User Manual

3-Heights®
PDF Optimizer Shell

Version 6.27.2

pdftools

Contents

1 Introduction

1.1 Description

The 3-Heights® PDF Optimizer Shell optimizes PDF documents to suit specific target needs such as electronic document exchange, archiving, or printing.

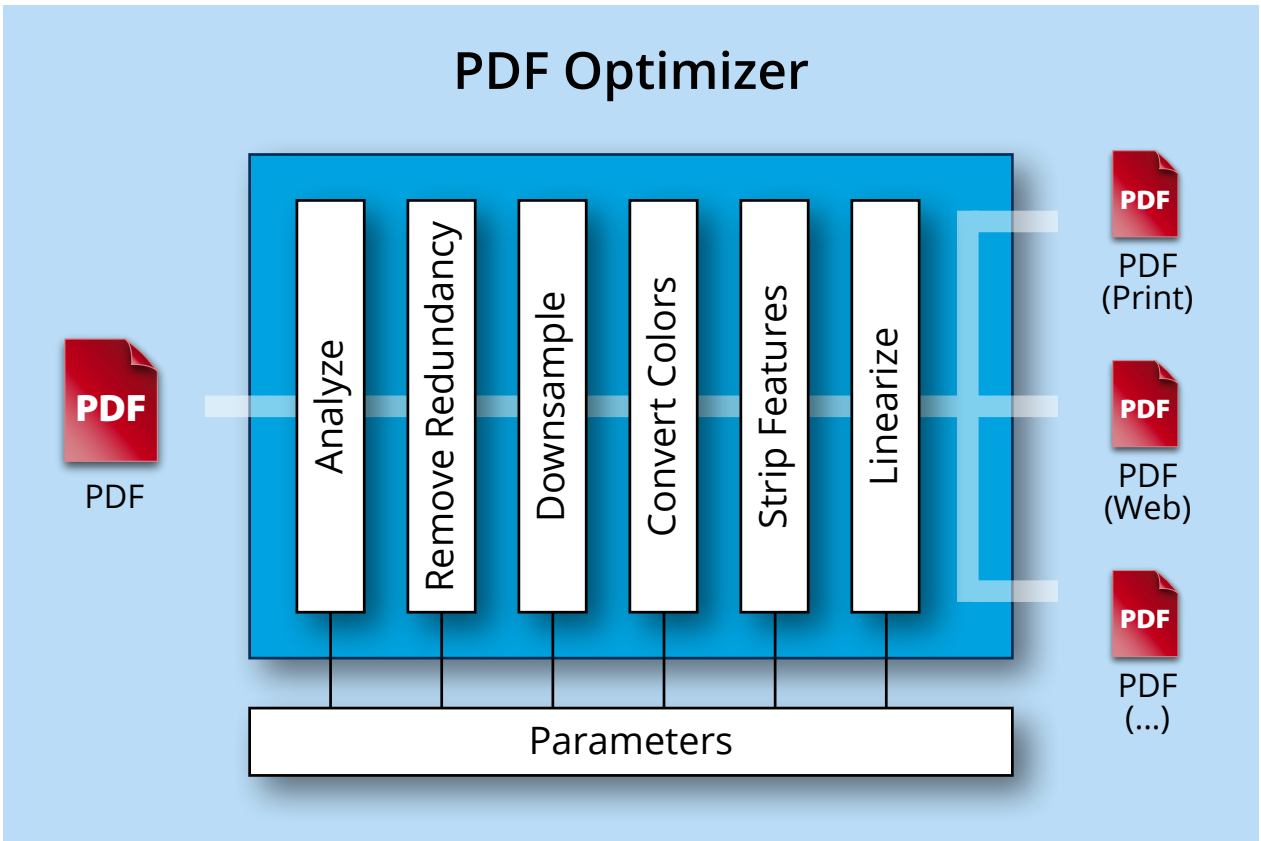
Many processes produce PDF documents that may not be optimized for their specific target application. For example, the file size may deteriorate download times, or the same font embedded multiple times may impede printing. In most cases, there is no advantage gained when trying to convert a PDF to some other file format. In contrary, document content may be compromised and file size may increase. Optimization, on the other hand, often leads to good results, or lets the user finely tune trade-offs.

The 3-Heights® PDF Optimizer Shell not only provides easy configuration through the use of optimization profiles, but also flexible fine-grained control through various specific options.

1.2 Functions

The 3-Heights® PDF Optimizer Shell reads an input document and writes the corresponding output document. Depending on the configured optimization options, various parts of the PDF are thereby processed as required.

The 3-Heights® PDF Optimizer Shell is capable of removing redundant or alternative information, sub-setting and merging font programs, downsampling images, intelligently choosing optimal compression algorithms, and linearizing the PDF for fast web view.



1.2.1 Features

- Configure easily through optimization profiles, three of which are as follows:
  - Web profile
    - Remove redundant and unnecessary data for electronic document exchange
    - Downsample, clip, and intelligently compress images
    - Merge and subset fonts
    - Linearize the output
    - Convert colors to RGB
  - Archiving profile
    - Remove redundant and unnecessary data for archiving
    - Intelligently compress images
    - Merge and subset fonts
  - Print profile
    - Remove redundant and unnecessary data for printing
    - Downsample, clip, and intelligently compress images
    - Merge and subset fonts
    - Convert colors to CMYK
- Optimize images
  - Separately configure compression of bitonal, indexed and continuous (i.e. color and grayscale) images
  - Define threshold in dots per inch (DPI) for triggering image downsampling
  - Define target image resolution in DPI for image downsampling
  - Automatically select best compression type for each image
  - Configure enforcement of configured compression types
  - Convert colors to CMYK, RGB, or grayscale
  - Remove invisible parts of images
  - Reduce the number of color channels used for images, image masks, and soft masks if applicable
  - Convert soft masks to image masks if applicable
  - Perform mixed raster content (MRC) optimization for images
  - Choose color management engine
  - Remove images entirely and substitute by empty XObjects
- Optimize fonts
  - Subset font programs to contain only the used glyphs
  - Merge compatible font programs and fonts
  - Compress Type 1 fonts (convert to CFF)
  - Remove font programs
- Optimize page content
  - Remove unused resources
  - Optimize page content automatically
  - Flatten or remove page annotations and form fields
- Configure and remove, where appropriate:
  - Redundant objects
  - Embedded standard fonts (e.g. Courier, Arial, Times)
  - Embedded, non-symbolic fonts
  - Unnecessary file information
  - Article threads
  - Alternative images
  - Metadata
  - Page piece information
  - Output intent
  - Document structure tree, including markup
  - Miniature page preview images

    - Spider (web capture) information
- Configure at file level
    - Read encrypted input files
    - Encrypt and set access authorization for the output file
    - Automatically remove obsolete objects that stem from previous changes to the file
    - Set minimum PDF version of the output file
    - Linearize output file for fast web view (not PDF 2.0)
- List and extract information
    - List fonts and their properties
    - List and extract images and their properties
    - Error code

1.2.2 Formats

Input formats

- PDF 1.x (PDF 1.0, …, PDF 1.7)
- PDF 2.0

Output formats

- PDF 1.x (PDF 1.0, …, PDF 1.7)
- PDF 2.0

1.2.3 Conformance

Standards:

- ISO 32000-1 (PDF 1.7)
- ISO 32000-2 (PDF 2.0)

1.3 Operating systems

The 3-Heights® PDF Optimizer Shell is available for the following operating systems:

- Windows Client 7+ | x86 and x64
- Windows Server 2008, 2008 R2, 2012, 2012 R2, 2016, 2019, 2022 | x86 and x64
- Linux:
    - Red Hat, CentOS, Oracle Linux 7+ | x64
    - Fedora 29+ | x64
    - Debian 8+ | x64
    - Other: Linux kernel 2.6+, GCC toolset 4.8+ | x64
- macOS 10.10+ | x64

'+' indicates the minimum supported version.

