

pdfToolbox CLI 4

Manual

pdfToolbox CLI 4 – Manual – Last modified: 18 November 2009

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

All trademarks are the property of their respective owners.

Content

Getting started	5
System requirements	5
Installing the software	5
Macintosh/Windows	5
Linux/Solaris	5
Registering	5
Requesting an activation code	5
Activating	6
Adjusting the preferences path	6
Limitations of the demo version	6
Displaying program information	7
Program version	7
Usage information	7
Status	7
Updating from pdfToolbox CLI 3	7
pdfInspektor	7
pdfCorrect	7
pdfColorConvert	7
DeviceLink Add-on	8
pdfImpose	8
Get in touch	9
Performance enhancement	9
Processing	10
General options	10
Only process certain pages	10
Setting the cache folder	11
Incremental saving	11
Defining an output file	11
Defining an output path	11
Define the suffix	12
Overwrite mode	12
Timestamp	12
Actions	13
Overview	13
Profiles	14
General profile options	14
Using dynamic profiles	14
Using response files	15
Provided profiles	15
Creating a report	16

Results	19
Reason codes	19
Return codes	19
Errors	19
Running a profile	19
Actions	20
Arrange	20
Booklet	20
N-Up	20
Fill page	21
Merge & Impose	22
Impose	22
Slice	23
Reader spreads	23
Split in half	24
Step & Repeat	24
Split PDF	24
Merge PDF	27
Present	27
Presentation	27
Handout	28
Passe partout	28
Light table	29
Document	29
Overlay	29
Create EPS	30
Save as image	30
Colors	31
Process conversion	31
Extract ICC profiles	31
Layers	32
Enumerate layer	32
Import as layer	32
Split layers	32
Reports	33
Extract XMP metadata	33
DeviceLink Profiles	34
Using DeviceLink profiles	34
Using your own profiles	34

Getting started

pdfToolbox CLI 4 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 4 is available for the following operating systems:

Windows 2000 SP3

Windows XP

Windows Vista

Windows 7

Max OS X version 10.4 or newer

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

Sun Solaris SPARC version 9 or newer

Linux SuSE 9.3

- Spoken generally, pdfToolbox CLI 4 should work with other Linux distributions as well, as long as there are system libraries installed that are compatible to gcc-v3.3 or newer. The dependent libstdc++ is delivered with the pdfToolbox CLI 4.

You can easily test if pdfToolbox CLI 4 is working on your system: Just type "pdfToolbox4 --help" in the terminal.

- pdfToolbox CLI 4 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 4 installer. The installation program will then take you through the necessary steps.

Linux/Solaris

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 4 executable.

Additional information is provided in

<pdfToolbox CLI 4 installation directory>/ReadMe_CLI.txt

Registering

Before callas pdfToolbox CLI 4 can be used, the software has to be activated.

Requesting an activation code

Open a terminal window and change to your pdfToolbox CLI 4 installation directory. Type:

```
pdfToolbox4 --keycode <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 4 trial version) or "trialaddon" (for a DeviceLink Add-on trial version) for this parameter

This command will generate a text on `std::out` that must be send via email to the email address mentioned in the text in order to receive an activation code.

Activating

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:

```
pdfToolbox4 --activate <licence file>
```

Parameters

licence file Full path to licence file

Adjusting the preferences path

On Linux, Sun Solaris and AIX the activation and license information is written to `/usr/share/callas software gmbh/<product name>`. If this does collide with the security settings of your system you have several options to store the Keycodes file. Following is the order in which pdfToolbox CLI 4 searches for the Keycodes file:

- 1) A product specific environment variable is checked: The variables for pdfToolbox CLI 4 are `CALLAS_PDFTOOLBOX4_LICENSE_PATH` and `CALLAS_DEVICELINK_LICENSE_PATH` which have to define an absolute path to the "KeyCodes" file. These environment variables are used only when searching for the "KeyCodes" file and are never used when a keycode file is written using the `--keycode (-k)` option.
- 2) The directory of the executable binary is searched for a "KeyCodes" file.
- 3) As alternative, the environment variable `CALLAS_SYSTEM_PREFERENCES` can be used to define an absolute path to the preferences folder. Within this folder, the directory `callas software gmbh/<product name>` is searched for the file "KeyCodes". This file is checked for a valid key.

Limitations of the demo version

After requesting and entering a trial activation code, pdfToolbox CLI 4 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

```
pdfToolbox4 --version
```

will display the currently used version of pdfToolbox CLI 4.

Usage information

```
pdfToolbox4 --help
```

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

Status

```
pdfToolbox4 --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 4. Mainly this can be achieved by setting up profiles with the Desktop version of pdfToolbox 4 (see "callas pdfToolbox 4 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 4 profile containing only checks or use the option "--nofixups" to suppress execution of any corrections contained in a profile.

pdfToolbox 4 also offers the possibility to combine checking and modifying by defining fixups together with checks. pdfInspektor profiles can be read by pdfToolbox CLI 4.

pdfCorrect

For modifying only, define a pdfToolbox 4 profile containing only fixups. pdfToolbox 4 also offers the possibility to combine checking and modifying by defining fixups together with checks.

