

pdfToolbox CLI

Manual

pdfToolbox CLI – Manual – Last modified: 19 March 2012

© 2009-2012 by callas software gmbh, Berlin, Germany
All rights reserved

All trademarks are the property of their respective owners.

Content

Getting started	6
System requirements	6
Installing the software	6
Macintosh/Windows	6
Linux/Solaris/AIX	6
Activation	6
Request an activation code	6
Activating pdfToolbox CLI	7
Time-limited trial version	7
Displaying program information	7
Program version	7
Usage information	7
Status	8
Updating from pdfToolbox CLI 3	8
pdfInspektor	8
pdfCorrect	8
pdfColorConvert	8
DeviceLink Add-on	8
pdfImpose	9
Hints and troubleshooting	10
Performance enhancement	10
Get in touch	11
Processing	12
Processing files to PDF	12
Conversion options	12
General options	13
Only process certain pages	13
Setting the cache folder	13
Incremental saving	13
PDF structure and font optimization	14
Defining an output file	14
Defining an output path	14
Define the suffix	14
Overwrite mode	14
Timestamp	14
Actions	15
Overview	15
Profiles	16
General profile options	16
Using dynamic profiles	16
Using response files	17

Provided profiles	17
Creating a report	18
Results	21
<hr/>	
Reason codes	21
Return codes	21
Errors	21
Running a profile	21
Actions	22
<hr/>	
Arrange	22
Booklet	22
N-Up	22
Fill page	23
Merge & Impose	23
Impose	24
Slice	25
Reader spreads	25
Split in half	26
Step & Repeat	26
Split PDF	26
Merge PDF	29
Present	29
Presentation	29
Handout	30
Passe partout	30
Light table	31
Document	31
Overlay	31
Create EPS	32
Create PostScript	32
Save as image	33
Extract text	34
Extract content	34
Redistill	34
Optimize PDF	35
To PDF	35
Uncertify	35
Colors	35
Process conversion	35
Extract ICC profiles	36
Layers	36
Enumerate layers	36
Import as layer	36
Split layers	37

Reports	37
Extract XMP metadata	37
Visualizer	37
Compare	39
DeviceLink Conversion	41
<hr/>	
Using DeviceLink profiles	41
Using your own profiles	41

Getting started

pdfToolbox CLI offers a wide range of options to analyze, correct and enhance PDF files as well as impositioning features and color conversion.

System requirements

The command line version of pdfToolbox is available for the following operating systems:

Windows 2000/XP/Server 2003/Vista/Server 2008/7

Mac OS X 10.5 or newer, Intel or PPC

IBM AIX version 5.3 or newer, oslevel 5.3.7.0 (call `oslevel -q` to check)

Sun Solaris SPARC version 8 or newer

Sun Solaris Intel version 10 or newer

Linux Debian 5.0 (Lenny)

- Spoken generally, pdfToolbox CLI should work with other Linux distributions as well, as long as there are system libraries installed that are compatible to gcc-v3.4 or newer. The dependent libstdc++ is delivered with the pdfToolbox CLI.

You can easily test if pdfToolbox CLI is working on your system: Just type `pdfToolbox --help` in the terminal.

- There are 64 bit versions of pdfToolbox CLI available for Windows and Linux. The 32 bit version of pdfToolbox CLI does also run on 64 bit systems if the required 32 bit compatibility packages are available.

Installing the software

Macintosh/Windows

To install the software start the pdfToolbox Server installer. The installation program will then take you through the necessary steps.

Linux/Solaris/AIX

Extract all files from the archive to a destination folder of your choice.

For automation purposes you should set the PATH variable to the path of the pdfToolbox CLI executable.

Additional information is provided in

`<pdfToolbox CLI installation directory>/ReadMe.txt`

Activation

Before callas pdfToolbox CLI can be used, the software has to be activated.

Request an activation code

Open a terminal window and change to your pdfToolbox CLI installation directory. Type:

```
pdfToolbox --keycode [-aws] <name> <company> <licenceCode>
```

Parameters

name Name of licensee (e.g. "Registered User")

company Name of company (e.g. "User's company")

licencecode Licence key obtained from the registration card
 To make a request for a trial version, please use the keyword "trial" (for a pdfToolbox trial version) or "trialaddon" (for a DeviceLink Add-on trial version) for this parameter

aws for installation on Amazon Web Services (Windows only)

The textual output of --keycode has to be send via email to the email address named in the text in order to receive an activation code from the registratin server.

Activating pdfToolbox CLI

After having received the automatical reply email to the activation request, save the attached licence file to the file system. Then use the following command:

```
pdfToolbox --activate <licence file>
```

Parameters

licence file Full path to licence file

🔗 In order to activate pdfToolbox CLI for all user accounts of one machine, save the license file next to the pdfToolbox binary instead of installing it by using the `--activate` command.

🔗 pdfToolbox CLI is searching for the license file at various folders:

- user-preferences-folder of actual user
- next to the pdfToolbox CLI binary
- cachefolder (if set)
- user-preferences-folder for all users (shared)

When using UNIX-based-systems the environment variable

`CALLAS_SYSTEM_PREFERENCES` the path of the standard `/usr/share/callas software/callas pdfToolbox CLI` can be changed:

```
CALLAS_SYSTEM_PREFERENCES=tmp
```

would result in the searchpath: `/tmp/callas software/callas pdfToolbox CLI`

It is highly recommended to use the option `--cachefolder` instead.

Time-limited trial version

After requesting and entering a trial activation code, pdfToolbox CLI can be tested without any restrictions. When the evaluation period has expired, processing PDF files will no longer be possible until you request and enter a new activation code.

Displaying program information

Program version

```
pdfToolbox --version
```

will display the currently used version of pdfToolbox CLI.

Usage information

```
pdfToolbox --help
```

will give you a complete overview about all available commands for processing.

```
pdfToolbox --help <command>
```

will give you an overview about all available options for the command.

Status

```
pdfToolbox --status
```

will inform you about the current license state as well as the possible return and reason codes (see "Results").

Updating from pdfToolbox CLI 3

You can easily rebuild your pdfToolbox 3 workflow with pdfToolbox CLI. Mainly this can be achieved by setting up profiles with the Desktop version of pdfToolbox (see "callas pdfEngine Reference" for details). In addition there are some predefined actions that will help you perform your former tasks.

pdfInspektor

For preflighting only, define a pdfToolbox profile containing only checks or use the option `--analyse` to suppress execution of any corrections contained in a profile.

pdfToolbox also offers the possibility to combine checking and modifying by defining fixups together with checks. pdfInspektor profiles can be read by pdfToolbox CLI.

pdfCorrect

For modifying only, define a pdfToolbox profile containing only fixups. pdfToolbox also offers the possibility to combine checking and modifying by defining fixups together with checks.