2 Installation

2.1 Windows

The 3-Heights® PDF Optimizer Shell comes as a ZIP archive or as an MSI installer.

To install the software, proceed as follows:

1. You need administrator rights to install this software.
2. Log in to your download account at https://www.pdf-tools.com. Select the product "PDF Optimizer Shell". If you have no active downloads available or cannot log in, please contact pdfsales@pdf-tools.com for assistance.
   You can find different versions of the product available. Download the version that is selected by default. You can select a different version.
   There is an MSI (`*.msi`) package and a ZIP (`*.zip`) archive available. The MSI (Microsoft Installer) package provides an installation routine that installs and uninstalls the product for you. The ZIP archive allows you to select and install everything manually.
   There is a 32 and a 64-bit version of the product available. While the 32-bit version runs on both 32 and 64-bit platforms, the 64-bit version runs on 64-bit platforms only. The MSI installs the 64-bit version, whereas the ZIP archive contains both the 32-bit and the 64-bit version of the product. Therefore, on 32-bit systems, the ZIP archive must be used.
3. If you select an MSI package, start it and follow the steps in the installation routine.
4. If you are using the ZIP archive, uU nzip the archive to a local folder, e.g. `C:\Program Files\PDF Tools AG\`.
   This creates the following subdirectories:

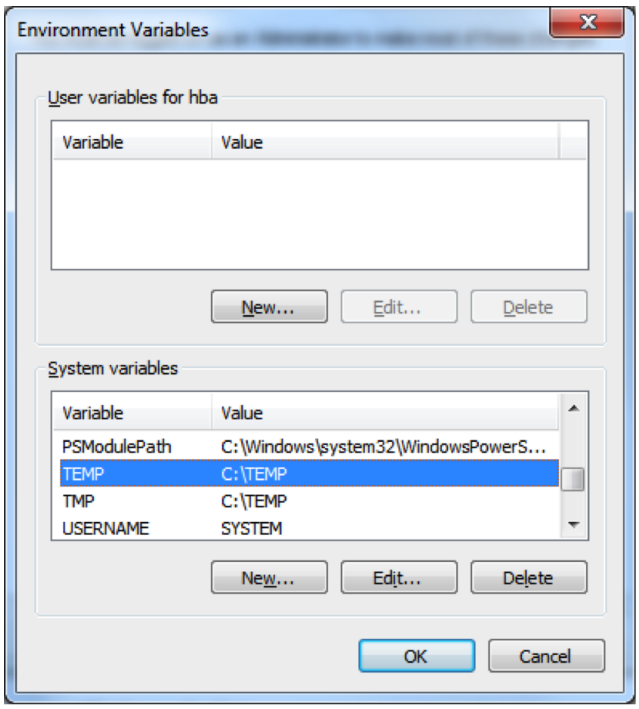
Subdirectory	Description
`bin`	Runtime executable binaries
`doc`	Documentation

5. (Optional) To easily use the 3-Heights® PDF Optimizer Shell from a shell, the directory needs to be included in the "Path" environment variable.
6. (Optional) Register your license key using the License management.
7. Make sure your platform meets the requirements regarding color spaces described in Color profiles.

2.1.1 How to set the environment variable "Path"

To set the environment variable "Path" in Windows, go to Start → Control Panel (classic view) → System → Advanced → Environment Variables.

Select "Path" and "Edit", then add the directory where `pdfoptimize.exe` is located to the "Path" variable. If the environment variable "Path" does not exist, create it.



2.2 Linux and macOS

This section describes installation steps required on Linux or macOS.

Here is an overview of the files that come with the 3-Heights® PDF Optimizer Shell:

File description

Name	Description
`bin/x64/pdfoptimize`	Main executable
`doc/*.*`	Documentation

2.2.1 Linux

1. Unpack the archive in an installation directory, e.g. `/opt/pdf-tools.com/`
2. Verify that the GNU shared libraries required by the product are available on your system:

```
ldd pdfoptimize
```

   If the previous step reports any missing libraries, you have two options:
   a. Download an archive that is linked to a different version of the GNU shared libraries and verify whether they are available on your system. Use any version whose requirements are met. Note that this option is not available for all platforms.
   b. Use your system's package manager to install the missing libraries. It usually suffices to install the package `libstdc++6`.
3. Create a link to the executable from one of the standard executable directories, e.g.

```
ln -s /opt/pdf-tools.com/bin/x64/pdfoptimize /usr/bin
```

4. Optionally, register your license key using the license manager.
5. Make sure your platform meets the requirements regarding color spaces described in Color profiles.

2.3 Uninstall

If you have used the MSI for the installation, go to Start → 3-Heights® PDF Optimizer Shell… → Uninstall …

If you have used the ZIP file for the installation, undo all the steps done during installation.

2.4 Color profiles

The color conversion feature of the 3-Heights® PDF Optimizer Shell uses color profiles by default.

For calibrated color spaces (such color spaces with an associated ICC color profile), the color conversion is well defined. For the conversion of uncalibrated device color spaces (DeviceGray, DeviceRGB, DeviceCMYK), however, the 3-Heights® PDF Optimizer Shell requires apropriate color profiles. Therefore, it is important that the profiles are available and that they describe the colors of the device your input documents are intended for.

> **Note:** When setting an alternative color management system such as Neugebauer, no color profiles are required.

If no color profiles are available, default profiles for both RGB and CMYK are generated on the fly by the 3-Heights® PDF Optimizer Shell.

2.4.1 Default color profiles

If no particular color profiles are set, default profiles are used. For device RGB colors, a color profile named `"sRGB Color Space Profile.icm"` and for device CMYK, a profile named `"USWebCoatedSWOP.icc"` are searched for in the following directories:

Windows

1. `%SystemRoot%\System32\spool\drivers\color`
2. directory `Icc`, which must be a direct subdirectory of where the `pdfoptimize.exe` resides.

Linux and macOS

1. `$PDF_ICC_PATH` if the environment variable is defined
2. the current working directory

2.4.2 Get other color profiles

Most systems have pre-installed color profiles available. For example, on Windows at `%SystemRoot%\system32\spool\drivers\color\`. Color profiles can also be downloaded from the links provided in the directory `bin\Icc\` or from the following websites:

- https://www.pdf-tools.com/public/downloads/resources/colorprofiles.zip
- https://www.color.org/srgbprofiles.html

2.5 Note about the evaluation license

With the evaluation license, the 3-Heights® PDF Optimizer Shell automatically adds a watermark to the output files.

3 License management

The 3-Heights® PDF Optimizer Shell requires a valid license in order to run correctly. If no license key is set or the license is not valid, then the executable will fail and the return code is set to 10.

More information about license management is available in the license key technote.

3.1 License features

The functionality of the 3-Heights® PDF Optimizer Shell contains two areas to which the following license features are assigned:

`Optimize` General optimization

`Color` Optimizations involving color conversion

A license can include an arbitrary set of these features. The presence of any feature in a given license key can be checked in the license manager. The Interface reference specifies in more detail which functions are included in which license features.

