i.e. the master password has not been provided for an encrypted file, not all objects may be available (see the pCOS Path Reference for details). If the path designates a simple object, its value is printed, dictionary objects are enumerated recursively up to the level specified with *--depth*, and array objects are completely enumerated recursively.

Table 2.4 pCOS command-line options for advanced output retrieval

option	*parameters*	*function*
--binary		*Retrieved string objects are treated as binary data, i.e. will not be subject to Unicode and EBCDIC conversions. This option is useful for binary string data, e.g.* `Contents` *of a signature dictionary; it is not required for stream data since these are always treated in binary mode.*
--extended[1]	<*type*> <*keys*>	*Extended object retrieval for one of the following types:* `annotation`, `attachment`, `block`, `bookmark`, `destination`, `docinfo`, `embeddedfile`, `field`, `font`, `layer`, `pagefield`, `pagesize`, `signature`, `weblink` <keys> *contains a list of keys to be retrieved from the respective object(s). Use* xx *to query all existing keys (excluding pseudo keys if they exist for an object, e.g. a font dictionary, and some low-level bookkeeping keys for maintaining tree structures). The list of keys must be provided as a single command-line argument (in some environments this requires surrounding double quotes).*
--extract[1]	<*type*>	*Extract the binary data associated with one of the following types and print general information about the items):* ***attachment*** *All file attachments on page level (takes into account the* `--firstpage` *and* `--lastpage` *options)* ***embeddedfile*** *All file attachments on document level* ***font*** *All embedded fonts*[2] ***javascript*** *All JavaScripts for document open action, bookmarks, document-level scripts, page-level scripts, annotation activation, and fields.* ***metadata*** *XMP document metadata (without any format conversion)* ***outputintent*** *All output intent ICC profiles* ***signature*** *All signature values, i.e. the* `Contents` *entry of signature field values. It contains a PKCS#7/CMS object.* *Each data item is written to a separate disk file. Starting at the directory specified with the* `--targetdir` *option, a directory is created using the name of the input PDF (without any* `.pdf` *or* `.PDF` *suffix, and with critical characters replaced with »_«). Within this directory various subdirectories for the data items are created. The* `--outfile` *option is ignored.* *In addition to the generated data files a description of all extracted data items is created on standard output.*

Table 2.4 pCOS command-line options for advanced output retrieval

option	*parameters*	*function*
--format ***-f***	<*string*>	*(Affects only* `--extended` *and* `--pcospath`*) Output format for recursion level 0. Expressions within (...) will iterate over all existing keys. Format examples can be found in Table 2.3. The following placeholders can be used in addition to regular characters:* ***IF*** *input file name* ***PP*** *pCOS path of the object* ***KEY*** *name of the object* ***VAL*** *value of the object* ***\n*** *carriage return plus linefeed on Windows; single linefeed on all other systems* ***\r*** *carriage return* ***\t*** *horizontal tab* *Default:* `PP/KEY: VAL\n` *for* `--extended`*,* `VAL\n` *for* `--pcospath` *(or* `VAL` *for binary data)*
--pcospath[1]	<*path*>...	*pCOS path of an object that will be queried. Examples for object paths can be found in Table 2.3, and a full description in the pCOS Path Reference.*

1. This option can be supplied more than once.
2. This function can be used to retrieve embedded font data from a PDF. Users are reminded that fonts are subject to the respective font vendor's license agreement, and must not be reused without the explicit permission of the respective intellectual property owners. Please contact your font vendor to discuss the relevant license agreement.

2.6 Output Options

Table 2.5 lists options for controlling details of the generated output.