- 🔔 pdfCorrect profiles are not compatible with pdfToolbox 4 and have to be redefined.

pdfColorConvert

For converting colors you can choose from two possibilities:

- 1) Define a pdfToolbox 4 profile with a fixup performing color conversion (preferred method)
- 2) Run pdfToolbox CLI 4 with the action parameter "--convertcolors" (see "Actions").

For tone value adjustment see "Adjusting tone values" in the callas pdfToolbox 4 Reference.

DeviceLink Add-on

DeviceLink conversions can be performed with the fixup "Convert colors using DeviceLink profiles" exported as kfx file from pdfToolbox 4 Plug-In. For more details see "DeviceLink Conversion".

pdfImpose

pdfToolbox 4 offers various ways to impose a PDF file. For a detailed explanation see "Actions".

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 sending an e-mail to support@callassoftware.com.

If you file a bug report please make sure your email contains the following information:

- operating system
- pdfToolbox 4 version (call `pdfToolbox4 --version`)
- command line call
- original PDF (please delete unnecessary pages to avoid long file transfers), used profiles or configuration files
- converted PDF (if available)

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.

Performance enhancement

If you want to enhance the performance of your pdfToolbox CLI 4 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 `--nofixups` (e.g. when using `--nofixups`, initialization of ICC profiles for color conversion fixups still takes place).
- If you are using any font embedding fixups, your system font folder will be scanned unless defined otherwise in the fixup configuration. pdfToolbox 4 will create a font cache 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 XSLT (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.

Processing

Optional parameters are marked with [].

Run a profile:

```
pdfToolbox4 [--hitsperpage=hitsperpage]
[--hitsperdoc=hitsperdoc] [--setvariable=setvariable] [-r=r]
[-l=l] [-p=p] [--cachefolder=cachefolder] [-o=o] [-f=f] [-s=s]
[--incremental] [--nofixups] [--noprogess] [--nosummary]
[--nohits] [--uncompressimg] [--fontquickembed] [-w] [-t]
<profile> <input file> [<input file> [...] ]
```

Run an action:

```
pdfToolbox4 <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.

General options

Run a profile:

```
pdfToolbox4 [--hitsperpage=hitsperpage]
[--hitsperdoc=hitsperdoc] [--setvariable=setvariable] [-r=r]
[-l=l] [-p=p] [--cachefolder=cachefolder] [-o=o] [-f=f] [-s=s]
[--incremental] [--nofixups] [--noprogess] [--nosummary]
[--nohits] [--uncompressimg] [--fontquickembed] [-w] [-t]
<profile> <input file> [<input file> [...] ]
```

Run an action:

```
pdfToolbox4 <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 "--nofixups" 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 4

Macintosh: /Users/<user>/Library/Preferences/callas software/callas pdfToolbox CLI 4

Unix: <home directory as defined in /etc/passwd>/callas software/callas pdfToolbox CLI 4

- ⚠ 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 4 does not need to create a new copy of the file.

- ⚠ When using the action "--mergeimpose" or the action "--impose" together with option "--preprocessingprofile", 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.

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 4 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

```
pdfToolbox4 <action> [--cachefolder=cachefolder] [-o=o] [-f=f]
[-s=s] [--incremental] [-w] [-t] {parameters} <input file>
[<input file> [...] ]
```

For further information see chapter "Actions".

Overview

--lighttable	Positions pages on a virtual light table
--passepartout	Creates a passe partout
--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
--handout	Creates a handout from a PDF presentation
--createeps	Converts the PDF into EPS
--overlay	Places the chosen content on top of the input PDF
--saveasimg	Renders an image per page preserving the page aspect ratio
--convertcolors	Performs a color conversion as defined in the configuration files
--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

Profiles

```
pdfToolbox4 [--hitsperpage=hitsperpage]
[--hitsperdoc=hitsperdoc] [--setvariable=setvariable] [-r=r]
[-l=l] [-p=p] [--cachefolder=cachefolder] [-o=o] [-f=f] [-s=s]
[--incremental] [--nofixups] [--noprogess] [--nosummary]
[--nohits] [--uncompressing] [--fontquickembed] [-w] [-t]
<profile> <input file> [<input file> [...]]
```

General profile options

Disable fixups

```
--nofixups
```

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

```
--uncompressing
```

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.

