callas software gmbh

pdfToolbox CLI

Manual

pdfToolbox CLI – Manual – Last modified: 25 June 2014

Content

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 2008/7/8
Mac OS X 10.6 or newer, Intel
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 compatibel 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 (using Windows, Linux 32bit and 64bit)

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 registration 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.

Activation using the Standalone application

Using Windows and MacOS, also the activation dialog of the Standalone Application can be used for requesting a Activation as well as using a Keycode or just for a trial version. Also the Activation itself can be done using that Interface.
All activations (for Desktop, Server as well as for the DeviceLink Addons) can be done using this dialog.

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 kfpx 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

Ensure sufficient free disk space

To ensure stable processing, it is recommended to have at least 4 times of the input file size of processed files available for intermediate file system storage (e.g. /tmp on Unix and similar on other systems).

Avoid stopping workflows on Windows

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*

Limiting the maximum memory used by pdfToolbox

Using Linux, you can limit the amount of memory used by a single process by an additional parameter:

```
--maxmemory=<max. memory in MB>
```

Processing will stop and result in an error if memory is exceeded.

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 `--uncompressimg` (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.

Optimization of needed installation space

To reduce the pace needed by the installation of pdfToolbox, it is possible to delete some subfolders of the CLI component (in subfolder /cli) if their respective functions are not needed in the individual use case.

To avoid processing errors or unexpected behaviour of pdfToolbox any modification should be done well-considered.

etc/Actions	If no Arrange action is used
etc/APDFL	If no font embedding or PDF/A conversion is used (or if font situation is clear)
etc/Backgrounds	If no layer/image mask report is used
etc/Certify	If no preflight certification is used
etc/ColorConversion	If no color conversion is used
etc/HtmlConverter	If no PDF report based on HTML template is used
etc/Inventory	If no inventory report is used
etc/MailConverter	If no e-mails are processed
etc/PDFOfficeTool	If no Office-files are processed
etc/PDFPSTool	If no PostScript-files are processed
etc/Reports	If no PDF/A-HTML Report or ZUGFeRD is used
etc/TPex	If no tagged PDF to HTML/EPUB export is used
etc/Visualizer	If no Comparison is used

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 [-r=r] [-l=l] [-p=p] [--hitsperpage=hitsperpage]
[--hitsperdoc=hitsperdoc] [--setvariable=setvariable] [-t]
[--cachefolder=cachefolder] [-o=o] [-f=f] [--analyze] [-w]
[--incremental] [--noprogress] [--nosummary] [--nohits]
[--uncompressimg] [--password] [--timeout=timeout] [-s=s]
<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

Using Linux, Office file conversion requires an OpenOffice or LibreOffice installation on the respective system.

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

Conversion options for Office files

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.

Excel-Sheets without removing white space

```
--topdf_useexcelpagelayout
```

Uses the Excel page layout for creating the PDF, white space will not be removed.

Conversion options for Postscript files

The following switches do only apply for Postscript files.

Define folders used for font search

```
--topdf_psaddfonts=<path>
```

Parameters

path	Path to addtional font folder for PS to PDF conversion which will be additionally used for font search. System font folders will also be used.

```
--topdf_psfontsonly=<path>
```

Parameters

path	Path to font folder for PS to PDF conversion, no usage of system fonts will take place.

General options

Run a profile:

```
pdfToolbox [-w] [-t] [-o=o] [-f=f] [-s=s]  [--incremental]
[-p=p] [--hitsperpage=hitsperpage] [--hitsperdoc=hitsperdoc]
[--setvariable=setvariable] [-r=r] [-l=l] [--analyze]
[--cachefolder=cachefolder] [--noprogress] [--nosummary]
[--nohits] [--uncompressimg] [--timeout=timeout] <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 perfoming 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

Empty the profile cache

```
--emptyprofilecache [--cachefolder=<path>]
```

For performance reasons, bigger profiles are unpacked during first usage and containing ICC-profiles and config files are stored in a local profile cache. This command deletes this profile cache.

Empty the font cache

```
--emptyfontcache [--cachefolder=<path>]
```

For performance reasons, fonts found on the respective system are catalogized in an internal font cache. This command deletes this font cache.

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.PDF structure and font optimization

Enable processing PDF with password protection for editing and printing

```
--password=<password>
```

To enable profile-processing of an password-protected PDF. Only PDFs with restrictions for editing and printing can be unsecured. The resulting PDF will have no security setting.

The entered password will be visible and may be grabbed or logged by other processes on the machine.

Parameters

password — password set to avoid editing or printing of the PDF

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.

Font folders

If a font is not embedded and an embedding is required by a profile, pdfToolbox CLI will search the system font directories in order to find the needed font file, which are:

Windows	• C:\Windows\Fonts
Macintosh	• /Users/<user>/Library/Fonts • /Library/Fonts • /System/Library/Fonts

Linux, Solaris Sparc, Solaris x86, AIX
- /usr/lib/X11/fonts
- /usr/local/X11R6/lib/X11/fonts
- /usr/share/fonts
- /<user home>/.fonts

Additionally the font folder installed together with pdfToolbox CLI will be searched. This folder lies next to the executable in *"<callas pdfToolbox CLI directory>|etc|APDFL|Resource|Font"*.

ICC-profiles folders

The following folders are searched for required ICC-profiles, unless they are already contained in the .kfpx-profile already.
These folders lies next to the executable in:

- *"<callas pdfToolbox CLI directory>|etc|ICC profiles"*
- *"<callas pdfToolbox CLI directory>|etc|APDFL|Resource|Color|Profiles"*

Some system folders for colors are searched addtionally:

MacOS:	\Library\Application Support\Adobe\Color
Windows:	\Windows\system32\spool\drivers\color

Set a processing timeout