Table 2.5 pCOS command-line options for controlling output details

option	parameters	function
--bracket ***-b***	<*keyword*>	*Bracketing of strings, arrays, names, dictionaries, and empty values (default: none):* ***none*** *no brackets* ***angle*** < > ***curly*** *{ }* ***round*** *( )* ***squared*** *[ ]* ***dquot*** " " ***squot*** ' '
--depth ***-d***	*1, 2, ...*	*Recursion depth for resolving dictionaries. For higher recursion levels the string supplied with* `--replace dictionary` *is printed. Default: 2*
--headline ***-h***	<*string*>	*Header line for each file. The following placeholders can be used in addition to regular characters (default: no header when a single file is processed, and* `\nIF:\n` *when multiple files are processed):* ***IF*** *input file name* ***OF*** *output file name* ***\n*** *carriage return plus linefeed on Windows; single linefeed on all other systems* ***\r*** *carriage return* ***\t*** *horizontal tab*
--help ***-?***		*Display help with a summary of available options.*
--indent	*0, 1, 2, ...*	*Indentation for hierarchical output of* `--bookmark`, `--field`, *and* `--layerdefault`. *Default: 3 (use* `--indent 0` *for creating tabular output)*
--outfile ***-o***	<*filename*>	*Output file name (ignored for* `--extract`). *The following special names are recognized (default: -*): **-** *standard output* **+** *base name of the input file with* `.pdf` *replaced with* `.txt`
--replace[1] ***-r***	<*keyword*> <*string*>	*Replacement strings. The following keywords are supported:* ***missing*** *String for non-existing objects. Default:* `<not found>` ***dictionary*** *String for unresolved dictionaries. Default:* `<dictionary>` ***control*** *Replacement of control characters (U+0000-U+001F and U+007F-U+009F). A C-style formatting expression (e.g.* `\%03o`) *is replaced with the formatted value of the character. The replacement is performed in textual and stream data. Default: no replacement*
--separator ***-s***	<*string*>	*Separator string between keys and values of type dictionary for recursion levels 1 and above. Default: =*
--targetdir ***-t***	<*dirname*>	*Output directory name; the directory must exist. Default: .*
--utf16 ***-u***		*(Ignored when writing to stdout) Convert the output to UTF-16 with BOM. Without this option the text is output in UTF-8 format, and stream contents are output without any modification.*

Table 2.5 pCOS command-line options for controlling output details

option	parameters	function	
--verbose **-v**	0, 1, 2, 3	Verbosity level (default: 1):	
		0	no output at all
		1	emit only warnings, errors, and banner
		2	like 1, but also emit file names
		3	detailed reporting

1. This option can be supplied more than once.

2.7 Unicode Output and Binary Data

Conversion rules. Subject to the PDF objects retrieved, the output created by pCOS can contain plain ASCII text (e.g. most font names), Unicode text (e.g. Japanese document info entries, or binary data (e.g. ICC profiles). pCOS creates output according to the following rules:

- Name and string objects are output in UTF-8 without BOM. This means that ASCII text will result in plain ASCII output, but Latin-1 special characters (e.g. umlauts or accented characters) will result in two-byte UTF-8 sequences. Users must be prepared for UTF-8 output, and must convert to other formats (e.g. *WinAnsi)* if required. Lines are terminated with *\r\n* (carriage return plus linefeed) on Windows, and with *\n* (single linefeed) on all other systems.
- If the *--utf16* option has been supplied and the output channel is not *stdout* the complete output is converted from UTF-8 to native UTF-16 with BOM (byte order mark). This only makes sense if all output items are UTF-8 (without any binary stream objects). pCOS emits a warning at the end of the output for some critical combinations, or if the output couldn't be converted from UTF-8 to UTF-16 (the most likely reason for this is that binary stream data was included in the output).
- Stream objects are output in binary format without any modification. This includes XMP metadata streams, but these are usually stored in the PDF as UTF-8 anyway. Be careful with the *--format* and *--replace* options since these may have undesired effects on binary data.

A Revision History

Revision history of this manual

Date	*Changes*
May 04, 2020	► *Minor refresh for PLOP/PLOP DS 5.4*
July 27, 2018	► *Repackaged the pCOS Command-line Tool in PLOP 5.3; API description removed*
August 02, 2013	► *Updates for pCOS 4.0*
October 29, 2010	► *Updates for pCOS 3.0*
July 22, 2010	► *Moved the pCOS reference for pCOS interface version 6 to a separate manual for use in multiple products*
December 07, 2009	► *Updates for pCOS interface 5 in PDFlib+PDI 8, PPS 8*
February 01, 2009	► *Updates for pCOS interface 4 in PLOP 4.0, TET 3.0, TET PDF IFilter 3.0*
October 19, 2007	► *Updates for pCOS interface 3 in pCOS 2.0*
March 28, 2006	► *Added a description of the Perl language binding*
September 30, 2005	► *Edition for pCOS interface 2 in pCOS 1.0*
June 20, 2005	► *Edition for pCOS interface 1 in TET 2.0*

PDFlib GmbH

PDFlib GmbH
Franziska-Bilek-Weg 9
80339 München, Germany
www.pdflib.com
phone +49 • 89 • 452 33 84-0

Licensing contact
sales@pdflib.com

Support
support@pdflib.com *(please include your license number)*