4 Getting started

The simplest command requires two parameters: The names of the PDF input and output files.

```
pdfoptimize input.pdf output.pdf
```

This command generates an new PDF file with almost no optimization done at all. To perform an actual optimization, more parameters have to be specified in the form of options. For example, in the following code sample, the "web" optimization profile is selected (see `-pr`):

```
pdfoptimize -pr web input.pdf output.pdf
```

All options start with a hyphen (-). Some options require additional parameters. All options are documented in the Interface reference.

4.1 Usage

By typing `pdfoptimize` without parameters, the usage, the version, and a list of available options is returned.

4.2 Specify the folder of the output file

The output folder can simply be added in front of the output file name.

```
pdfoptimize input.pdf myfolder\output.pdf
```

or absolute (Windows):

```
pdfoptimize input.pdf C:\myfolder\output.pdf
```

4.3 Processing all files in a folder

A simple way to optimize all files in a specified directory using a command line uses a for-loop and a variable to name the current input document inside the loop.

Example: Optimize all PDFs in the current directory and save the results in files that have a suffix `-opt` in the same directory

Windows CMD

```
for %i in (*.pdf) do pdfoptimize -v -pr web "%i" "%~ni-opt.pdf"
```

Unix shell

```
for f in *.pdf; do pdfoptimize -v -pr web "$f" "$(basename "$f" .pdf)-out.pdf"; done
```

Example: Optimize all PDFs in a specified input directory and save the results using the same file names in a specified output directory

Windows CMD

```
for %i in (C:\in\*.pdf) do pdfoptimize -v -pr web "%i" "C:\out\%~ni.pdf"
```

Unix shell

```
for f in ~/in/*.pdf; do pdfoptimize -v -pr web "$f" "~/out/$(basename "$")"; done
```

When using variables in a Windows batch file (`.bat`) instead of on the Windows command line, the variables have to be written with two leading percent characters (e.g. `%%i`) instead of just one (`%i`).

4.3.1 Windows batch sample

Where all files in a directory need to be processed and the optimized file must have the same name as the original document, i.e. overwrite it, the following approach can be used:

- Make sure you really want to perform the task: the original file is lost in this process!
- Create the output files, either with a different name or in a different directory.
- Ensure the output files are created correctly. This can be done by verifying the return code (it must be 0), or verify the document was created at all and is not empty.
- Delete the original file.
- Rename or copy back the new file to replace the original file.

The following sample does the steps described above. This sample does not ensure to always yield a correct result. Errors in the optimization or an abort of the process can still lead to loss of data. It is suggested to keep a backup of the original files.

```
@ECHO off
rem ***************************************************************************
rem * This batch files optimizes all PDF files in the current directory     *
rem *                                                                         *
rem * The steps are as following:                                           *
rem *                                                                         *
rem * 1. Optimize all files in a folder. The optimized output files         *
rem *    have the temporary extension .tmp.                                   *
rem *                                                                         *
rem * 2. If the return code of the pdfoptimize is 0 and an output is        *
rem *    created, the optimization process is considered successful.          *
rem *                                                                         *
rem * 3. If successful, the original input file is deleted and the          *
rem *    .tmp file is renamed to .pdf.                                        *
rem *                                                                         *
rem *    If the process was not successful, the .tmp file is deleted         *
rem *    and the original file is left as is.                                 *
rem ***************************************************************************

if exist *.tmp del /f /q *.tmp
for %%i in (*.pdf) do (
   set name=%%~ni
   call :_Optimize
)
```

```
goto :eof
rem ************************************************************************
:_Optimize
pdfoptimize -pr web "%name%.pdf" "%name%.tmp"
if not %errorlevel%==0 (
   @echo ** Optimization failed for %name%.pdf [error code %errorlevel%].
   if exist "%name%.tmp" del /f /q "%name%.tmp"
) else (
   if exist "%name%.tmp" (
      if exist "%name%.pdf" (
         del /f /q "%name%.pdf"
         if not exist "%name%.pdf" (
            rename "%name%.tmp" "%name%.pdf"
            @echo -- Optimization successful for %name%.pdf.
         ) else (
            del /f /q "%name%.tmp"
            @echo ** Optimization failed for %name%.pdf [file locked].
         )
      )
   ) else (
      @echo ** Optimization failed: %name%.pdf [no output file].
   )
)
goto :eof
```

To optimize all files in all subfolders, it is best to create a batch file that runs through all subfolders and executes the batch file above.

Create a batch file called `run.bat` and copy the above code in it.

Then create another batch file called, for example, `runsub.bat` and add the code below:

```
@echo off
for %%r in (.\) do set rootfolder=%%~pr
for /r %%s in (.) do (
   cd %%s
   call %rootfolder%run.bat
)
cd %rootfolder%
set rootfolder=
```

Now copy the two batch files to the root folder (i.e. the folder from which every PDF file in every subfolder should be processed) and run the batch `runsub.bat`.

5 Optimization process

5.1 Optimizing PDF documents

5.1.1 Identifying target application area

PDF documents are used in a wide variety of application areas, all having different requirements. As a very first step, you should precisely identify the targeted application area. A few typical fields of application are described briefly below. However, PDF documents can also be used in other ways or in combinations of the ones listed below.

Web

All documents related to the web should be kept small in file size. As a consequence, they take less storage on the web server and can be transferred quicker, resulting in shorter download times.

To reduce the file size as much as possible, all information that is not required for displaying the document without a visual loss can be removed. This may include:

- Downsampling images (`-dt`, `-dr`)
- Clipping images to their visible parts (`-oc`)
- Applying compressions algorithms with high compression ratios (`-fb`, `-fc`, `-fi`)
- Collapsing redundant objects (`-or`)
- Removing unused resources (`-od`)
- Removing irrelevant information such as article threads, metadata, alternate images, document structure information, etc. (Strip the file)
- Merging and sub-setting embedded font programs (`-s` and `-m`)
- Depending on the PDF documents to be optimized, font programs of embedded standard fonts can even be removed (`-rs`).

Additionally, PDF documents can be linearized (`-ow`). This is a method of preparing a PDF file so that pages can be accessed randomly via a PDF viewer web browser plugin, i.e. selected pages can be displayed before the whole file is downloaded. For this to work, the PDF viewer web browser plugin has to support correct interpretation of linearized PDF.

Documents that are intended to be displayed on a display should be saved in RGB (red green blue) color space. RGB is the native form for any light-emitting device, such as computer monitor or television. An RGB image uses three channels, and therefore takes up less space than a CMYK (cyan magenta yellow black) image, which uses four channels. (`-c`)

Printing

For printing applications, the file size is not the highest priority. It is more important to have a document which prints predictably. This means that correct fonts should be used, colors should look as expected, and images should be high in resolution.

For that reason, data from the original document that is used for a well-defined re-production should not be removed or altered. Fonts should not be un-embedded, images should not be downsampled. (Of course, there are always exceptions).

For many printing applications, it may be beneficial to convert images to the CMYK color space because this is primarily used in systems that reflect light (such as printed paper). (`-c`)

In certain documents, the same font is embedded multiple times. For example, if a PDF-producing software embeds the same font for each created page, then large multi-page documents may contain many copies of a font program. Also, a document can contain a complete font program of which only very few glyphs are used for display. In such situations, merging and sub-setting font programs can lead to faster printing. (`-m` and `-s`)

There are still further ways to decrease the file size:

- Clipping images to their visible parts (`-oc`)
- Compressing uncompressed images, e.g. with a lossless compression type (`-fb`, `-fc`, `-fi`, see also Supported image compression types)
- Collapsing redundant objects (`-or`)
- Removing unused resources (`-od`)
- Removing irrelevant information for printing such as thumbnails, article threads, document structure information, etc. (Strip the file)

Archiving

Archiving can have varying requirements, such as to minimize the file size, maximize the reproducibility of the document, and minimize the access time to find a specific archived document.