- 🔴 pdfCorrect profiles are not compatible with pdfToolbox and have to be redefined.

pdfColorConvert

For converting colors you can choose from two possibilities:

- 1) Define a pdfToolbox profile with a fixup performing color conversion (preferred method)
- 2) Run pdfToolbox CLI with the action parameter `--convertcolors` (see "Actions").

For tone value adjustment see "Adjusting tone values" in the callas pdfToolbox Reference.

DeviceLink Add-on

DeviceLink conversions can be performed with the fixup "Convert colors using DeviceLink profiles" exported as kfx file from pdfToolbox Plug-In. For more details see "DeviceLink Conversion".

pdfImpose

pdfToolbox offers various ways to impose a PDF file. For a detailed explanation see "Actions".

Hints and troubleshooting

On Windows, you can prevent your workflow from stopping in case of a pdfToolbox CLI crash by setting the following registry entry:

```
HKEY_LOCAL_MACHINE\  
SYSTEM\  
CurrentControlSet\  
Control\  
Windows\  
ErrorMode
```

If ErrorMode is set to "2", crash dialogs will be suppressed. For further details, see: <http://support.microsoft.com/kb/128642/en-us?fr=1>

Performance enhancement

If you want to enhance the performance of your pdfToolbox CLI processes, please keep in mind the following rules:

- For analysis, you can limit processing to a certain page range (see "Only process certain pages").
- Rather remove fixups from a profile only intended for analysis than using `--analyze` (e.g. when using `--analyse`, initialization of ICC profiles for color conversion fixups still takes place).
- Fixups containing an "Apply to" option need more processing time if this option is set to something else but "None", since an analysis of the file contents is required before the fixup can be executed.
- If you are using any font embedding fixups, your system font folder will be scanned unless defined otherwise in the fixup configuration. A font cache will be created to improve the performance time, but still it might be useful to remove fonts that are not needed from this directory.
- Keep in mind that the option `--uncompressing` (see "Analyze image data") will uncompress images and analyze every single pixel, which may take a long time for some files.
- Creation of XML or PDF reports takes less time than the XSLT option (see "Report types").
- Creation of reports takes additional time – even if a profile contains only fixups, an analysis will be executed for gathering report information.

Get in touch

If some necessary information is not provided by this manual or if there are any questions or feedback please contact the product management by using the "Contact Support" form on www.callassoftware.com.

You can also send an e-mail to support@callassoftware.com.

When filing a bug report, please include the following information:

- operating system
- pdfToolbox version (call `pdfToolbox --version`)
- command line call
- original PDF (please delete unnecessary pages to avoid long file transfers), used profiles or configuration files
- converted PDF (if available)
- error string/code or complete output of CLI call

You can also visit the support section on www.callassoftware.com to get answers to common questions or find a reseller near you. The latter might be useful if you want to send a support request that is neither in English nor German.


Processing

Optional parameters are marked with [].

Run a profile:

```
pdfToolbox [--hitsperpage=hitsperpage]
[--hitsperdoc=hitsperdoc] [--setvariable=setvariable] [-r=r]
[-l=l] [-p=p] [--cachefolder=cachefolder] [-o=o] [-f=f] [-s=s]
[--incremental] [--analyze] [--noprogess] [--nosummary]
[--nohits] [--uncompressimg] [-w] [-t] <profile> <input file>
<input file> [...] ]
```


Run an action:

- 
pdfToolbox <action> [--cachefolder=cachefolder] [-o=o] [-f=f] [-s=s]
 [--incremental] [-w] [-t] {action specific parameters} <input file> [<input file> [...]]
 On Unix systems, if the environment variable TMPDIR is defined, its value is used instead of the default /tmp directory for storing temporary files.

Processing files to PDF

pdfToolbox CLI is able to convert common file formats directly to PDF. For more information have a look at:

<http://www.callassoftware.com/callas/doku.php/en:support:faqs:topdf>

- 
 Office file conversion is currently not supported on Solaris and AIX systems.

Conversion options

The following switches do only apply for Office files.

OpenOffice

```
--topdf_forceopenoffice
```

When defined, Microsoft Office files are processed with OpenOffice.

Start page

```
--topdf_startpage
```

Defines the first page in the given Office document which should be converted to PDF. This is set to 1 by default.

End page

```
--topdf_endpage
```

Defines the last page in the given Office document which should be converted to PDF. This is set to the last page by default.

Create PDF for screen

```
--topdf_screen
```

The images of the created PDF file have a lower quality, the resulting file size is smaller.

General options

Run a profile:

```
pdfToolbox [--hitsperpage=hitsperpage]
[--hitsperdoc=hitsperdoc] [--setvariable=setvariable] [-r=r]
[-l=l] [-p=p] [--cachefolder=cachefolder] [-o=o] [-f=f] [-s=s]
[--incremental] [--analyze] [--noprogess] [--nosummary]
[--nohits] [--uncompressimg] [-w] [-t] <profile> <input file>
[<input file> [...] ]
```

Run an action:

```
pdfToolbox <action> [--cachefolder=cachefolder] [-o=o] [-f=f]
[-s=s] [--incremental] [-w] [-t] {action specific parameters}
<input file> [<input file> [...] ]
```

Only process certain pages

```
-p --pagerange=<firstpage>[-<lastpage>]
```

Allows to define a pagerange to process when performing the following tasks:

- Running a profile that contains only checks
 - Running the action `--createeps`
 - Running the action `--saveasimg`
- 🔑 Running a profile with the option `--analyze` also honors this option.

Parameters

first page first page to be processed
last page last page to be processed

Setting the cache folder

```
--cachefolder=<path>
```

Sets the cache folder path. This is set by default to:

Windows: *C:\Documents and Settings\<user>\Application data\callas software\callas pdfToolbox CLI*

Macintosh: */Users/<user>/Library/Preferences/callas software/callas pdfToolbox CLI*

Unix: *<home directory as defined in /etc/passwd>/callas software/callas pdfToolbox CLI*

- 🔑 This option is mandatory when running the CLI as a user without a home directory.

Parameters

path absolute path to custom cache folder

Incremental saving

```
--incremental
```

Allows to modify the input file, only writing the changes to the original PDF. This can increase the speed significantly since pdfToolbox CLI does not need to create a new copy of the file.

- 🔑 When using the action `--impose` together with option `--preprocessing-profile` or the action `--mergeimpose`, the incremental saving option can be used in conjunction with `--outputfile` or `--outputfolder` to speed

up the overall processing time, because all file modifications during these multi-step processes are then performed on a single temporary PDF file.

PDF structure and font optimization

```
--nooptimization
```

The internal PDF structure and fonts are not optimized when saving the PDF file.