```
--timeout=<seconds>
```

Sets the maximum processing time in seconds. If the process exceeds this duration, the execution process will be killed and the processing will result in an error.

Actions

```
pdfToolbox <action> [--cachefolder=cachefolder] [-o=o] [-f=f]
[-s=s] [--incremental] [-w] [-t] [--timeout=timeout]
{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
--duplicatepage	Duplicate pages of the PDF
--splitpdf	Splits multipage documents into smaller

	packages
--splitmergepdf	Splits multipage documents into smaller packages and merges them to one document
--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
--quickpdfinfo	Lists basic information about the specific PDF.
--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 [-r=r] [-l=l] [-p=p] [-o=o] [-f=f] [-w] [-t]
[--hitsperpage=hitsperpage] [--hitsperdoc=hitsperdoc]
[--setvariable=setvariable] [--cachefolder=cachefolder] [-s=s]
[--incremental] [--analyze] [--noprogress] [--nosummary]
[--nohits] [--uncompressimg] [--certify] [--timeout=timeout]
<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

```
--noprogress
```

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.

Certify

```
--certify
```

Embed a Preflight certificate (also known as "Audit Trail") after processing.

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 kfpx Profiles" in the callas pdfEngine Reference.

```
--listvariables
```

Lists all variables defined in a kfpx 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
```

```
--setvariable=PROFILENAME:Prepress profile v7.0
#
###################
# EOF
```

If strings are used in the response file, they shall not be escaped (--setvariable=PROFILENAME:"Prepress profile v7.0").
Using different responsefiles enables the easy definition of own, localized sets of strings for names of Profiles, Fixups and Checks as well as different settings for processing PDFs for different output environments.

Command line call:

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

Provided profiles

In order to generate, modify or view pdToolbox 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 (*.kfpx -file). For further details on setting up a profile see "Use of kfpx 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.)

PDFA compliance

Profiles for verifying compliancy with and converting to PDF/A

PDFE compliance

Profiles for verifying compliancy with and converting to PDF/E

PDFX 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

ProcessPlans

Processplans as examples how to use the dynamically controlled combination of Profiles, Fixups, Checks and Actions.

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
SUMMARY	PDF overview report
TEMPLATE=	PDF report based on HTML templates

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
DOWNSAMPLING[#<PPI>]	Downsample images in PDF reports of type Mask, Layer and Comment to the given resolution (Default: 150 ppi)

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

PDF inventory report options

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

XML report options

ALL	Include all ressources (Default)
NONE	Include no ressources
IMAGES[#<NUM>]	Include first <NUM> images
FONTS[#<NUM>]	Include first <NUM> fonts
PAGES[#<NUM>]	Include first <NUM> pages
COLORS[#<NUM>]	Include first <NUM> color spaces
SHADES[#<NUM>]	Include first <NUM> smooth shades
PATTERNS[#<NUM>]	Include first <NUM> patterns
FORMXOB[#<NUM>]	Include first <NUM> Form XObjects

Ink coverage options for XML reports:

INKCOX	Include ink coverage for every page
INKCOVRES[#<PPI>]	Rendering resolution used for ink coverage calcu lation (Default: 300 ppi)
INKCOVBOX[#<Box>]	Defines the PageBox (as ArtBox, TrimBox, BleedBox, CropBox or MediaBox) which will be used for rendering (Default: CropBox)

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.

PDF report based on HTML template

```
TEMPLATE=<path>
```

path — Path to template folder (or respective index.html directly)

PDF overview reports are created based on a style defined in a HTML/CSS template. A predefined template can be found in the Server/CLI path: ../var/Reports/Templates.

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 Portoguese
cz	Czech
da	Danish
nl	Dutch
fi	Finnish
ja	Japanese
ko	Korean
no	Norwegian
pl	Polish
sv	Swedish

Example

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

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

Errors for distributed processing

110	Action is not distributable
111	No dispatcher is found
112	No satellite was found or is ready for execution
130	Execution is cancelled after timeout

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 enviromental 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-ing 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]
[--border=0mm] [--bleed=0mm]
```

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, in, mm, cm)
hoffset	optional, horizontal offset from placement (pt, in, mm, cm)
pageheight	optional, page height of the new page (pt, in, mm, cm)
pagewidth	optional, page width of the new page (pt, in, mm, cm)
cutmarks	optional, place cutmarks around every imposed page
bleed	optional, width of bleed added at outward edges of slots (pt, in, mm, cm)
border	optional, width of border added at outward edges of slots (pt, in, mm, cm)

Example

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

N-Up

```
--nup [--cutmarks] [--voffset=0mm] [--hoffset=0mm]
[--pageheight=pageheight] [--pagewidth=pagewidth]
[--border=0mm] [--bleed=0mm] [--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, in, mm, cm)
hoffset	optional, horizontal offset from center (pt, in, mm, cm)
pageheight	optional, height of the page where the single pages are placed on (pt, in, mm, cm)
pagewidth	optional, width of the page where the single pages are placed on (pt, in, mm, cm)
distance	optional, distance between placed pages (pt, in, 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
bleed	optional, width of bleed added at outward edges of slots (pt, in, mm, cm)
border	optional, width of border added at outward edges of slots (pt, in, mm, cm)

Example

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

Fill page

```
--fillpage [--cutmarks] [--border=0mm] [--bleed=0mm]
--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, in, mm, cm)
pageheight	height of the page where the single pages are placed on (pt, in, mm, cm)
pagewidth	width of the page where the single pages are placed on (pt, in, mm, cm)
bleed	optional, width of bleed added at outward edges of slots (pt, in, mm, cm)
border	optional, width of border added at outward edges of slots (pt, in, 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] [--sheettemplate=<path to PDF>]
<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 kfpx 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"
sheettemplate	Places pages from the chosen PDF underneath of the imposed pages as a background.