The most common way for archiving a PDF is the PDF/A format, which is defined in the ISO 19005 standard. PDF/A requires fonts to be embedded, metadata to be included, and prohibits certain features such as LZW or JPEG2000 compression or alternate images. The 3-Heights® PDF Optimizer Shell does not create PDF/A output but can be used to reduce the file size prior to converting to PDF/A.

Scanned documents

For certain types of scanned documents, MRC (mixed raster content) optimization can have a significant impact on file size while still preserve the visual appearance of the document. The 3-Heights® PDF Optimizer Shell supports MRC, see also MRC optimization for images.

Special requirements

As an example for a specific requirement, the 3-Heights® PDF Optimizer Shell supports the restriction of compression types for images in a document to a specific list of types. (See `-ft`, `-fb`, `-fc`, and `-fi`.)

Another requirement may be to encrypt the resulting PDF and protect it by a user password and/or by an owner password. The 3-Heights® PDF Optimizer Shell provides both by means of the `-p`, `-u`, and `-o` options.

PDF documents that mainly consist of scanned images to which an OCR (optical character recognition) layer is applied at a later stage should be optimized in a way that the OCR process of the optimized document works as well as with the original. That means that image compression should either be lossless or at least perceptually lossless. Perceptually lossless refers to a compression that is lossy, but its visual quality is high enough that neither the human eye nor an OCR engine can distinguish between original and optimized document. (See also Supported image compression types.)

5.1.2 Using optimization profiles

The 3-Heights® PDF Optimizer Shell provides the notion of “optimization profiles” to quickly configure the tool for many application areas. Once the application area is defined, the optimization profile that best matches the requirements can be identified. The configuration is done by setting this profile (`-pr`) followed by adjusting individual settings that differ from the profile.

```
pdfoptimize -pr web -dbt -1 input.pdf output.pdf
```

The actual profile settings for all profiles are in Profile settings.

5.2 Optimizing images

For the 3-Heights® PDF Optimizer Shell, the target compression type of an image is specified by giving a numerical value between `-1` and `10` to the `-fb`, `-fc`, and `-fi` options. Several values can be combined in a comma-separated list. (No spaces are allowed in the list)

The `1` to `8` values directly correspond to PDF compression types. Other values indicate a special behavior.

Value	Compression
`0`	No compression (raw)
`1`	DCT (JPEG)
`2`	Flate (ZIP)
`3`	LZW
`4`	CCITT Fax Group 3 (CCITT Fax Group 3 and 4)
`5`	CCITT Fax Group 3 2D (CCITT Fax Group 3 and 4)
`6`	CCITT Fax Group 4 (CCITT Fax Group 3 and 4)
`7`	JBIG2 (Supported in PDF 1.4 or later)
`8`	JPEG2000 (Supported in PDF 1.5 or later. Bot supported in PDF/A-1)
`9`	In contrast to the values 0-8, this is not a single compression format. Instead, this enables MRC optimization on color and monochrome images. (See MRC optimization for images) Application area: Scanned documents.
`10`	In contrast to the values 0-8, this is not a single compression format. Instead, this tells 3-Heights® PDF Optimizer Shell to use the same compression as the original input image.
`-1`	Exclude from processing

5.2.1 Supported image compression types

In PDF, up to 8 different ways of compressing binary data are supported. (See also PDF Reference 1.7, Chapter 3.3 for more information on these types.)

No compression (raw)

Raw means no compression is applied.

DCT (JPEG)

Developer	Joint Photographic Experts Group committee
Version	PDF 1.2 and later, PDF/A-1
Color depth	8, 24 bits per pixel
Compression type	Lossy
Compression algorithm	The image is broken up into blocks that are 8 by 8 samples. On each of these blocks and color channel, a discrete cosine transformation (DCT) is applied and its coefficients are quantized. The visual quality of the resulting image depends on the loss of information defined by the step size of the quantization and on the image that is being compressed. The compression can be controlled via an image quality parameter—a value from 1 to 100 (default 75). Typical compression ratios are 15:1 (no perceptible loss of information) to 30:1.
Application area	Sampled continuous-tone pictures (photographs)

Flate (ZIP)

Developer	Flate compression is based on the public-domain zlib / deflate compression method.
Version	PDF 1.2 and later, PDF/A-1
Color depth	1-8, 24 bits per pixel
Compression type	Lossless
Compression algorithm	A lossless data compression algorithm that uses a combination of the LZ77 algorithm and Huffman coding.
Application area	Images

LZW

Developer	Abraham Lempel, Jacob Ziv, and Terry Welch's copyright-based issues, which expired in most countries in 2003/2004, reduced the popularity of this compression. As one of its consequences, it is not included in PDF/A standard.
Version	PDF 1.2 and later
Color depth	2-8 bits per pixel
Compression type	Lossless
Compression algorithm	An indexed based compression that is also used in the GIF and TIFF image formats.

Application area	Grayscale images, artificial images

CCITT Fax Group 3 and 4

Developer	International Telecommunications Union (ITU), formerly known as the Comité Consultatif International Téléphonique et Télégraphique
Version	PDF 1.0 and later, PDF/A-1
Color depth	1 bit per pixel
Compression type	Lossless
Compression algorithm	**Group3** 1-dimensional version of the CCITT Group 3 Huffman encoding algorithm. **Group 3 2D** 2-dimensional version of the CCITT Group 3 Huffman encoding algorithm. **Group 4** An advanced version of a bitonal algorithm based on the CCITT Fax Group 3 2D compression.
Application area	Line-art image, bitonal, faxes

JBIG2

Developer	Joint Bi-Level Image Experts Group
Version	PDF 1.4 and later, PDF/A-1
Color depth	1 bit per pixel
Compression type	Lossless
Compression algorithm	The image is broken down into individual symbols, which are stored in a table. A symbol is added to the table if it does not exist yet. If a matching symbol already exists, it is used as a reference. This algorithm works especially well for images with a lot of similar symbols such as scanned text or images that use patterns. Generally, JBIG2 provides a better compression ratio than CCITT Group 3 or Group 4 compression. Typical compression ratios for text pages are 20:1 to 50:1.
Application area	Line-art image, bitonal

JPEG2000

Developer	Joint Photographic Experts Group committee
Version	PDF 1.5 and later, PDF/A-2
Color depth	8, 24 bits per pixel
Compression type	Lossless if the image quality index is set to `100`. Lossy otherwise
Compression algorithm	JPEG2000 is a wavelet-based image compression standard. It was developed with the intention of superseding the original discrete cosine transform-based JPEG standard.
Application area	Sampled continuous-tone pictures (photographs)

5.2.2 Relevant factors for file size

The size of an image is basically determined by four factors:

Pixel mass The total amount of pixels the image has. An image with a size of 600 by 800 pixels has 480'000 pixels total.

Color depth Number of bits are required to describe 1 pixel. The table below gives the answer for different types of images. For example, an RGB image with 600 by 800 pixels requires therefore 600 x 800 x 3 bytes = 1.44 Mbytes in uncompressed format.

Color space	Description	Bits/Pixel
Bitonal	Black and white	1
Indexed	Colors are stored in an index table that usually holds 2 to 256 entries, e.g. GIF.	2-8
Grayscale	Monochrome	8
Color RGB	Color using Red, Green, Blue	24
Color CMYK	Color using Cyan, Magenta, Yellow, Key (=black)	32

Compression type A compression algorithm can compress data (such as an image) to reduce its file size. Such an algorithm belongs to either of the following two classes:

Lossless The original image can be restored exactly.