Defining an output file

```
-o --outputfile=<path>
```

Defines the absolute path of the destination file. The parent folder must exist.

- 🔗 Consult section "Results" to see if a new file was created. When running a profile containing checks only, no new output file is created.

Parameters

path absolute path to output file

Defining an output path

```
-f --outputfolder=<path>
```

Defines an absolute path to a folder where pdfToolbox CLI stores the resulting files of an execution.

- 🔗 If neither an output path nor an output folder is defined any result will be created next to the input file (filename will be indexed if necessary).
- 🔗 The use of `--outputfile` together with `--outputfolder` is not supported within one CLI call.

Parameters

path absolute path to output folder

Define the suffix

```
-s --suffix=<suffix>
```

Defines the suffix that will be appended to the resulting file(s) filename. The defined suffix is added before the files type suffix (e.g. Output.pdf will become Output_PDFA.pdf when using `--suffix=_PDFA`).

Parameters

suffix string to append to filename

Overwrite mode

```
-w --overwrite
```

Overwrites existing files instead of indexing the filename.

Timestamp

```
-t --timestamp
```

Every line in the Standard output (stdout) is prefixed using a time stamp.

Actions

```
pdfToolbox <action> [--cachefolder=cachefolder] [-o=o] [-f=f]
[-s=s] [--incremental] [-w] [-t] {parameters} <input file>
[<input file> [...] ]
```

For further information see chapter "Actions".

Overview

--handout	Creates a handout from a PDF presentation
--passepartout	Creates a passe partout
--lighttable	Positions pages on a virtual light table
--overlay	Places the chosen content on top of the input PDF
--createeps	Converts the PDF into EPS
--createps	Converts the PDF into PostScript
--saveasimg	Renders an image per page preserving the page aspect ratio
--extracttext	Extract text from PDF
--extractcontent	Extract content from PDF
--redistill	Recreates the PDF via PostScript, prepares for use with older equipment (RIPs)
--convertcolors	Performs a color conversion as defined in the configuration files
--presentation	Prepares for presentation in Acrobat
--mergepdf	Merges PDF files
--splitpdf	Splits multipage documents into smaller packages
--steprepeat	Imposes by Step and Repeat
--splithalf	Creates single pages from spread
--readerspreads	Combines two pages to one spread
--mergeimpose	Merges PDF files and imposes merged PDF based on rules defined in run list and sheet setup
--impose	Imposes the PDF based on rules defined in run list and sheet setup
--nup	Imposes by N-Up
--booklet	Creates booklet for printing
--fillpage	Imposes by filling up a page
--splitlayers	Splits layers/layer views into single PDFs with just the content of the layers/layer views
--slice	Slices in two files by object type
--importaslayer	Imports a PDF file and puts the content on a layer
--extracticcprofiles	Extracts ICC profiles
--enumeratelayers	Creates layers with respect to names of fonts, spot colors or ICC profiles
--extractxmpmetadata	Extracts XMP Metadata from the PDF into an XML file
--visualizer	Creates a visualizer report
--compare	Compares two PDF documents and creates a report.
--topdf	Converts supported non-PDF files to PDF
--optimizepdf	Optimizes the internal structure of the PDF and saves for Fast Web View.

`--uncertify` Removes a Preflight certificate if present

Profiles

```
pdfToolbox [--hitsperpage=hitsperpage]
[--hitsperdoc=hitsperdoc] [--setvariable=setvariable] [-r=r]
[-l=l] [-p=p] [--cachefolder=cachefolder] [-o=o] [-f=f] [-s=s]
[--incremental] [--analyze] [--noprogess] [--nosummary]
[--nohits] [--uncompressimg] [-w] [-t] <profile> <input file>
<input file> [...] ]
```

General profile options

Disable fixups

`--analyze`

Disable execution of fixups defined in the used profile. Only defined checks are carried out.

Display

`--noprogess`

Do not show progress information in Standard output (stdout) during processing.

`--nohits`

Do not show detailed hit information in Standard output (stdout) during processing.

`--nosummary`

Do not show summary of hits and fixups in Standard output (stdout) at the end of processing.

Analyze image data

`--uncompressimg`

Images get uncompressed during the checking to allow a proper calculation of used colorants. This option may increase the processing time depending on the amount of images contained in the processed PDFs.

Using dynamic profiles

A pdfToolbox profile may contain variable values which can be exchanged during runtime. For more details on setting up those dynamic profiles please see section "Use of kfx Profiles" in the callas pdfEngine Reference.

`--listvariables`

Lists all variables defined in a kfx profile.

`--setvariable=<Key>:<Value>`

Set variable 'KEY' to 'VALUE' in the provided profile.

Parameters

Key identifier used in dynamic profile

Value value to be set for this key

- ⚠ If you want to use values containing spaces, you either have to put the string into quotes or escape the space character (e.g. "Pantone 300 U" or Pantone\ 300\ U).

Example

```
pdfToolbox --listvariables <profile>
```

```
pdfToolbox --setvariable=RESOLUTION:300 <profile> <PDF file>
```

🔑 The following characters need to be escaped with \:

```
' ! ? * $ [ ] \ ( ) | / ]
```

Using response files

To keep the command line call structured and straightforward, pdfToolbox CLI supports the usage of response files. These offer the possibility to define each command line switch line by line and also add some comments.

Example

Response file variables.rsp:

```
#####
# Set resolution
#
--setvariable=RESOLUTION:300
#
#####
# EOF
```

Command line call:

```
pdfToolbox @<absolute path to variables.rsp> <profile>
<PDF file>
```

Provided profiles

In order to generate, modify or view pdfToolbox profiles you need the Desktop version of pdfToolbox. pdfToolbox CLI is able to work with all profiles set up with pdfToolbox Plug-In or Standalone. Profiles delivered with pdfToolbox CLI may be edited as well. In order to use a certain profile you will have to export it as a profile package (*.kfp -file). For further details on setting up a profile see "Use of kfp Profiles" in the callas pdfEngine Reference.

pdfToolbox CLI gets shipped with a set of predefined profiles stored in logical groups within <Application folder>/var/Profiles:

Acrobat PDF version compatibility

Profiles for checking the compatibility of a file to a specified Acrobat version

Convert colors

Profiles to perform color conversions

🔑 To perform a color conversion using the DeviceLink profiles available as payable option of pdfToolbox, you have to have a valid license for the callas DeviceLink Add-on. The list of provided profiles can be found in section "DeviceLink Profiles" of "callas pdfEngine Reference".

Create PDF layers

Profiles to put specified objects to different layers

Digital printing and online publishing

Profiles to optimize PDF files for digital printing or online publishing

PDF analysis

Profiles for general analysis of the PDF and its objects (e.g. number of plates, image resolution etc.)