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
```

Listing all fonts available for usage in an imposition runlist

```
--listfonts
```

Lists all font names that can be used for the "Set TextFont" command in an imposition runlist.

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

Purpose

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

Parameters

voffset	optional, vertical offset from center (pt, in, mm, cm)
hoffset	optional, horizontal offset from center (pt, in, mm, cm)
pageheight	optional, page height of the new page (pt, in, mm, cm)
pagewidth	optional, page width of the new page (pt, in, mm, cm)
nocoverpage	optional, diables front cover handling

Example

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

Split in half

```
--splithalf [--setclipping]
```

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.

Parameters

setclipping	optional, sets clipping path to page dimension

Example

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

Step & Repeat

```
--steprepeat [--cutmarks] [--voffset=0mm] [--hoffset=0mm]
[--pageheight=pageheight] [--pagewidth=pagewidth]
[--border=0mm] [--bleed=0mm] [--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, in, mm, cm)
hoffset	optional, horizontal offset from center (pt, in, mm, cm)
pageheight	optional, page height of the new page (pt, in, mm, cm)
pagewidth	optional, page width of the new page (pt, in, mm, cm)
distance	distance between placed pages (pt, in, 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
bleed	optional, width of bleed added at outward edges of slots (pt, in, mm, cm)
border	optional, width of border added at outward edges of slots (pt, in, mm, cm)

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 `--digits`
<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".

Parameters

expression	can be a single value or a combination of values like the examples below for "Simple expressions" "Multipage expressions", "Simple expressions

list" and the "Joker"

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,-3--1,$
```

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]...

Type	Syntax	Example	
Simple expression with Simple Range	`number[-number]_ number[-number]`	`1-2_-2--1`	First and last 2 pages: [1,2,n-1,n]
		`1-2_-2--1,$`	Split PDF into 2 documents: First and last 2 pages [1,2,n-1,n] and remaining inner pages [3, ... ,n-2]
		`1_1_1_1`	4 times page 1: [1,1,1,1]
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, starting 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	`simple_expression {"," simple_expression} [ "," joker ]`	`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>
```

Split and merge PDF

```
--splitmergepdf [--splitscheme=splitscheme]
```

Purpose

Splits multipage documents into smaller packages and merges them to one document.

Parameters

splitscheme — optional, custom split scheme (see *"Split scheme"*)

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".

Parameters

expression	can be a single value or a combination of values like the examples below for "Simple expressions" "Multipage expressions", "Simple expressions list" and the "Joker"

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 plave in the PDF.

Example

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

would create 1 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]...
Simple expression with Simple Range	`number[-number]_ number[-number]`	`1-2_-2--1`	First and last 2 pages: [1,2,n-1,n]
		`1-2_-2--1,$`	First and last 2 pages [1,2,n-1,n] and remaining inner pages [3, ... ,n-2]
		`1_1_1_1`	4 times page 1: [1,1,1,1]

Type	Syntax	Example	
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, starting with page 2

	Intervall
	`*number`
	`[(start_page)]`
		`*5`	Every 5th page

		`*5(2)`	Every 5th page, starting with page 2
		`*5(-20)`	Every 5th page of the last 20 pages of a document (totally 4 pages)

Simple expression list	`simple_expression`
	`{","`
	`simple_expression}`
	`[ "," joker ]`
		`1-5,8,-1-3`	1 PDFs with page 1-5, page 8 and the last 3 pages of the input PDF

		`5*(2)`	Packages of 5 pages, starting with page 2
		`*5(2)`	Every 5th page, starting with page 2
		`*5(-20)`	Every 5th page of the last 20 pages of a document (totally 4 pages)

Example

```
pdfToolbox --splitmergepdf --splitscheme=-3--1 <PDF file>
```

Duplicate page

```
--duplicatepage [--times=1] [--pageorder] [--pagerange]
```

Duplicate pages of the PDF.

Parameters

times	optional, defines the number of copies created (default = 1)
pageorder	optional, respects the order of pages and add duplicates as a complete set (only available if times=1)
pagerange	optional, defines the rage of page to be duplicated, can not be combined with --pageorder

Example

```
pdfToolbox --duplicatepage --times=1 --pageorder <PDF file>
```

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, in, 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, in, mm, cm)
pagewidth	page width of the new page (pt, in, 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]
```

```
[--placebelow[=1|2]] <overlay file>
```

Purpose

Places the chosen content on top of the processed PDF.

Parameters

hoffset	optional, horizontal offset from placement (pt, in, mm, cm)
voffset	optional, vertical offset from placement (pt, in, mm, cm)
placement	optional, placement of the pages (any of TopLeft, TopCenter, TopRight, LeftCenter, Center, RightCenter, BottomLeft, BottomCenter, BottomRight)
placebelow	optional, places the chosen PDF underneath of the input PDF. Number of pages in resulting PDF is determined from number of pages in input PDF opened from 1: first argument (default) 2: second argument If no output name is defined, name of output file will be derived from second input file name.
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] [--ascii] [--workingspacecmyk=<ICC-profile>]
[--workingspacergb=<ICC-profile>]
[--workingspacegray=<ICC-profile>]
```

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)') Not available on Unix
colormanagement	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)
ascii	optional, Postscript is written 'Clean 7 Bit'
workingspacecmyk	optional, working space profile CMYK (default: ISO Coated v2 (ECI))
workingspacergb	optional, working space profile RGB (default: sRGB IEC61966-2.1)
workingspacegray	optional, working space profile Gray (default: Dot Gain 15%)