Lossy The compression modifies the pixels. The original image cannot be restored from the compressed version. This is typically applied to photographic images where the human eye cannot distinguish whether the image was modified. The most common lossy compression is JPEG. The benefit of lossy compression is the higher compression ratio.

See also Supported image compression types.

Content of the image The simpler the image, the better it compresses. For most compression algorithms, a simple image (e.g. completely white) compresses much better than a complex image (e.g. a photo).

Examples:

CCITT Fax compression was designed to compress black text written on a white background. The algorithm was optimized under the assumption that a page contains more white pixels than black pixels. Therefore a bitonal image with a lot of black does generally not compress as well as an image with more white even if they have the same pixel mass.

JBIG2 compression searches for patterns, and uses them multiple times. For example, in a scanned text document, the same few dozen of characters are used over and over again. The algorithm is optimized to save frequent patterns more efficiently than rare ones.

5.2.3 Provided features for optimizing images

The 3-Heights® PDF Optimizer Shell offers the following possibilities to optimize images:

Pixel mass You can reduce pixel mass. (It cannot be increased.) This is done by clipping (cropping) the image size to its visible extent and/or by reducing the image resolution.

The resolution defines how many pixels there are in given length of the image. The most common unit for resolution is DPI (dots per inch). If an image has a resolution of 200 DPI, it means when displayed at 100% zoom, there are 200 pixels for 1 inch of image. The higher the resolution, the "sharper" the image. A monitor has usually a resolution of at least 96 DPI; a laser printer of at least 600 DPI. When the file size matters, a common resolution for color and grayscale images in PDF is 150 DPI (usually higher for bitonal).

Re-sampling is the process of changing the amount of pixels in an image. Downsampling is when the result has less pixels than the original image.

In the 3-Heights® PDF Optimizer Shell, downsampling is applied by setting a target resolution and a threshold resolution. The default values are 150 DPI for the target resolution and 225 DPI for the threshold resolution. This means that every image with a resolution of 225 DPI or higher is potentially downsampled to 150 DPI. Of course, the threshold resolution can be set equal to the target resolution. However, there are many cases where downsampling by just a little bit has disadvantages. In particular, lossy images (e.g. JPEG compression) lose visual quality every time they are newly compressed. On top of that, the compressed output can be larger than the input because artifacts introduced by the previous compression(s) are now considered as part of the image that needs to be compressed and lead to a worse compression even when the resolution is reduced. By default, the 3-Heights® PDF Optimizer Shell prevents such unnecessary re-sampling.

Color depth You can modify color depth for color images. The color depth can be left unchanged, set to grayscale (8 bit), RGB (24 bit) or CMYK (32 bit). It cannot be changed to black and white (1 bit).

> **Note:** In certain circumstances, the color depth of the image is not converted, e.g. if the resulting file size increases or if the image is pre-blended with a matte color.

Color complexity You can reduce color complexity. "Color complexity" means the following hierarchy of possible image pixel contents:

1. All pixels have the same color.
2. All pixels are either black or white.
3. All pixels are colored gray.
4. Pixels have differing colors.

With color complexity reduction, images are converted to their lowest possible color complexity. For example, a color image with only black and white pixels is converted to a bitonal image. Furthermore, an image with color complexity 1 (single color) is downsampled to one pixel.

Color complexity reduction is also applied to masks and soft masks. Soft masks of complexity 2 (bitonal) are converted to masks. Masks and soft masks with complexity 1 (single color) whose color is such that the (soft) mask is opaque are removed.

> **Note:** Currently, color complexity reduction is only carried out for images that have a device color space (DeviceRGB, DeviceGray, or DeviceCMYK) or an indexed color space whose base color space is a device color space.

Compression You can set up compression independently for the following three image compression types:

Type	Description
Bitonal	Black and white images.
Indexed	Images with an indexed (also known as "paletted") color space.
Continuous	Color (RGB and CMYK) images and grayscale images.

Bitonal images usually contain text or black and white graphics. Indexed images usually contain color graphics such as logos. Continuous images usually contain photographs.

For each of the above image types, several compression algorithms can be set. The 3-Heights® PDF Optimizer Shell tries all the given compression algorithms and takes the one that yields the smallest file size. The more compression algorithms set, the longer the process of optimizing images takes.

Furthermore, a more conservative image processing strategy can be enabled. This strategy prevents all the compression trials if the image has neither been clipped nor downsampled nor undergone a color-conversion. Hence, if the image has not been altered, then the original image from the input document is taken.

Content of the image You cannot change the content directly. However, changing the resolution or applying a lossy compression algorithm modifies the content of the image.

> **Note:** Unless forcing of re-compression is enabled, the 3-Heights® PDF Optimizer Shell never increases the file size of an image because it chooses the smallest among all tried compression algorithms and the original image in the input file. This means the 3-Heights® PDF Optimizer Shell cannot be used to "un-compress" embedded images.

5.2.4 MRC optimization for images

Some raster images—typically scanned documents—consist mainly of text, possibly in several colors and interspersed with some pictures. Such images are difficult to compress with one single compression type because of the diverse or even conflicting features of different parts of the image.

> **Note:** There exists an optimization profile for MRC optimization. See Profile settings.

Mixed Raster Content (MRC) optimization is a way of breaking such images down into parts, such that each part is well suited for one type of a compression algorithm. With this approach, the resulting file size often can be reduced without significantly reducing the visual quality of the document.

Note:

- MRC optimization can only be enabled for continuous images, i.e. not for bitonal images and images with an indexed color space.
- Monochrome (grayscale) images are treated as entire photographic regions. See also Stage 1: Cutting out pictures.
- MRC optimization may yield unexpected results, e.g. because the input image is not suitable for MRC. As another example, images in the original PDF may be stored as small slices, and MRC optimization fails because the 3-Heights® PDF Optimizer Shell has no option to concatenate such image slices.
- A PDF that contains MRC-optimized images is not suited for optical character recognition (OCR) and image extraction.

In the 3-Heights® PDF Optimizer Shell, MRC optimization works in three stages:

Stage 1: Cutting out pictures

In this stage, the input image is analyzed and rectangular areas containing photographic features are detected. Each detected region is cut out and placed as a separate image in the resulting PDF.

Depending on the input image, it is possible that this stage decides that the whole input image consists of one photographic region covering the whole image. In this case, the second phase (Stage 2: Separating into layers) is omitted.

On the other hand, it is possible that actual photographic regions present in the input image are not recognized correctly. This can happen, for example, if a photographic region contains parts with uniform color.

For the cut-out images, a compression type can be set.

Note: The resulting cut pictures are neither downsampled nor color-converted. Monochrome (grayscale) images are always treated as entire photographic regions.

Stage 2: Separating into layers

For this second stage, the image is not supposed to contain any photographic features. Instead, the image is assumed to consist of text and graphic, potentially with varying color.

The whole image is separated into two layers, a foreground and a background layer. Additionally, a mask is created, which can be thought of as a bitonal image that is not displayed directly, but tells for each pixel whether to show the foreground layer or the background layer.

Example:

The image consists of a yellow background with black paragraph text and a title text in red. Then the resulting background layer contains the yellow color only. The foreground layer contains the black text color where the paragraph text is located and the red text color where the title is located. In the mask, pixels for which the foreground layer should be displayed are set to 1, the others are set to 0. I.e. the mask contains ones where the black and the red text is and zeros everywhere else.