PDF fixups

Profiles for modifying the contents of a PDF (e.g. downsampling of images, embedding of fonts etc.)

PDF/A compliance

Profiles for verifying compliancy with and converting to PDF/A

PDF/E compliance

Profiles for verifying compliancy with and converting to PDF/E

PDF/X compliance

Profiles for verifying compliancy with and converting to PDF/X

PDF/X-ready profiles ger

Profiles of the pdfx-ready initiative. For more information see: www.pdfx-ready.org

Prepress

Profiles based on the recommendations of the Ghent PDF Workgroup that are based on PDF/X and specify further requirements for various printing conditions. For more information see: www.gwg.org

Creating a report

You have a wide variety of options for creating a report. Only adding `-r` to your call would create an XML report next to the input PDF file, no matter if any hits occurred. You can modify this behavior by the following parameters:

```
--report=<type>, <trigger>, [options, ]<PATH=path>
```

- You can use `--report` as often as you like in one run to create different type of reports.

Parameters

type	see "Report types"
trigger	see "Report triggers"
options	see "Further options"
path	see "Report path"

Report types

XML	XML report
XSLT=<type>	XSLT report, type can be a custom type or one of the types delivered with pdfToolbox CLI ("compacttext" or "compacthtml")
MASK	PDF report, problems highlighted by transparent masks

COMMENT	PDF report, problems highlighted by annotations
LAYER	PDF report, problems separated on layers
INVENTORY	PDF report which lists all resources used in the PDF file
COMPARE	PDF compare report

Report triggers

ALWAYS	Always create report (default)
ERROR	Create if at least one problem with severity "Error" was found
WARNING	Create if at least one problem with severity "Warning" was found
INFO	Create if at least one problem with severity "Info" was found
HIT	Same as INFO,WARNING,ERROR
NOHIT	Create if no problem was found

Further options

OVERVIEW Include overview for PDF reports

PDF layer report options

ICCNAMES Create layer for all ICC color space names
 SPOTNAMES Create layer for all spot color space names
 FONTNAMES Create layer for all font names

Inventory report

FONTS Include fonts
 COLORS Include colors
 SHADES Include smooth shades
 PATTERNS Include patterns
 IMAGES Include images, optional number of pixels like IMAGES_100
 FORMXOB Include Form XObjects
 XMP Include XMP
 XMPADV Include XMP advanced

Report path

PATH=<path>

path Path to report file (if not defined, report is created next to input file)
 ⓘ When defined, this must always be the last element of the `--report` parameter.

Hits per page

`--hitsperpage=<number>`

Maximum number of hits per page reported.

Parameters

number maximum number of hits per page that will be reported

Hits per document

`--hitsperdoc=<number>`

Maximum number of hits per document reported.

Parameters

number maximum number of hits per document that will be reported

Setting the report language

```
-l --language=<language>
```

Sets the desired language for report files.

Parameters

language language of report files

Supported values are:

en	English
de	German
fr	French
es	Spanish
it	Italian
pt	Brazilian Portuguese
cz	Czech
da	Danish
nl	Dutch
fi	Finnish
ja	Japanese
ko	Korean
no	Norwegian
pl	Polish
sv	Swedish

Example

```
pdfToolbox <profile> <PDF file> --language=fr  
--report=ERROR,WARNING,LAYER,OVERVIEW,PATH=<path to report file>
```

Results

Reason codes

The reason codes will be printed in the command line output if a general error occurs.

1000	Unknown reason
1001	A parameter is wrong
1002	A requested file could not be found
1003	A requested folder could not be found
1004	A requested folder is a file
1005	A requested file is a folder
1006	30 days trial period expired
1007	Time limited keycode expired
1008	Not activated (no keycode)
1009	PDF does not contain ICC profiles
1010	File could not be opened
1011	File is encrypted and could not be opened for writing
1012	File could not be saved

Return codes

All return codes below 100 indicate a successful operation.

0	Successful operation
---	----------------------

Errors

100	Not serialized (no valid serialization found or keycode expired)
101	Command line parameter error
102	Command line syntax error (illegal command)
103	Unknown error (internal error)
104	File could not be opened
105	File is encrypted and could not be opened for writing
106	File could not be saved

Running a profile

0	No hit, no fixups executed
1	At least one hit with severity 'info', no fixups executed
2	At least one hit with severity 'warning', no fixups executed
3	At least one hit with severity 'error', no fixups executed
5	No hit, fixups have been executed
6	At least one hit with severity 'info', fixups have been executed
7	At least one hit with severity 'warning', fixups have been executed
8	At least one hit with severity "error", fixups have been executed; fixups failed

🔊 Other codes (e.g. 137-139) are indicating an error (crash) of the environmental system. Please report such cases together with details and files to support@callassoftware.com.

Actions

Predefined actions ease the use of often needed processes like imposition or color conversions. For more information on options to be used with all actions see "General options".

```
pdfToolbox <action> {parameters} <PDF file>
```

Further syntax information about the single actions can be achieved by calling

```
pdfToolbox --help <action>
```

The following documentation gives you a short overview of purpose and configuration of the actions. Optional parameters are marked with [].

Arrange

Booklet

```
--booklet [--voffset=0mm] [--hoffset=0mm]
[--pageheight=pageheight] [--pagewidth=pagewidth] [--cutmarks]
```

Purpose

Prepares a PDF document for double sided printing, such that the printout can be folded and saddle-stitched.

Parameters

voffset	optional, vertical offset from placement (pt, inch, mm, cm)
hoffset	optional, horizontal offset from placement (pt, inch, mm, cm)
pageheight	optional, page height of the new page (pt, inch, mm, cm)
pagewidth	optional, page width of the new page (pt, inch, mm, cm)
cutmarks	optional, place cutmarks around every imposed page

Example

```
pdfToolbox --booklet --cutmarks <PDF file>
```

N-Up

```
--nup [--cutmarks] [--voffset=0mm] [--hoffset=0mm]
[--pageheight=pageheight] [--pagewidth=pagewidth]
[--distance=distance] --htimes=<> --vtimes=<>
```

Purpose

Puts several pages onto a new page. You have to define how many pages should be placed next to each other horizontally and vertically as well as the distance between the placed pages.

Parameters

cutmarks	optional, place cutmarks around every imposed page
voffset	optional, vertical offset from center (pt, inch, mm, cm)

hoffset	optional, horizontal offset from center (pt, inch, mm, cm)
pageheight	optional, height of the page where the single pages are placed on (pt, inch, mm, cm)
pagewidth	optional, width of the page where the single pages are placed on (pt, inch, mm, cm)
distance	optional, distance between placed pages (pt, inch, mm, cm)
htimes	number of pages to be placed next to each other horizontally
vtimes	number of pages to be placed next to each other vertically

Example

```
pdfToolbox --nup --htimes=3 --vtimes=2 --distance=10mm
<PDF file>
```

Fill page

```
--fillpage [--cutmarks] --distance=distance
--pageheight=pageheight --pagewidth=pagewidth
```

Purpose

Puts several pages onto a new sheet with a defined page size. Distributes pages across/down as space permits.