Quick font embedding

```
--fontquickembed
```

Missing glyphs and width mismatches are ignored while processing any profile containing the "Embed missing fonts" fixup (for further details see "callas pdfToolbox 4 Reference").

Using dynamic profiles

A pdfToolbox 4 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 pdfToolbox 4 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

```
pdfToolbox4 --listvariables <profile>
```

```
pdfToolbox4 --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 4 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:

```
pdfToolbox4 @<absolute path to variables.rsp> <profile>
<PDF file>
```

Provided profiles

In order to generate, modify or view pdfToolbox 4 profiles you need the Desktop version of pdfToolbox 4. pdfToolbox CLI 4 is able to work with all profiles set up with pdfToolbox 4 Plug-In or Standalone. Profiles delivered with pdfToolbox CLI 4 may be edited as well. In order to use a certain profile you will have to export it as a profile package (*.kfx -file). For further details on setting up a profile see "Use of kfx Profiles" in the callas pdfToolbox 4 Reference.

pdfToolbox CLI 4 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 4, 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 pdfToolbox 4 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

PDFX-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 4 ("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

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

SHORT	Without details (default)
LONG	Include all details
PAGEINFO	Detailed page information
PAGERSRC	Detailed information for page resources
DOCRSRC	Detailed information for document resources
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

```
pdfToolbox4 <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

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".

```
pdfToolbox4 <action> {parameters} <PDF file>
```

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

```
pdfToolbox4 --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 [--cutmarks] [--sheetheight=420mm]
[--sheetwidth=297mm] [--voffset=0mm] [--hoffset=0mm]
```

Purpose

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

Parameters

cutmarks	optional, place cutmarks around every imposed page
sheetheight	optional, height of the new sheet (pt, inch, mm, cm)
sheetwidth	optional, width of the new sheet (pt, inch, mm, cm)
voffset	optional, vertical offset from center (pt, inch, mm, cm)
hoffset	optional, horizontal offset from center (pt, inch, mm, cm)

Example

```
pdfToolbox4 --booklet --cutmarks <PDF file>
```

N-Up

```
--nup --htimes=<> --vtimes=<> --distance=<>pt [--cutmarks]
[--sheetheight=420mm] [--sheetwidth=297mm] [--voffset=0mm]
[--hoffset=0mm]
```

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

htimes	number of pages to be placed next to each other horizontally
vtimes	number of pages to be placed next to each other vertically
distance	distance between placed pages (pt, inch, mm, cm)

cutmarks	optional, place cutmarks around every imposed page
sheetheight	optional, height of the new sheet (pt, inch, mm, cm)
sheetwidth	optional, width of the new sheet (pt, inch, mm, cm)
voffset	optional, vertical offset from center (pt, inch, mm, cm)
hoffset	optional, horizontal offset from center (pt, inch, mm, cm)

Example

```
pdfToolbox4 --nup --htimes=3 --vtimes=2 --distance=10mm
<PDF file>
```

Fill page

```
--fillpage --distance=<>pt --pageheight=<>pt --pagewidth=<>pt
[--cutmarks]
```

Purpose

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

Parameters

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)
distance	distance between placed pages (pt, inch, mm, cm)
cutmarks	optional, place cutmarks around every imposed page

Example

```
pdfToolbox4 --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 pdfToolbox 4 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 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

```
pdfToolbox4 --mergeimpose <runlist>
<sheet config> <PDF file> <PDF file>
```

Impose

```
--impose [--preprocessingprofile=preprocessingprofile]
<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 pdfToolbox 4 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
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

```
pdfToolbox4 --impose --preprocessingprofile=<profile>
<runlist> <sheet config> <PDF file>
```

Listing all available imposition configurations

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

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

Parameters

runlists	optional, this will list all available runlist configuration files										
sheetconfigs	optional, this will list all available sheet configuration files										
language	optional, language used for listing, supported values are: <table> <tr> <td>en</td> <td>English</td> </tr> <tr> <td>de</td> <td>German</td> </tr> <tr> <td>fr</td> <td>French</td> </tr> <tr> <td>es</td> <td>Spanish</td> </tr> <tr> <td>it</td> <td>Italian</td> </tr> </table>	en	English	de	German	fr	French	es	Spanish	it	Italian
en	English										
de	German										
fr	French										
es	Spanish										
it	Italian										