Example

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

Create PostScript

```
--createps [--transparencyquality=100]
[--gradientresolution=360] [--bitmapresolution=1200]
[--applyoutputpreviewsettings]
[--simulationprofile='ISO Coated v2 (ECI)'] [--colormanagement]
[--marksweight=0.125] [--pageinformation] [--colorbars]
[--registrationmarks] [--cutmarks] [--simulateoverprint]
[--postscript=3] [--ascii] [--workingspacecmyk=<ICC-profile>]
[--workingspacergb=<ICC-profile>]
[--workingspacegray=<ICC-profile>]
```

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 % (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)') Not available on Unix
colormanagement	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)
ascii	optional, Postscript is written 'Clean 7 Bit
workingspacecmyk	optional, working space profile CMYK (default: ISO Coated v2 (ECI))
workingspacergb	optional, working space profile RGB (default: sRGB IEC61966-2.1)
workingspacegray	optional, working space profile Gray (default: Dot Gain 15%)'

Example

```
pdfToolbox --createps --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.
For rendering purposes, the order in which profiles used as a working space (and in which is rendered) are determined:

- if an simulationprofile is defined, it will be used
- if a simulationprofile not defined, the Output Intent is used
- if no simulationprofile or Output Intent exists, the following profiles will be used: sRGB IEC61966-2.1; CMYK: ISO Coated v2 (ECI); Gray: Dot Gain 15%.

The defined simulation profile will only replace the default profile for the respective colorspace.
If the destination colorspace is same as used for rendering, this ICC profile will be used. Otherwise one of the following is used: sRGB IEC61966-2.1; CMYK: ISO Coated v2 (ECI); Gray: Dot Gain 15%.
As Rendering Intent "AC_RelColorimetric" is used by default.

Parameters

nosimulateoverprint	optional, avoids the overprint-simulation
simulationprofile	optional, using a user-defined ICC-profile for rendering
smoothing	optional, None, All, Lines, Images, Text, NTLH (default: All; NTLH includes "All")
resolution	optional, resolution in ppi or width x height

	in pixel, e.g. 1024x800 (default: 72)
colorspace	optional, one of RGB, RGBA, CMYK, Gray, Multichannel (default: RGB) availabilty depends on imageformat
jpegformat	optional, Baseline_Standard, Progressive_3_Scan (default: Baseline_Standard)
compression	optional, for JPEG: JPEG_minimum, JPEG_low, JPEG_medium, JPEG_high, JPEG_maximum (default: JPEG_medium) for TIFF: TIFF_None, TIFF_LZW, TIFF_Flate (default: TIFF_LZW)
imgformat	optional, JPEG, PNG, TIFF, PDF (default: JPEG)
pagebox	optional, using a geometry box as size for image: CROPBOX, TRIMBOX, BLEEDBOX, MEDIABOX (default: CROPBOX)
rect	optional, render only the part defined by lower left and upper right from origin geometry box (default: CROPBOX); in pt or mm (default:pt)
simulatepaper	optional, simulates paper color (not available if --nosimulateoverprint is set)
blackpointcompensation	optional, using blackpoint compensation (not available if --nosimulateoverprint is set)

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 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>
```

Extract images

```
--extractimages [--threshold=0] [--report=<path>]
```

Purpose

Extracts images from the file and creates a special XML report, which lists all extracted images with their relevant details.

Parameters

threshold	Extracts only images with width and height larger than threshold (default: 0)
report	Creates a report with details about the extracted images and their former posi tion in the PDF.

Example

```
pdfToolbox --extractimages --report --threshold=250 <PDF file>
```

This action can not be used with distributed processing.

Redistill

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

Purpose

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

Parameters

topdf_pdfsetting	full path to PDF settings file, must be a Distiller .joboptions file

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	full path to PDF settings file, must be a Distiller .joboptions file

Example

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

Uncertify

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

Purpose

Removes a Preflight certificate if present.

Example

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

Secure PDF

```
--securepdf --password=<password>
```

Restrict editing and printing of the PDF. A password is needed in order to change these permission settings or to perform changes. The PDF can only be read afterwards

The entered password will be visible and may be grabbed or logged by other processes on the machine.

Parameters

password	password to avoid editing or printing

Creating file packages

Some PDF standards allows the embedding of PDF- and also non-PDF-files into another PDF file. Sometime these file packages are also called collections. Using pdfToolbox CLI it is possible to create such file packages from a complete folder or to define different ways how a file which shall be embedded is handled.
In general a file package is created with `--collection` This will create an

index document, which lists all embedded files from the given folder. Also an existing folder structure will be respected

```
--collection <folder>
```

In general a file package is created with `--collection` This will create an index document, which lists all embedded files.

```
--collection <file> [<file>]
```

Settings for file embedding

```
--collection [--embedinto=[target],<file>] [--embedfile=
[target,[relationship],<file>] [--embedwithlink=
[area,<file>]
```

--embedinto

It is possible to use own templates or normal PDF for embedding files. The standard for the file where other files will be embedded can be defined using the conversion target (see below). If no file is defined, an index file is created.

--embedfile

Also for files to embed a conversion target can be defined using the conversion target. For PDF/A-3 standards also a relationship entry for each embedded file can be set.

Parameters

target	A3b, A3u, A3a, A2b, A2u, A2a, A1b, A1a or No (Default)

Using the target "No", no conversion to PDF is done. (Only available for embedded files.)

relationship	Source, Data, Alternative, Supplement, Unspecified (Default)

--embedwithlink

Alternatively, files can be embedded with defining an area in the containing document, where a link to the contained file is created. No conversion will take place with the file to embed.

Parameters

area	X1,X2,Y1,Y2[pt, in, cm, mm]

Defines a rectangular area, based on the lower left corner of the page, where a link to the embedded file is inserted. Default unit is pt.