Parameters

cutmarks	optional, place cutmarks around every imposed page
distance	distance between placed pages (pt, inch, mm, cm)
pageheight	height of the page where the single pages are placed on (pt, inch, mm, cm)
pagewidth	width of the page where the single pages are placed on (pt, inch, mm, cm)

Example

```
pdfToolbox --fillpage --distance=10mm --pageheight=420mm
--pagewidth=594mm <PDF file>
```

Merge & Impose

```
--mergeimpose <runlist> <sheet config>
```

Purpose

Merges PDF files and imposes merged PDF based on rules defined in run list and sheet setup. For more information see "Use of Imposition cfgs" in the callas pdfEngine Reference.

- 🔧 This is the same as running the actions `--mergepdf` and `--impose` in sequence.
- 🔧 To speed up this process, consider using the `--incremental` parameter.

Parameters

runlist	imposition run list folder or file; file extension has to be ".runlist"
sheet config	sheet configuration files; file extension has to be ".sheetconfig"

- 🔗 Pre-installed imposition configurations can be found in `<Application folder>/var/Actions/Impose`

Example

```
pdfToolbox --mergeimpose <runlist>
<sheet config> <PDF file> <PDF file>
```

Impose

```
--impose [--preprocessingprofile=preprocessingprofile]
[--setvariable=setvariable] <runlist folder> <sheet config
folder>
```

Purpose

Imposes the PDF based on rules defined in run list and sheet setup. For more information see "Use of Imposition cfgs" in the callas pdfEngine Reference.

- 🔗 To speed up this process, consider using the `--incremental` parameter.

Parameters

preprocessingprofile	optional, path to a kfx profile that is executed as a preprocessing step; the used profile can not contain variables
setvariable	optional, set imposition runlist environment variable (for examples see "Use of dynamic profiles")
runlist	imposition run list configuration files; file extension has to be ".runlist"
sheet config	sheet configuration files; file extension has to be ".sheetconfig"

- 🔗 Pre-installed imposition configurations can be found in `<Application folder>/var/Actions/Impose`

Example

```
pdfToolbox --impose --preprocessingprofile=<profile>
<runlist> <sheet config> <PDF file>
```

Listing all available imposition configurations

```
--list [--language=en] [--runlists] [--sheetconfigs]
```

Lists all imposition configuration files which are stored in `<Application folder>/var/Actions/Impose`.

Parameters

language	optional, language used for listing, supported values are: en English de German fr French
runlists	optional, this will list all available runlist

configuration files
 sheetconfigs optional, this will list all available sheet configuration files

- The usage of both `--runlists` and `--sheetconfigs` equals the usage of `--list` without any additional parameters.

Example

```
pdfToolbox --list --runlists --language=fr
```

Slice

```
--slice <profile>
```

Purpose

Extracts objects from the current document defined by a preflight check and saves two files as a result (one containing the chosen objects, one containing the remaining objects).

Parameters

profile pdfToolbox profile containing a single check that defines the objects to be sliced from the current document (e.g. color images with a resolution below 150dpi), pre-configured profiles can be found in `<Application folder>/var/Actions/Slice`

Example

```
pdfToolbox --slice <profile> <PDF file>
```

Reader spreads

```
--readerspreads [--voffset=0mm] [--hoffset=0mm]
[--pageheight=pageheight] [--pagewidth=pagewidth]
```

Purpose

Imposes a PDF document by placing two contiguous pages next to each other.

Parameters

voffset optional, vertical offset from center (pt, inch, mm, cm)
 hoffset optional, horizontal offset from center (pt, inch, mm, cm)
 pageheight optional, page height of the new page (pt, inch, mm, cm)
 pagewidth optional, page width of the new page (pt, inch, mm, cm)

Example

```
pdfToolbox --readerspreads <PDF file>
```

Split in half

```
--splithalf
```

Purpose

Splits double pages into single pages. Recognizes if a document contains single pages as well as double pages are, and then only splits the double pages, leaving the single pages as they are.

Example

```
pdfToolbox --splithalf <PDF file>
```

Step & Repeat

```
--steprepeat [--cutmarks] [--voffset=0mm] [--hoffset=0mm]
[--pageheight=pageheight] [--pagewidth=pagewidth]
[--distance=distance] --htimes=<> --vtimes=<>
```

Purpose

Imposes a PDF by placing a page multiple times onto a newly created page.

Parameters

cutmarks	optional, place cutmarks around every imposed page
voffset	optional, vertical offset from center (pt, inch, mm, cm)
hoffset	optional, horizontal offset from center (pt, inch, mm, cm)
pageheight	optional, page height of the new page (pt, inch, mm, cm)
pagewidth	optional, page width of the new page (pt, inch, mm, cm)
distance	distance between placed pages (pt, inch, mm, cm)
htimes	number of pages to be placed next to each other horizontally
vtimes	number of pages to be placed next to each other vertically

Example

```
pdfToolbox --steprepeat --htimes=3 --vtimes=2 --distance=10mm
<PDF file>
```

Split PDF

```
--splitpdf [--digits=4] [--splitscheme=splitscheme]
```

Purpose

Splits multipage documents into smaller packages.

Parameters

digits	optional, defines the number of digits for page number (Default = 4)
splitscheme	optional, custom split scheme (see "Split scheme")

Naming of output files

If output option defines a folder, packages are always named as

```
<document_name>_<suffix with 4 digits that has the number of
the first page in file>
```

If output option defines a file name, the first package is named according to this file name. Further packages are created at the same place as the first package using a name as described above for folders.

Simple tokens may also be used in order to define the output:

<docname>	Defines the name of the original document
<firstpage>	Defines the first page number
<lastpage>	Defines the first page number
<firstpage><lastpage>	Use 4 digits if not changed using <code>--digits</code>
<firstpagelabel>	Evaluates page label of first page of splitted file
<lastpagelabel>	Evaluates page label of last page of splitted file

For more possible tokens please see "Token Engine" in the "callas pdfToolbox Reference".

Split Scheme

```
--splitscheme=<expression>
```

Expression may be a number with an asterisk "*" or a more complex string. If it is a number with an asterisk "*" it creates PDF files with the defined number of pages. E.g. if the number is 3* it would create 3 packages with 3 pages and one package with one page from a 10 page file.

For possible expressions, please see table "Expressions".