In the resulting PDF, the foreground layer, the background layer, and the mask are stored as three images and thus are allowed to have different resolution and different compression types. Since all the detailed features have been

moved to the mask, it makes sense to downsample the foreground and background layers and use a low image quality. The mask, on the other hand, is usually stored with a lossless compression type, optimized for text.

Stage 3: Reconstructing the image

In this stage, the results of stage 1 (the cut-out images) and stage 2 (the layers and the mask) are used to synthesize the desired result. If a single photographic region covering the entire image is detected in stage 1, then the original image is used and the reconstruction is finished. Otherwise, the reconstruction first places the background layer, followed by the foreground layer with the mask. Finally, if any cut-images are found, they are placed at their respective locations on top of the foreground layer.

5.3 Optimizing fonts

Every text in a PDF document is written with a font. This font can either be embedded or not embedded in the resources of the PDF. Embedded means a font program is embedded that describes how glyphs are drawn. If a font is not embedded, the application rendering the PDF (e.g. 3-Heights® PDF Viewer or Adobe Acrobat) has to select a replacement font. Therefore, the visual appearance of text written with an embedded font is determinable, whereas it is not when the font is not embedded.

A font program can be quite large. An embedded font that contains all WinAnsi characters has a size of about 20-100 Kbytes. If it contains a large Unicode range (e.g. Asian characters), it can be several Mbytes. An non-embedded font requires much less.

This leads to the following ways to optimize fonts:

Remove the embedded font: Removing embedded fonts can reduce the file size of a document, particularly when the document contains many fonts. Removing fonts is best applied to (PDF-) standard fonts such as Arial, Courier, Courier New, Helvetica, Times, Times New Roman. Removing fonts should not be applied to barcode fonts or fancy types.

> **Note:** PDF/A requires fonts to be embedded.

Subset fonts: Only keep the information in the font program that is required to render the characters that are actually used in text in this document. All unused characters are removed.

Merge fonts: A document can have the same font, or a subset of it, embedded multiple times. This commonly occurs when multiple input documents are merged into one large output document. The 3-Heights® PDF Optimizer Shell Tool can merge these fonts into one font (if they can be merged).

6 Interface reference

6.1 Profile settings

The following table lists all settings for optimization all profiles. The profile labeled "(default)" contains the settings in effect when no profile is selected.

> **Note:** Values in parentheses, although set, have no effect because images are excluded from processing or image downsampling is disabled due to a threshold set to `-1`.

> **Note:** The license feature `Color` does support none of the following profiles, but the default profile.

Profile settings

	(default)	web	print	archive	max	mrc
Compression types for bitonal images (`-fb`):						
Exclude (`-1`)	✓					✓
CCITT Group 4 (6)		✓	✓	✓	✓	
JBIG2 (7)		✓		✓	✓	
Source (`10`)		✓	✓	✓	✓	
Compression types for continuous images (`-fc`):						
Exclude (`-1`)	✓					
JPEG (1)		✓	✓	✓	✓	
Flate (2)		✓	✓	✓	✓	
JPEG2000 (8)		✓		✓	✓	
MRC (9)						✓
Source (`10`)		✓	✓	✓	✓	
Compression types for indexed images (`-fi`):						
Exclude (`-1`)	✓					✓
Flate (2)		✓	✓	✓	✓	

Profile settings

	(default)	web	print	archive	max	mrc
LZW (3)					✓	
Source (10)		✓	✓	✓	✓	
Resolution for bitonal images (Resolution values per image type)	(200)[1]	200	(200)[1]	(200)[1]	160	(200)[1]
Threshold for bitonal images (Threshold values per image type)	-1	280	-1	-1	220	-1
Resolution for monochrome images (Resolution values per image type)	(150)[1]	150	(150)[1]	(150)[1]	130	(150)[1]
Threshold for monochrome images (Threshold values per image type)	-1	210	-1	-1	180	-1
Resolution for color images (Resolution values per image type)	(150)[1]	150	(150)[1]	(150)[1]	130	(150)[1]
Threshold for color images (Threshold values per image type)	-1	210	-1	-1	180	-1
Image quality (-q)	75	75	80	80	70	75
Color conversion (-c)	0	1	2	0	1	1
Clip images (-oc)		✓	✓	✓	✓	✓
Reduce color complexity (-rc)		✓	✓	✓	✓	
Force re-compression (-ff)						
Force compression types (-ft)						
Dithering mode (-h)	0 (Floyd Steinberg)					
MRC layer compression (-ml)	8 (JPEG2000)					
MRC layer resolution (-mlr)	70					
MRC layer quality (-mm)	6 (CCITT Fax Group 4)					
MRC layer quality (-mlq)	20					
MRC cut picture compression (-mp)	1 (JPEG)					
Convert fonts to CFF (-cff)		✓	✓	✓	✓	
Merge font programs (-m)		✓	✓	✓	✓	✓

Profile settings

	(default)	web	print	archive	max	mrc
Remove standard fonts (`-rs`)					✓	
Subset font programs (`-s`)		✓	✓	✓	✓	✓
Optimize resources (`-od`)		✓	✓	✓	✓	✓
Linearize (`-ow`)						
Linearize (`-owa`)		✓				
Remove redundant objects (`-or`)		✓	✓	✓	✓	✓
Strip the file:						
Article threads (Strip the file)		✓	✓		✓	✓
Metadata (Strip the file)		✓			✓	
Piece info (Strip the file)		✓	✓		✓	✓
Document structure (Strip the file)		✓	✓		✓	✓
Thumbnails (Strip the file)		✓	✓	✓	✓	✓
Spider (Strip the file)		✓	✓		✓	✓
Alternates (Strip the file)		✓		✓	✓	
Outlines (Bookmarks) (Strip the file)					✓	
Other annotations (Strip the file)					✓	
Form fields (Strip the file)					✓	
Link annotations (Strip the file)						

6.2 Optimization options

An option is a configuration string of the form

```
-‹option› ‹parameters›
```

where ‹`option`› is a 1 to 3-letter string that names the option and ‹`parameters`› are further configuration values given to this options. Many options don't support any ‹`parameters`›.

[1] These values, although set, have no effect because downsampling of images is disabled.

Options are provided on the command line after the command pdfoptimize. They define how the document should be optimized. The last two strings of the command line should always be the input and the output file. (There is no output file required when using any of the listing-options `-li` and `-lf`.)

Options are parsed from left to right, the last set value is applied. (An exception to this is setting a profile with `-pr`: Profiles are always applied first.)

Example: With the following options, the resolution for re-sampling of all raster image types (color, monochrome, bitonal) is first set to 100, then the monochrome resolution is set explicitly to 120.

```
pdfoptimize -dr 100 -dmr 120 input.pdf output.pdf
```

In the above command, if the setting `-dmr 120` was set before `-dr 100`, it would not have any influence, since `-dr 100` applies to all compressions and therefore would overwrite the previous setting.

In the following, all options are listed in alphabetical order.

Options that are marked with a license features are available if the listed feature is included in the license. See also License features.

6.2.1 -c Set the color conversion

Set the color conversion `-c ‹n›`
License feature: `Color`

This option activates conversion of raster images from one color space into another, e.g. convert all RGB images to CMYK images.

This option does not have any impact on objects other than raster images that use color spaces, such as vector graphics or text. Color key masked images are not color converted. Pre-blended images can be converted from RGB to grayscale.

This option is affected when setting a profile (`-pr`).