Example:

```
--collection --embedinto=A3b,<PDF file> --embedfile=A3b,
Alternative,<file> --embedfile=A2b,Source,<Office file>
--embedfile=No,Data,<file>
```

```
--collection --embedwithlink=10,10,100,100,<file> --emb
edwithlink=10mm,100mm,100mm,200mm,<file>
```

Extracting files from file packages

```
--extractembeddedfiles [--plain] [--filter=<filtersettings>]
<PDF file>
```

Purpose

Extracts embedded files from a PDF.

Parameters

plain	Files are extracted directly into the destina tion folder without restoring an existing folder structure of the embedded files.
filter	

Example:

```
--extractembeddedfiles --plain --
```

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]
[--language=<language>]
```

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
language	language for naming of layers Supported values are: en English de German fr French es Spanish it Italian pt Brazilian Portoguese cz Czech da Danish nl Dutch fi Finnish ja Japanese ko Korean no Norwegian pl Polish sv Swedish

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>
```

List basic PDF info

```
--quickpdfinfo
```

Purpose

Lists some basic PDF information to output.

Example

```
pdfToolbox --quickpdfinfo <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)
sep_colors	Renders all individual separations in their respective color

Resolution output

Following you will find the values taken as tresholds 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
sep:<NAME>	separation of colorant with name <NAME>
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 default 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 [--threshold=20] [--areathreshold=5] [--diffres=150]
[--format=images] [--channels=rgb] [--nosimulateoverprint]
[--colorspace=colorspace] [--jpegformat=Baseline_Standard]
[--compression=JPEG_medium] [--imgformat=JPEG] [--resolu-
tion=72] <PDF file 1> <PDF file 2>
```

Purpose

Compares two PDF documents and creates a report.

Parameters

channels	optional, channels to be compared, any of: RGB (default, including spot colors), CMYK, CMY (both including spot colors), or as separation (without spot colors): PROCESS_CMYK, PROCESS_CMY, PROCESS_C, PROCESS_M, PROCESS_Y, PROCESS_K
highlighting	optional, highlighting format for differ ences, redmask (default), mask
diffres	Resolution used for comparison in ppi (Default = 72 ppi)
threshold	optional, defines the difference highlighting in % of the compared pixel, to be used

	instead of --sensitivity Default = 0% (highest sensibility)
sensitivity	deprecated parameter - use threshold controls difference highlighting, any of: maximum: Maximum sensitivity (default) medium: Medium sensitivity minimum: Minimum sensitivity
areathreshold	optional, defines the threshold for the area with pixel differences above value defined with parameter "threshold"
format	Compare report format, any of: pdfreport (default), imgreport, images, template
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) (only applicable if report format is images)
jpegformat	optional, Baseline_Standard (default), Progressive_3_Scan (only applicable if report format is images)
compression	optional, JPEG_low, JPEG_medium (default), JPEG_high (only applicable if report format is images)
imgformat	optional, JPEG, PNG, TIFF (default: JPEG) (only applicable if report format is images)

Example

```
pdfToolbox --compare --threshold=2 --areathreshold=5 <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 reseparation 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/Standalone can be defined in a XML-file.

Run as a Server / Distributed Processing

Start as a server

callas pdfToolbox Server/CLI can also be used for processing hotfolders on platforms, where no user interface for the configuration of the require settings is available (like Linux, SunSparc, SunIntel).
Remote access is not available for AIX.
This possiblity to start a server without a user interface is available on MacOS and Windows also of course.
First, the pdfToolbox CLI has to be started in server mode:

```
--server [--quiet] [--accesskey=accesskey]
[--port=port] [--cachefolder=cachefolder]
```

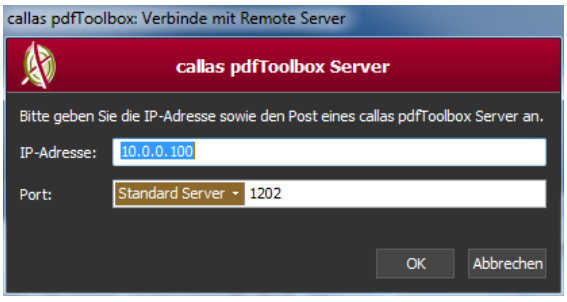
Options

--quiet	optional, suppresses outpput
--accesskey	optional, sets a accesskey for restricting the possibilty to change configuration using the server user interface
--port	optional, defines the port for communication between CLI and server user interface via the network (Default: 1202)
--cachefolder	optional, defines the path to cachefolder

Example:

```
--server --port=1202 --accesskey=123456
```

For setting up a job for hotfolder processing, connect to the remote server using any pdfToolbox desktop installation in the same network:



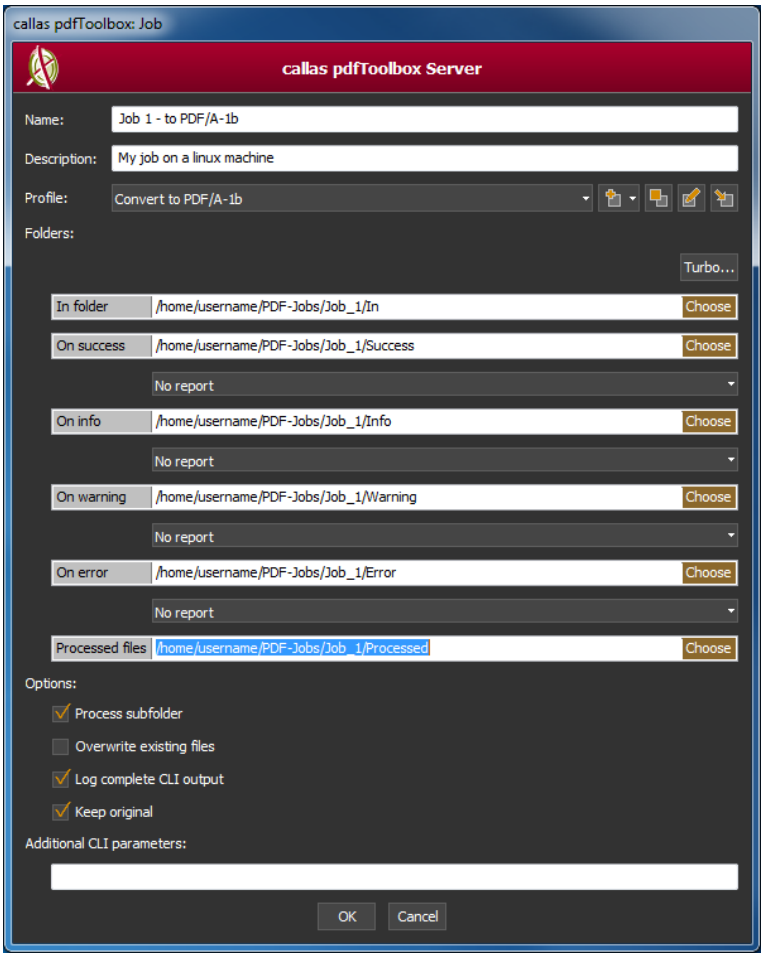
After entering the accesskey, new jobs can be configured, started or stopped. Profiles and Settings will be transfered to the remote server, where the are stored at /usr/share/callas software (path must be writable)

If this location can not be used caused by limitations on the respective environment, the additional option --cachefolder can be used for defining a custom path when starting the server:

Example:

```
--server --port=1202 --cachefolder=<PATH>
```

All paths defined for hotfolder processing need to be entered manually and have to be valid path specifications. (Hotfolder paths of any remote server jobs (IN, OUT, etc.) have to be configured so that they are valid from the service's perspective (the system where the service is running) - and not from the perspective of the controlling standalone application.) The server can also be stopped by remote, but not started.

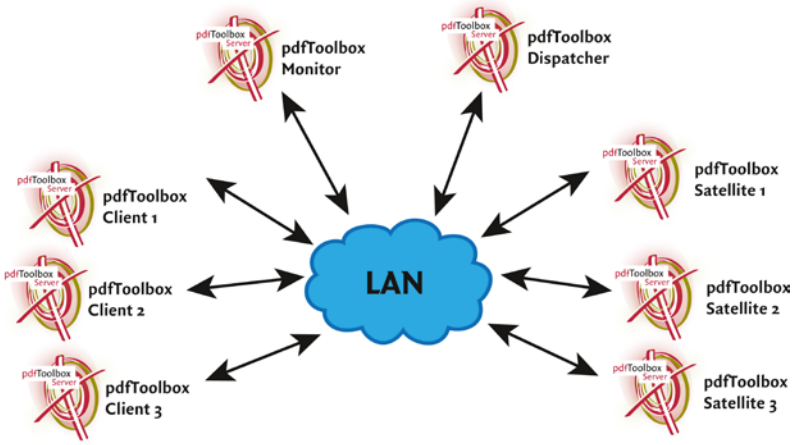


Distributed Processing mode

callas pdfToolbox Server/CLI can be used in distributed processing mode in which all jobs are distributed over the network to as many "satellites" as being present and results are send back to the originator. Therefore pdfToolbox Server/CLI may be started in different modes:

- "Dispatcher" must to be present exactly one time in the network. This node controls which jobs are to be processed by which machines: the "satellites".
- "Satellite" receives jobs from the clients or directly from the dispatcher (if the dispatcher is run with hotfolders), processes them and sends them back to the clients.
- "Client" asks the dispatcher for satellites and after receiving the next satellite it sends jobs to the satellites and receives the results.
- "Monitor" monitors the dispatcher and displays the current situation.

All of these modules can run on the same or on different machines. There needs to be exactly one dispatcher and at least one satellite. In order to submit jobs at least one client is required.
Distributed processing is supported for Windows, MacOS, Linux, SunSolaris and SunIntel. It is not available on AIX.



Starting a Dispatcher

```
--dispatcher [--port=<port number>]
```

Example:

```
--dispatcher [--port=1200]
```

Port is the port number on which the dispatcher can be called over the network. This port is set to 1200 as default.

Starting a Dispatcher using the ServerUI

There is also the possibility to start a server as a dispatcher on Windows and MacOS using the user interface. Also hotfolder-processing can be set up here. In this mode, the dispatcher will also distribute jobs which are send by other clients.

Starting a Satellite

```
--satellite --endpoint=<dispatcher IP
number>[:<dispatcher port]> [--port=<port number>]
[--connections=<number of concurrent connections]
```

Example:

```
--satellite --endpoint=10.0.0.100:1200 --port=1201
```

In order to process jobs at least one satellite is required. Endpoint is the IP number and the port of the dispatcher. Port is the port that the satellite is using in order to communicate with the clients. The port of the satellite is 1201 as default and can be defined optionally to another one.

Starting a Satellite using the ServerUI

There is also the possibility to start a server as a satellite on Windows and MacOS using the user interface. In this mode, the satellite will not process any hotfolder jobs on the computer.

A Satellite will always use the number of CPUs on the respective machine as the number of concurrent connections/processes. To limit this number, the Satellite has to be started by CLI with the `--connections` parameter.

Distribute a process using a Client

The client is called using any regular pdfToolbox command line command. In order to distribute the call over the network the command line parameters --dist and --endpoint are added. The client will then first ask the dispatcher to receive a satellite connection and then send the command to the satellite and wait until the result is sent back from the satellite.