General Syntax

Start Page	(number)
Digit	0 1 2 3 4 5 6 7 8 9
Unsigned	digit {digit}.2
Number	[+ -] unsigned

Joker

```
<expression>,$
```

Can be combined with other expressions (has to be the last item in a list) in order to save all pages that are not part of any other expression into a separate PDF.

Example

```
1-5,8,-1-3,$
```

would create 4 PDFs with page 1-5, page 8, the last 3 pages and the rest of the pages of an input PDF.

Expressions

Type	Syntax	Example	
Simple expression	number [-number]	1-5	Page 1 to 5: [1,2,3,4,5]
		5-1	Page 5 to 1: [5,4,3,2,1]
		8	Only page 8
		-1	Last page
		-3--1	Last 3 pages: [n, n-1, n-2]
		-1--3	Last 3 pages in reverse order: [n-2, n-1, n]
		-1-3	Last n - 2 pages in reverse order: [n, n-1,...,3]
	*2 (2)	[2][4][6]...	
Multipage expression	even_pages even	even	All even pages (same as *2(2))
	uneven_pages uneven odd	uneven	All uneven pages (same as *2(1))
	Package number* [(start_page)]	5*	Packages of 5 pages
		5* (2)	Packages of 5 pages, start- ing with page 2
	Intervall *number [(start_page)]	*5	Every 5th page
		*5 (2)	Every 5th page, starting with page 2
		*5 (-20)	Single page PDFs for every 5th page of the last 20 pages of a document (totally 4 PDFs)

Type	Syntax	Example	
Simple expression list	<pre>simple_expression { "," simple_expression } ["," joker]</pre>	1-5,8,-1-3	3 PDFs with page 1-5, page 8 and the last 3 pages of an input PDF
		5*(2)	Packages of 5 pages, starting with page 2
		*5(2)	Every 5th page, starting with page 2
		*5(-20)	Single page PDFs for every 5th page of the last 20 pages of a document (totally 4 PDFs)

Example

```
pdfToolbox --splitpdf --digits=2 --splitscheme=-3--1
<PDF file>
```

Merge PDF

```
--mergepdf {<List of PDF files> | <Folder>}
```

Merges PDF files.

PDFs are merged in the order as defined in the call. PDFs in folders are merged in alphabetical order.

- 🔊 If a folder is defined as input path, only PDF files inside this folder will be merged. Any other file formats and subfolders get ignored.
- 🔊 If no name is defined via `-o` the name of the first original PDF is used.

Example

```
pdfToolbox --mergepdf <input folder with PDF files to merge>
```

Present**Presentation**

```
--presentation [--progressthermometer] [--blackslideatend]
[--fullscreen] [--selfrunning=0] [--transition=transition]
```

Purpose

Prepares a PDF for use as a slide presentation.

Parameters

progressthermometer	optional, add a progress thermometer at the bottom of the pages of the document
blackslideatend	optional, add black slide at the end of the document
fullscreen	optional, display presentation in full screen mode
selfrunning	optional, number of seconds for each page in a selfrunning presentation
transition	optional, transition between pages (any of: blinds, box, comb, cover, dissolve, fade, glitter, push, random, replace, split, uncover, wipe, zoomin, zoomout)

Example

```
pdfToolbox --presentation --selfrunning=5 --transition=split
--progressthermometer --fullscreen <PDF file>
```

Handout

```
--handout [--pagesize=DINA4] [--firstpageonlyoneslide]
[--slidesperpage=3]
```

Purpose

Creates a handout containing several slides on one page and optionally some lines for notes.

Parameters

pagesize	optional, pagesize of resulting document (any of: Letter, DINA4)
firstpageonlyoneslide	optional, place only one slide on the first page
slidesperpage	optional, number and layout of slides on page (any of: 2, 2nonotes, 3, 3nonotes, 2x2, 2x2nonotes, 2x3nonotes, 3x2)

Example

```
pdfToolbox --handout --pagesize=Letter
--firstpageonlyoneslide --slidesperpage=2x2 <PDF file>
```

Passe partout

```
--passepartout [--backgroundborderwidth=5mm] --background=<>
```

Purpose

Adds a background border around the current page content.

Parameters

backgroundborderwidth	optional, width of background border around each page (pt, inch, mm, cm)
background	path to a pdf file used as the background pattern, pre-installed background pattern files can be found in <Application folder>/var/Actions/PassePartout

Example

```
pdfToolbox --passepartout --backgroundborderwidth=1cm
--background=<Background PDF> <PDF file>
```

Light table

```
--lighttable --background=<> [--numberofcolumns=5]
--pageheight=<> --pagewidth=<>
```

Purpose

Puts several pages on one new page to give the impression of a light table.

Parameters

background	path to a pdf file with the background pattern, pre-installed background pattern files can be found in <Application folder>/var/Actions/LightTable
numberofcolumns	optional, number of columns
pageheight	page height of the new page (pt, inch, mm, cm)
pagewidth	page width of the new page (pt, inch, mm, cm)

Example

```
pdfToolbox --lighttable --numberofcolumns=3
--background=<Background PDF> --pageheight=420mm
--pagewidth=594mm <PDF file>
```

Document**Overlay**

```
--overlay [--voffset=0] [--hoffset=0] [--placement=TopRight]
<overlay file>
```

Purpose

Places the chosen content on top of the processed PDF.

Parameters

hoffset	optional, horizontal offset from placement (pt, inch, mm, cm)
voffset	optional, vertical offset from placement (pt, inch, mm, cm)
placement	optional, placement of the pages (any of TopLeft, TopCenter, TopRight, LeftCenter, Center, RightCenter, BottomLeft, BottomCenter, BottomRight)
overlay file	full path to PDF to put on top of the input PDF, pre-installed overlay files can be found in <Application folder>/var/Actions/Overlay

Example

```
pdfToolbox --overlay --voffset=10 --hoffset=50 <overlay file>
<PDF file>
```


Create EPS

```
--createeps [--transparencyquality=100]
[--gradientresolution=360] [--bitmapresolution=1200]
[--applyoutputpreviewsettings]
[--simulationprofile='ISO Coated v2 (ECI)'] [--colormanagement]
[--marksweight=0.125] [--pageinformation] [--colorbars]
[--registrationmarks] [--cutmarks] [--simulateoverprint]
[--postscript=3]
```

Purpose

Converts all pages of the PDF into EPS. The EPS files are saved next to the input PDF file unless you use `-f` to define an output path.