The parameter `‹n›` has the following meaning:

Color conversions

‹n›	Conversion	Color values
0	(default) Don't convert colors	
1	Convert to ICE sRGB colors	red, green, blue
2	Convert to CYMK color (using profiles)	cyan, yellow, magenta, key
3	Convert color images to grayscale	gray

Example: To convert all embedded color images that use the RGB color space to images of the CMYK color space, use the following command

```
pdfoptimize -c 2 input.pdf output.pdf
```

6.2.2 -cff Compress Type1 fonts (convert to CFF)

Compress Type1 fonts (convert to CFF) `-cff`
License feature: `Optimize`

Convert embedded Type1 (PostScript) fonts to Type1C (Compact font format). This reduces the file size.

This option is affected when setting a profile (`-pr`).

6.2.3 -cms Set the color management engine

Set the color management engine `-cms ‹engine›`
License feature: `Color`

The transformation of colors from one color space to another is performed using a color management engine.

Supported engines are:

none The algorithms specified in the PDF reference are used. This results in the maximum possible contrast.

neugebauer The Neugebauer algorithm efficiently converts CMYK to RGB. It does not need any color profiles. The results look similar to conversion using color profiles.

lcms (default): Use ICC color profiles. Default profiles are used for all unmanaged device color spaces as described in Color profiles.

‹FileName› When providing a file name, a configurable version of the Neugebauer algorithm is applied. The coefficients can be defined in the text file. The default Neugebauer coefficients are listed below (Red, Green, Blue; Color):

```
1.000000, 1.000000, 1.000000; White
0.000000, 0.682353, 0.937255; C
0.925490, 0.000000, 0.549020; M
1.000000, 0.949020, 0.000000; Y
0.137255, 0.121569, 0.125490; K
0.180392, 0.188235, 0.572549; CM
0.000000, 0.650980, 0.313725; CY
0.000000, 0.054902, 0.137255; CK
0.929412, 0.109804, 0.141176; MY
0.137255, 0.000000, 0.000000; MK
0.105882, 0.098039, 0.000000; YK
0.211765, 0.211765, 0.223529; CMY
0.000000, 0.000000, 0.003922; CMK
0.000000, 0.070588, 0.000000; CYK
0.133333, 0.000000, 0.000000; MYK
0.000000, 0.000000, 0.000000; CMYK
```

The Neugebauer algorithm mixes the colors based on the amount of color and the corresponding weighted coefficient. Altering the values for a pure color specifically changes the result for this pure color.

The color transition remains smooth.

Example: The following command selects the Neugebauer color management engine

```
pdfoptimize -cms neugebauer input.pdf output.pdf
```

6.2.4 Resolution values per image type

DPI for bitonal images `-dbr ‹dpi›`
License feature: `Optimize`

DPI for color images `-dcr ‹dpi›`
License feature: `Optimize`

DPI for monochrome images `-dmr ‹dpi›`
License feature: `Optimize`

The target resolution values for downsampling images can be set individually for different types of images.

This option is affected when setting a profile (`-pr`).

Parameter:

`‹dpi›` The target resolution in DPI. The default values for `‹dpi›` are as follows:

Bitonal images `200`

Color images `150`

Monochrome images `150`

6.2.5 Threshold values per image type

DPI for bitonal images `-dbt ‹dpi›`
License feature: `Optimize`

DPI for color images `-dct ‹dpi›`
License feature: `Optimize`

DPI for monochrome images `-dmt ‹dpi›`
License feature: `Optimize`

The threshold values above which downsampling an image is activated can be set with these options.

This option is affected when setting a profile (`-pr`).

Parameter:

`‹dpi›` The threshold in DPI. A value of `-1` indicates that all images of this type are excluded from downsampling.

Default: `-1`.

6.2.6 -dr Resolution in DPI

Resolution in DPI -dr ‹dpi›
License feature: Optimize

Set the target resolution after re-sampling images, image masks and image's soft masks in dots per inch (DPI). Only those images with a resolution value higher than the threshold value, which is set with -dt option are processed. The default target resolution is 150 DPI.

This option is affected when setting a profile (-pr).

Pre-blended images, images with a color key mask, masks, and soft mask images are not re-sampled.

Example: To downsample all raster images with a resolution greater than 150 DPI to 75 DPI, apply the following

```
pdfoptimize -dt 150 -dr 75 input.pdf output.pdf
```

6.2.7 -dt Threshold in DPI

Threshold in DPI -dt ‹dpi›
License feature: Optimize

This option defines the minimum resolution an image must have to be optimized. The threshold value for downsampling raster images is used in conjunction with the -dr option, which sets the actual target resolution for the downsampled images.

This option is affected when setting a profile (-pr).

The threshold resolution must be equal or higher than the target resolution. If the value is set to -1, downsampling is turned off. This is the default.

Example: Select the "web" profile and downsample all raster images with an original resolution higher or equal to 150 DPI to a new resolution of 75 DPI.

```
pdfoptimize -pr web -dt 150 -dr 75 input.pdf output.pdf
```

Example: Select the "web" profile but disable downsampling of images

```
pdfoptimize -pr web -dt -1 input.pdf output.pdf
```

If the size (in terms of bytes) of the re-sampled image is larger than its original size, the original image is kept instead.

6.2.8 -fb Compression types for bitonal images

Compression types for bitonal images -fb ‹compr›

This option affects only bitonal (black and white) images. The `-fb` option is followed by a comma-separated list of numerical values (no spaces allowed in the list). The following values are possible:

This option is affected when setting a profile (`-pr`).

Bitonal compression

Value	Compression filter
`0`	RAW data
2	Flate (ZIP) compression
3	Lempel-Ziv-Welch (LZW) compression
4	CCITT Fax Group 3 compression
5	CCITT Fax Group 3 2D compression
6	(default) CCITT Fax Group 4 compression
7	JBIG2 compression
`10`	Take the compression type from the original image in the input PDF
`-1`	Exclude bitonal images from processing

Example: To let the 3-Heights® PDF Optimizer Shell try CCITT Group 3 compression, JBIG2 compression, and the compression that is used in the source image of the original file, use the following command

```
pdfoptimize -fb 3,7,10 input.pdf output.pdf
```

The above command makes the 3-Heights® PDF Optimizer Shell go through all bitonal images and process each image individually as follows. All the given compression algorithms are executed. If the input image has a compression different from CCITT Fax Group 3 and JBIG2, then the compression of the input image is also executed. As a result, several candidate versions are obtained. Now a choice is made among all these versions (including the original image) based on the size in bytes. The smallest candidate is chosen and used in the output document.

6.2.9 `-fc` Compression types for color and grayscale images

Compression types for color and grayscale images `-fc ‹compr›`

This option affects normal color images (RGB and CMYK) and grayscale (monochrome) images. The `-fc` option is followed by a comma-separated list of numerical values (no spaces allowed in the list). The following values are possible:

This option is affected when setting a profile (`-pr`).

Color/Monochrome compression

Value	Compression filter	License feature
0	RAW data	Optimize, Color
1	(default) DCT(JPEG) compression	Optimize, Color
2	Flate (ZIP) compression	Optimize, Color
8	JPEG2000 compression	Optimize, Color
9	Perform MRC optimization (See MRC optimization for images)	Optimize
10	Take the compression type from the original image in the input PDF	Optimize, Color
-1	Exclude continuous images from processing	Optimize, Color

Example: To let the 3-Heights® PDF Optimizer Shell try JPEG compression, JPEG2000 compression, and the compression that is used in the source image of the original file, use the following command.

```
pdfoptimize -fc 1,8,10 input.pdf output.pdf
```

The above command makes the 3-Heights® PDF Optimizer Shell go through all color and grayscale images, and process each image individually as follows. All the given compression algorithms are executed. If the input image has a compression different from JPEG and JPEG2000, then the compression of the input image is also executed. As a result, several candidate versions are obtained. Now a choice is made among all these versions and the original image based on the size in bytes. The smallest candidate is chosen and used in the output document.