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

Example

```
pdfToolbox4 --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 4 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 <code><Application folder>/var/Actions/Slice</code>
---------	--

Example

```
pdfToolbox4 --slice <profile> <PDF file>
```

Reader spreads

```
--readerspreads [--sheetheight=420mm] [--sheetwidth=297mm]
[--voffset=0mm] [--hoffset=0mm]
```

Purpose

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

Parameters

sheetheight	optional, height of the new sheet (pt, inch, mm, cm)
sheetwidth	optional, width of the new sheet (pt, inch, mm, cm)
voffset	optional, vertical offset from center (pt, inch, mm, cm)
hoffset	optional, horizontal offset from center (pt, inch, mm, cm)

Example

```
pdfToolbox4 --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

```
pdfToolbox4 --splithalf <PDF file>
```

Step & Repeat

```
--steprepeat --htimes=<> --vtimes=<> --distance=<>pt  
[--sheetheight=420mm] [--sheetwidth=297mm] [--voffset=0mm]  
[--hoffset=0mm]
```

Purpose

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

Parameters

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

Example

```
pdfToolbox4 --steprepeat --htimes=3 --vtimes=2 --distance=10mm  
<PDF file>
```

Split PDF

```
--splitpdf [--digits=4] [--splitscheme=*1(1)]
```

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 --digits

For more possible tokens please see "Token Engine" in the "callas pdfToolbox 4 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

```
pdfToolbox4 --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

```
pdfToolbox4 --mergepdf <input folder with PDF files to merge>
```

Present**Presentation**

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

Purpose

Prepares a PDF for use as a slide presentation.

Parameters

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)
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

Example

```
pdfToolbox4 --presentation --selfrunning=5 --transition=split
--progressthermometer --fullscreen <PDF file>
```

Handout

```
--handout [--pagesize=A4] [--slidesperpage=3]
[--firstpageonlyoneslide]
```

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, DIN A4)
slidesperpage	optional, number and layout of slides on page (any of: 2, 2nonotes, 3, 3nonotes, 2x2, 2x2nonotes, 2x3nonotes, 3x2)
firstpageonlyoneslide	optional, place only one slide on the first page

Example

```
pdfToolbox4 --handout --pagesize=Letter --slidesperpage=2x2
--firstpageonlyoneslide <PDF file>
```

Passpartout

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

Purpose

Adds a background border around the current page content.

Parameters

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

Example

```
pdfToolbox4 --passepartout --backgroundborderwidth=1cm
--background=<Background PDF> <PDF file>
```

Light table

```
--lighttable [--numberofcolumns=5] --background=<>
--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 <i><Application folder>/var/Actions/LightTable</i>
pageheight	page height of the new page (pt, inch, mm, cm)
pagewidth	page width of the new page (pt, inch, mm, cm)
numberofcolumns	optional, number of columns

Example

```
pdfToolbox4 --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

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

Example

```
pdfToolbox4 --overlay --voffset=10 --hoffset=50 <overlay file>
<PDF file>
```

Create EPS

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

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

Example

```
pdfToolbox4 --createeps --postscript=2 --pageinformation
--colorbars --registrationmarks --cutmarks <PDF file>
```

Save as image

```
--saveasimg [--resolution=72] [--jpegformat=Baseline_Standard]
[--compression=JPEG_medium] [--imgformat=JPEG]
[--simulateoverprint]
```

Purpose

Renders an image per page preserving the page’s aspect ratio.

Parameters

resolution	optional, resolution in ppi or width x height in pixel, e.g. 1024x800 (default: 72)
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 (default: JPEG)
simulateoverprint	optional, simulate overprint

Example

```
pdfToolbox4 --saveasimg --imgformat=PNG --resolution=800x600
<PDF file>
```

Colors**Process conversion**

```
--convertcolors --spotcolor=<> --destination=<> --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 pdfToolbox 4 Reference.

Parameters

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
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
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

Example

```
pdfToolbox4 --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

```
pdfToolbox4 --extracticcprofiles <PDF file>
```

Layers

Enumerate layer

```
--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

```
pdfToolbox4 --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

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

Example

```
pdfToolbox4 --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

```
pdfToolbox4 --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

```
pdfToolbox4 <report config file> <PDF file>
```

DeviceLink Profiles

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 4 is rather simple. Just set up a new profile with pdfToolbox Plug-In 4 or pdfToolbox Standalone 4 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 pdfToolbox 4 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.