```
pdfToolbox --dist --endpoint=<dispatcher IP
number>[:<dispatcher port>] <any regular pdfToolbox
call>
```

Examples:

```
pdfToolbox --dist --endpoint=10.0.0.100:1200
<anyProfile.kfpx> <myPDF.pdf>
```

```
pdfToolbox --dist --endpoint=10.0.0.100:1200
--redistill <myPDF.pdf>
```

Set type of satellite

As some kinds of jobs shall only be processed on a defined type of Satellite, it is possible to start a Satellite with one or more types set.
Every CLI call can also be amended with one or more typification of allowed types of Satellites the job shall be processed by.

Set typification for Satellite:

```
pdfToolbox --satellite --endpoint=<dispatcher
IP number> --satellite_type=<type> [--satellite_
type=<type>]
```

for example:

```
pdfToolbox --satellite --endpoint=10.0.0.100
--satellite_type=A
```

```
pdfToolbox --satellite --endpoint=10.0.0.100
--satellite_type=A --satellite_type=B
```

Set typification for Client:

```
pdfToolbox --dist --endpoint=<dispatcher IP number>
--satellite_type=<type> [--satellite_type=<type>] <any
regular pdfToolbox call>
```

for example:

```
pdfToolbox --dist --endpoint=10.0.0.100 --satellite_
type=A <any regular pdfToolbox call>
```

Implementation details:

- If a Satellite has been started with a typification, only Client calls with the same type set will be send to this satellite.
- If a Client call contains a number of typifications, all typifications must match with those set for a satellite.
- If a Client call has no typfication set, it can be processe on all satellites, even they have been started with a typfication.
- The `<type>`-string has to be alpha-numeric and is case sensitive.

Avoid local processing

As a fallback, processing can be performed locally if either the action can not be distributed, a Satellite can not be assigned within a timeframe or if no Dispatcher is available.
This type of local processing might be not desired for several reasons.
To avoid such local processing, the Client call can be amended as well as the start of a Dispatcher (if run as a server with hotfolders) with the option `--nolocal`.

Example for Client:

```
pdfToolbox --dist --endpoint=<dispatcher IP number>
--nolocal <any regular pdfToolbox call>
```

Example for Dispatcher:

```
pdfToolbox --dispatcher --nolocal
```

Fallback for Dispatcher

In some workflow systems, a fallback for a Dispatcher might be required to ensure production stability.
To cover this, a number of Dispatcher can be set up, which will run individually. One or multiple Dispatcher can be assigned to a Satellite.

Define multiple Dispatcher to a Satellite

Connects a satellite to two (or more) Dispatcher.

```
pdfToolbox --satellite --endpoint=<dispatcher 1 IP>
[--endpoint=<dispatcher 2 IP> [--endpoint=<dispatcher
IP>]
```

Set multiple Dispatcher in a Client call

Distributes a Client call via two (or more) Dispatcher. First reachable Dispatcher with free satellite will process the job.

```
pdfToolbox --dist --endpoint=<dispatcher 1 IP>
--endpoint=<dispatcher 2 IP> [--endpoint=<dispatcher
IP>] <any regular pdfToolbox call>
```

Define a timeout for processing

In some workflow systems, long running processes might not be allowed and shall be cancelled if a give timeframe is reached.
Due to the flexibility of distributed processing, a variety of timeouts for the individual parts can be set:

- for the Cient call
- for the Satellite
- for the Dispatcher

Timeout for processing on a Satellite

When defining a timeout for the Client call, the execution will be cancelled after the given period.
When defining a timeout when starting a Satellite, all jobs processed by this Satellite will be cancelled after the given period.
If both are defined, the shorter timeframe will be used.

Example for Client:

```
pdfToolbox --dist --endpoint=<dispatcher IP> --timeout_
satellite=<seconds> <any regular pdfToolbox call>
```

Example for Satellite:

```
pdfToolbox --satellite --endpoint=<dispatcher IP>
--timeout=<seconds>
```

Timeout for local processing of Dispatcher or Client

A processing timeout (if no satellite is available or if the type of job can not be distributed) for the fallback to local processing on the Client or the Dispacher (when used as a server for hotfolders) can also be defined.
If both are defined, the shorter timeframe will be used.

Example for Client:

```
pdfToolbox --dist --endpoint=<dispatcher IP>
--timeout=<seconds> <any regular pdfToolbox call>
```

Example for Dispatcher:

```
pdfToolbox --dispatcher --timeout=<seconds>
```

Timeout for Dispatcher to search for Satellites

Additionally, also a timeout for the Dispatcher can be set, which will define the timeframe in which is searched for Satellites.
This can also be set individually for every Client call or when starting the Dispatcher (will have effect on all distributed files then).
If both are defined, the shorter timeframe will be used.

Example for Client:

```
pdfToolbox --dist --endpoint=<dispatcher IP> --timeout_
dispatcher=<seconds> <any regular pdfToolbox call>
```

Example for Dispatcher:

```
pdfToolbox --dispatcher --timeout_dispatcher=<seconds>
```

- If a timeout for satellites or dispatcher is set and the --nolocal option has been defined, it will not be tried to process the job locally. Processing will end up in an error.
- Setting --timeout... or --nolocal parameters in the "Additional CLI parameter" area of the Server UI is not supported at the moment.

Using the CLI-Monitor

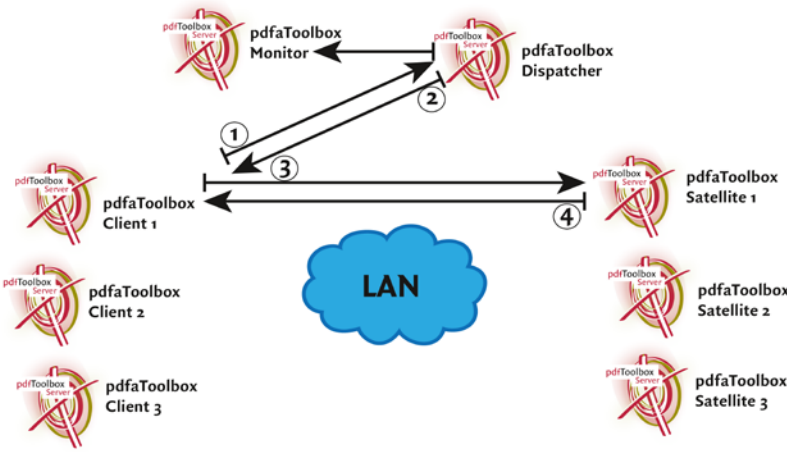
```
pdfToolbox --monitor --endpoint=<dispatcher
IP>:<dispatcher port> [--endpoint=<dispatcher
IP>:<dispatcher port>]
```

Example:

```
--monitor --endpoint=10.0.0.100:1200
```

Monitor is optional and mirrors the command line output of the dispatcher to another computer. Endpoint is the IP number and the port of the dispatcher.
When using more than one Dispatcher, also multiple Dispatcher IPs can be entered and observed.

Communication



1) Clients sends a request for Satellite to Dispatcher
2) Dispatcher assigns a Satellite and send the address to the Client
3) Client send the job to the Satellite
4) Satellite send the result back to the Client

Licensing

- Server: Regular pdfToolbox Server/CLI license required
- Dispatcher: Dispatcher pdfToolbox Server/CLI license required
- Satellite: Regular pdfToolbox Server/CLI license required
- Monitor: No license required
- Client: No license required

Run as a Service

Start as a server, dispatcher or satellite

The callas pdfToolbox Services application is only available for Windows at the moment.

Installation

1.
Ensure there is an installation of callas pdfToolbox Server/CLI on the system and the application has been activated successfully.

2.
A special executable, which is needed to run pdfToolbox as a service, is located in /cli/var/Service.
Copy the executable into the subfolder "/cli" of the application folder of the server installation.

3.
To install, the following command has to be executed on the command line:

```
pdfToolboxService.exe --install
```

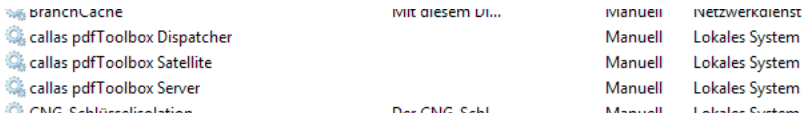
Please confirm the security question of Windows if shown.

4.
Open the "Services dialog" of Windows. This dialog can easily opened by typing the following string into the search field of the Windows start menu or use the following command on the command line:

```
Services.msc
```

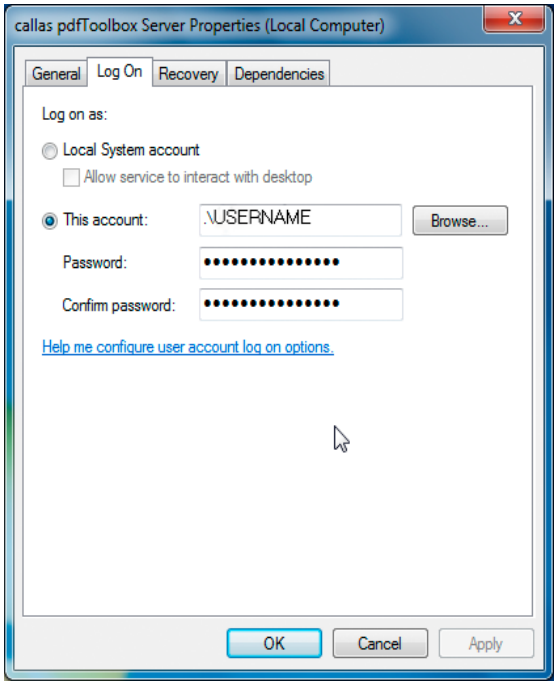
5.
There should show up 3 new services:

- callas pdfToolbox Server
- callas pdfToolbox Satellite
- callas pdfToolbox Dispatcher



6.

Select "pdfToolbox Server" and use the right-click menu item "Action" [or "Properties" depending on the Windows-version] in order to use this service and set in "General" the "Startup Type" to "Automatic" or "Manual".
When "Automatic" is choosen, every started job will continue processing, even when no user is logged on the system. It will even start processing, when the operating system is started.
When using "Automatic", also user details have to be entered into the "Log On" tab.
It must be ensured, that all folders used in the jobsettings can be accessed by the defined user (especially when network paths shall be used by the job).



Configration of a job

Now a job can be configured using the ServerUI, which can be accessed using the Standalone version (Menu: Tools - Server).
When a job is started, pdfToolbox Standalone can be closed. The services apllication ensures, that the processing will continue even when the user is logging off.

Access by remote

It is possible to connect to a Server running as a service by remote via the local network.
Start pdfToolbox standalone, select menu: "Tools" - "Server" and choose "Connect with remote server".
Enter the IP of the remote server where the service is running.
After connecting all jobs on the remote server are shown and can be started, stopped or even modified. (Hotfolder paths of any server jobs (IN, OUT, etc.) have to be configured so that they are valid from the service's perspective (the system where the service is running) - and not from the perspective of the controlling standalone application.)

Troubleshooting

If network paths are used for processing jobs, the user should have sufficient rights to access them.

There may be special requirements for converting Office files to PDF when using pdfToolbox as a service. Check http://www.callassoftware.com/callas/doku.php/en:support:faqs:topdf for the latest details.