Parameters

transparencyquality	optional, transparency quality in % (default: 100)
gradientresolution	optional, gradient resolution in ppi (default: 360)
bitmapresolution	optional, bitmap resolution in ppi (default: 1200)
applyoutputpreviewsettings	optional, apply output preview settings
simulationprofile	optional, simulation profile (default: 'ISO Coated v2 (ECI)')
colormanagement	 Not usable on Unix optional, apply host based color management
marksweight	optional, line weight of cut marks in pt (default: 0.125)
pageinformation	optional, add page information
colorbars	optional, add color bars
registrationmarks	optional, add registration marks
cutmarks	optional, add cutmarks
simulateoverprint	optional, simulate overprint
postscript	optional, Postscript level [2 3] (default: 3)

Example

```
pdfToolbox --createeps --postscript=2 --pageinformation
--colorbars --registrationmarks --cutmarks <PDF file>
```

Create PostScript

```
--createeps [--transparencyquality=100]
[--gradientresolution=360] [--bitmapresolution=1200]
[--applyoutputpreviewsettings]
[--simulationprofile='ISO Coated v2 (ECI)'] [--colormanagement]
[--marksweight=0.125] [--pageinformation] [--colorbars]
[--registrationmarks] [--cutmarks] [--simulateoverprint]
[--postscript=3]
```

Purpose

Converts all pages of the PDF into PostScript. The PostScript files are saved next to the input PDF file unless you use `-f` to define an output path.

Parameters

transparencyquality	optional, transparency quality in %
---------------------	-------------------------------------

gradientresolution	(default: 100) optional, gradient resolution in ppi (default: 360)
bitmapresolution	optional, bitmap resolution in ppi (default: 1200)
applyoutputpreviewsettings	optional, apply output preview settings
simulationprofile	optional, simulation profile (default: 'ISO Coated v2 (ECI)')
colormanagement	❗ Not usable on Unix optional, apply host based color management
marksweight	optional, line weight of cut marks in pt (default: 0.125)
pageinformation	optional, add page information
colorbars	optional, add color bars
registrationmarks	optional, add registration marks
cutmarks	optional, add cutmarks
simulateoverprint	optional, simulate overprint
postscript	optional, Postscript level [2 3] (default: 3)

Example

```
pdfToolbox --createeps --postscript=2 --pageinformation
--colorbars --registrationmarks --cutmarks <PDF file>
```

Save as image

```
--saveasimg [--nosimulateoverprint] [--simulationprofile=<ICC
profile>] [--smoothing=lines] [--resolution=72]
[--colorspace=colorspace] [--jpegformat=Baseline_Stan-
dard] [--compression=JPEG_medium] [--imgformat=JPEG]
[--pagebox=cropbox] [--rect=<left>,<bottom>,<right>,<top>[un
it]]
```

Purpose

Renders an image per page preserving the page's aspect ratio. RGB images always use sRGB as Destination ICC profile which gets embedded into the resulting image. CMYK and gray images are saved without an ICC profile.

Parameters

nosimulateoverprint	optional, avoids the overprint-simulation
simulationprofile	optional, using a user-defined ICC-profile for rendering
smoothing	optional, None, All, Lines, Images, Text (default: All)
resolution	optional, resolution in ppi or width x height in pixel, e.g. 1024x800 (default: 72)
colorspace	optional, one of RGB, CMYK, Gray (default: RGB)
jpegformat	optional, Baseline_Standard, Progressive_3_Scan (default: Baseline_Standard)
compression	optional, JPEG_minimum, JPEG_low, JPEG_medium, JPEG_high, JPEG_maximum

imgformat	(default: JPEG_medium) optional, JPEG, PNG, TIFF (default: JPEG)
pagebox	optional, using a geometry box as size for image: CROPBOX, TRIMBOX, BLEEDBOX (default: CROPBOX)
rect	optional, defines lower left and upper right from origin geometry box (default: CROPBOX); in pt or mm (default:pt)

Example

```
pdfToolbox --saveasimg --imgformat=PNG --resolution=800x600
<PDF file>
```

Extract text

```
--extracttext
```

Purpose

Extracts the text of PDF documents to the command line or to a specified file.

Example

```
pdfToolbox --extracttext <PDF file>
```

Extract content

```
--extractcontent [--words] [--wordbbox] [--wordquads]
[--chars] [--docxmp] [--docinfo] [--annots]
```

Purpose

Extracts the the text in the form of words or characters to an XML file.

Parameters

words	Include words
wordbbox	Include bounding box information for words
wordquads	Include quad point information for word parts
chars	Include quad point information for individual characters
docxmp	Include document XMP metadata
docinfo	Include document info
annots	Include link annotations

Example

```
pdfToolbox --extractcontent [--words] [--docinfo] <PDF file>
```

Redistill

```
--redistill [--topdf_pdfsetting] <PDF file>
```

Purpose

Recreates the PDF via PostScript, prepares for use with older equipment (RIPs).

Parameters

topdf_pdfsetting PDF setting

Example

```
pdfToolbox --redistill <PDF file>
```

Optimize PDF

```
--optimizepdf <PDF file>
```

Purpose

Optimizes the internal structure of the PDF and saves for Fast Web View.

Example

```
pdfToolbox --optimizepdf <PDF file>
```

To PDF

```
--topdf [--topdf_pdfsetting]
```

Purpose

Converts supported non-PDF files to PDF. Information about supported file types can be found here:

<http://www.callassoftware.com/callas/doku.php/en:support:faqs:topdf>

Parameters

topdf_pdfsetting PDF setting

Example

```
pdfToolbox --topdf <non-PDF file>
```

Uncertify

```
--uncertify <PDF file>
```

Purpose

Removes a Preflight certificate if present.

Example

```
pdfToolbox --uncertify <PDF file>
```

Colors**Process conversion**

```
--convertcolors [--spotcolor=spotcolor]
[--destination=destination] [--source=source]
```

Purpose

Prepares the current PDF for the chosen printing condition and carries out the necessary color conversion. For further information on setting up configuration files (*.cfg) see "Use of color conversion" in the callas pdfEngine Reference. You can either define a separate config file for each of source, spot color and destination or only use one of the switches with a single config file containing all necessary conversion parameters.

Parameters

spotcolor	full path to a cfg file defining the spot color options, pre-installed config files can be found in <Application folder>/var/Actions/ConvertColors/SpotColors
destination	full path to a cfg file defining the destination options, pre-installed config files can be found in <Application folder>/var/Actions/ConvertColors/Destination
source	full path to a cfg file defining the source options, pre-installed config files can be found in <Application folder>/var/Actions/ConvertColors/Source

Example

```
pdfToolbox --convertcolors --spotcolor=<spot config file>
--destination=<destination config file>
--source=<source config file> <PDF file>
```

Extract ICC profiles

```
--extracticcprofiles
```

Purpose

Saves all embedded ICC profiles from the document. Profiles of ICC based color spaces as well as ICC profiles used in Output Intents are extracted.

Example

```
pdfToolbox --extracticcprofiles <PDF file>
```

Layers

Enumerate layers

```
--enumeratelayers [--iccnames] [--fontnames] [--spotnames]
```

Purpose

Enumerate the chosen objects on separate layers.

Parameters

iccnames	optional, creates a layer for each ICC profile in the PDF
fontnames	optional, creates a layer for each font in the PDF
spotnames	optional, creates a layer for each spot color in the PDF

Example

```
pdfToolbox --enumeratelayers --fontnames <PDF file>
```

Import as layer

```
--importaslayer [--name="Layer 1"] <import file>
```

Purpose

Imports the chosen PDF document as a layer in the processed PDF.

If e.g. the imported PDF contains 2 page and the document it is imported to contains 5, only page 1 and 2 of the resulting document would contain a layer.

If e.g. the imported PDF contains 2 pages and the document it is imported to 1 page, then only the first page would be imported.

Parameters

name optional, name of the new layer
import file full path to a PDF to be imported

Example

```
pdfToolbox --name="My Layer" <PDF file> <PDF file>
```

Split layers

```
--splitlayers [--singlepages]
```

Creates a separate PDF file for every layer view or layer with the content visible when viewing this layer view or single layer.

- The name of the output files (if not defined via `-o`) will have the following syntax

```
originalfilename_layer(view)
```

Parameters

singlepages optional, creates a separate PDF per page of the original PDF
File names are suffixed using the page number

Example

```
pdfToolbox --splitlayers --singlepages <PDF file>
```

Reports

Extract XMP metadata

```
--extractxmpmetadata <report config file>
```

Purpose

Extract XMP Metadata of the processed PDF into a configurable XML file. For more information see "Use of XMP Metadata reports".

Parameters

report config file full path to a meta data report configuration file, pre-installed config files can be found in
<Application folder>/var/Actions/Metadata/Filters/Export

Example

```
pdfToolbox <report config file> <PDF file>
```

Visualizer

```
--visualizer [--smlobj=smlobj] [--inkcov=inkcov]
[--bmpres=bmpres] [--imgres=imgres] [--part=part]
[--format=format] [--resolution=resolution] [-p=p] [-l=l]
```

Purpose

Create a report listing print relevant aspects of a PDF document.

Parameters

smlobj	Threshold for small object coverage highlighting, see "Resolution output" (default: low)
inkcov	Threshold for ink coverage highlighting (default: 250)
bmpres	Threshold for bitmap resolution highlighting (default: 550)
imgres	Threshold for image resolution highlighting (default: 150)
part	See "Report parts"
format	See "Report type"
resolution	Resolution in ppi or width x height in pixel, e.g. 1024x800 (default: 72)
language	report language (e.g. en (English, default), de, es, fr or it)

Resolution output

Following you will find the values taken as thresholds for the chosen output resolution.

	Text	Multicolored text	Line	Multicolored line
low	8 pt	10 pt	0.5 pt	2 pt
medium	5 pt	9 pt	0.125 pt	0.25 pt
high	5 pt	8 pt	0.125 pt	0.25 pt

Report parts**PDF report**

all	all visualizer parts
ink	all ink coverage views
sep	all individual separations
imgres	all image resolution views
smallobj	all small object views
safety	all safety zone views

Image report

all	all visualizer parts
full	regular page view
ink	all ink coverage views
ink_temp	ink coverage above threshold
ink_topo	ink coverage topographic view
process	all process color views
process_CMYK	CMYK channels (without spots)
process_CMY	CMY
process_K	K channel only
spot	all spot color views
spot_spots	spot color channels
spot_spots_K	spot color + K channels
spot_CMYK	CMYK channels (without spots)

sep	all individual separations
sep_process	all individual process separations
sep_spot	all individual spot color separations
imgres	all image resolution views
imgres_img	image resolution below threshold
imgres_bmp	bitmap resolution below threshold
smallobj	all small object views
smallobj_text	very small text objects below threshold
smallobj_lines	very small vector objects below threshold
safety	all safety zone views
safety_bleed	bleed area safety zone
safety_trim	page border safety zone
safety_full	safety zone regular page view

Report type

pdfreport	Create visualizer PDF report ⚠ Please note that PDF reports always have a
resolution of 72 ppi. images	Create individual visualizer images for every report part

Example

```
pdfToolbox --visualizer --smlobj=medium --inkcov=300
--part=ink --format=pdfreport -p=1-5 <PDF file>
```

Compare

```
--compare [--sensitivity] [--format]
[--nosimulateoverprint] [--resolution=72]
[--colorspace=colorspace] [--jpegformat=Baseline_Standard]
[--compression=JPEG_medium] [--imgformat=JPEG] <PDF file 1>
<PDF file 2>
```

Purpose

Compares two PDF documents and creates a report.

Parameters

sensitivity	Difference highlighting, any of: maximum: Maximum sensitivity (default) medium: Medium sensitivity minimum: Minimum sensitivity
format	Compare report format, any of: pdfreports or images
nosimulateoverprint	optional, deactivates simulate overprinting
resolution	optional, resolution in ppi or width x height in pixel, e.g. 1024x800 (default: 72), only applicable if report format is images
colorspace	optional, one of RGB, CMYK, Gray (default: RGB)
jpegformat	optional, Baseline_Standard, Progressive_3_Scan (default: Baseline_Standard)

compression	optional, JPEG_low, JPEG_medium, JPEG_high (default: JPEG_medium)
imgformat	optional, JPEG, PNG, TIFF (default: JPEG)

Example

```
pdfToolbox --compare --sensitivity=medium <PDF file 1>  
<PDF file 2>
```

DeviceLink Conversion

DeviceLink profiles complement the usage of regular ICC profiles to avoid weaknesses mainly are the CMYK to CMYK transformation and e.g. the preservation of pure black text in a picture. A CMYK to CMYK transformation with ICC profiles is always performed via the device independent Lab color space, which leads to a complete re-separation of the data with partly unpredictable and unwanted results. This does not happen with DeviceLink profiles, which offer a direct control of color composition.

Using DeviceLink profiles

Performing a DeviceLink conversion with pdfToolbox CLI is rather simple. Just set up a new profile with pdfToolbox Desktop and set up the fixup "Convert colors using DeviceLink profiles". Choose the desired profile from the drop down list and define if the conversion should only take place for a certain type of objects or color spaces. Then add the fixup "Embed Output Intent" with the desired settings.

- No extra software installation is needed after entering the license string when purchasing the DeviceLink Add-on.

You will find more information on the provided DeviceLink profiles and their settings in the "callas pdfEngine Reference".

Using your own profiles

You can also use your own DeviceLink profiles – no DeviceLink Add-on license is required in that case. For installation use the "Import" button at the bottom of the Fixup dialog "Convert colors using DeviceLink profile" and choose the profile location from your hard disc. Additional display properties for the user interface of the Plug-In/Stand-alone can be defined in a XML-file.