6.2.10 -ff Force re-compression

Force re-compression -ff
License feature: Optimize

If this option is set, then images are always re-compressed, i.e., the original image is never used as a candidate for inclusion in the output document. If not set (default), then images are only re-compressed if the resulting image is smaller than the original, i.e. occupies less bytes to store in the file.

This option is affected when setting a profile (-pr).

6.2.11 -fi Compression types for indexed (paletted) images

Compression types for indexed (paletted) images -fi ‹compr›

This affects only images with an indexed color space. This type of color space is sometimes used for color graphics and logos. The -fc option is followed by a comma-separated list of numerical values (no spaces allowed in the list). The following values are possible:

This option is affected when setting a profile (-pr).

Indexed compression types

Value	Compression filter
0	RAW data
2	Flate (ZIP) compression
3	Lempel-Ziv-Welch (LZW) compression
10	Take the compression type from the original image in the input PDF

Example: To let the 3-Heights® PDF Optimizer Shell try Flate compression and LZW compression, use the following command:

```
pdfoptimize -fi 2,3 input.pdf output.pdf
```

The above command makes the 3-Heights® PDF Optimizer Shell go through all images with indexed color space and process each image individually as follows. All the given compression algorithms are executed. As a result, two candidate versions are obtained. Now a choice is made among these two versions and the original image based on the size in bytes. The smallest candidate is chosen and used in the output document.

6.2.12 -fn File name

File name `-fn ‹file.pdf›`

The intension of this option is to provide support for file names that start with a dash character and would therefore cause a parameter error.

The parameter after the `-fn` option is a file name. It can optionally also be used for file names not starting with a dash character.

Example:

```
pdfoptimize -fn -input.pdf output.pdf
```

6.2.13 -ft Force compression types

Force compression types `-ft`
License feature: `Optimize`

If this option is set, then re-compression of images is forced if an image in the input PDF has a compression type that differs from the compression types given in `-fb`, `-fc`, or `-fi`. Use this option if you want to allow only the given compression types for images in the output PDF.

This option is affected when setting a profile (`-pr`).

6.2.14 -fv Minimum PDF version

Minimum PDF version `-fv ‹1.x›`

This option allows to set the minimum PDF version of the created PDF output file. Supported values are 1.1 to 1.7 and 2.0. (PDF 1.4 corresponds to Acrobat 5, PDF 1.5 to Acrobat 6, etc.) There are three parameters that influence the version of the PDF output file:

- The value set using the option `-fv`.
- The PDF version of the input file.
- Other settings in the optimization (JBIG2 requires PDF 1.4, JPEG2000 requires PDF 1.5) The maximum of the three values above sets the PDF version in the output file.

Example: 1. Input PDF is version 1.5 and the following setting is used

```
pdfoptimize -fv 1.4 input.pdf output.pdf
```

The output file is PDF version 1.5.

Example: 2. Input PDF is version 1.4 or lower and the following setting is used

```
pdfoptimize -fv 1.4 input.pdf output.pdf
```

The output file is PDF version 1.4.

Example: Input PDF is version 1.3 and the following setting is used

```
pdfoptimize -fv 1.4 -fc 8 input.pdf output.pdf
```

If input.pdf contains color images to which JPEG2000 compression is applied, the output file is version 1.5. Otherwise, it is version 1.4.

6.2.15 -h Dithering mode for bitonal images

Dithering mode for bitonal images `-h ‹mode›`

This option enables or disables dithering when downsampling bitonal images.

This option is affected when setting a profile (`-pr`).

‹mode›	Description
0	No dithering
1	Floyd-Steinberg dithering algorithm

Some bitonal images try to evoke the impression of different levels of gray by randomly setting pixels to black. If dithering is applied during downsampling, then the gray levels of such images are preserved better. If dithering is switched off, then lines (e.g. text glyphs) are preserved better.

6.2.16 `-id` Set value in the document information dictionary

Set value in the document information dictionary `-id ‹key› ‹value›`

Set the value of an document information dictionary entry `‹key›`. Popular entries specified in the PDF Reference 1.7 are `"Title"`, `"Author"`, `"Subject"`, `"Creator"` (sometimes referred to as Application), and `"Producer"` (sometimes referred to as PDF Creator). If the entry already exists then the previous entry is overwritten. If the key corresponds to a standard metadata key, then the XMP metadata is updated accordingly.

Example: Overwrite the default producer:

```
pdfoptimize -id Producer "MyProgram 1.2" input.pdf output.pdf
```

6.2.17 `-lf` List fonts

List fonts `-lf`
License feature: `Optimize`

List all fonts and their properties.

List fonts

Parameter	Description	Example
`FontName`	The name of the font. Subsetting-prefixes are not listed as name of the font.	"Arial-BoldMT", "Verdana"
`FontType`	The Font Type	TrueType, Type1
`Encoding`	The encoding of the font, see examples.	Difference Encoding, IntrinsicEncoding, MacRomanEncoding, SymbolEncoding, WinAnsiEncoding
`IsCID`	Whether the font is a CID font (Character Identifier Font) or not.	CID, Non-CID
`IsEmbedded`	Whether the font has an embedded font program or not.	Embedded, Non-embedded
`IsSubsetted`	Whether a font program is subsetted or not. This value is only set for fonts, which have an embedded font program.	Subsetted, Non-Subsetted
`FileName`	The file name of the font program. This is the name under which the font is saved to file in case the option `-xf` is applied. For all non-embedded fonts, there is no file name available (N/A).	`fnt12.ttf`, `fnt2477.cff`, N/A

Example: The following command lists all fonts of a PDF document

```
pdfoptimize -lf input.pdf
FontName, FontType, Encoding, IsCID, IsEmbedded, IsSubsetted, Filename
"Arial-BoldMT", TrueType, MacRomanEncoding, Non-CID, Non-embedded, N/A,
"Arial-BlackItalic", TrueType, MacRomanEncoding, Non-CID, Non-embedded, N/A,
"Verdana", TrueType, WinAnsiEncoding, Non-CID, Embedded, Subsetted, fnt38.ttf
```

The first line in the above example is the actual command, the following lines list the output.

See also `-xf` option for extracting fonts.

6.2.18 `-li` List images

List images `-li`

List all images and their properties.

List images

Parameter	Description	Example
ObjectNumber	The PDF object number	9
Width	The width of the image in pixel.	400
Height	The height of the image in pixel.	589
BitsPerComponent	The number of bits that are used to represent one component. This number is in most cases either 1 (bitonal) or 8 (RGB, CMYK, Gray).	8
ColorSpace	The color space of the image.	DeviceCMYK, DeviceRGB, DeviceGray, ICCBased, Indexed
Resolution	The resolution in dots per inch (DPI).	96
Filter	The compression filter.	DCTDecode, FlateDecode
ImageSize	The uncompressed image size.	706800
CompressedSize	The compressed image size.	28172
CompressionRatio	The ratio compressed image size divided by uncompressed images size. The smaller this value, the higher the compression.	3.99%
FileName	The file name of the image. This is the name under which the image is saved to file in case the option `-xi` is applied.	img9.tif

Example: The following command lists all images in the file input.pdf. In this case, there is one image.

```
pdfoptimize -li input.pdf
ObjectNumber, Width, Height, BitsPerComponent, ColorSpace, Resolution,
Filter, ImageSize, CompressedSize, CompressionRatio, FileName
9, 400, 589, 8, ICCBased, 96, DCTDecode, 706800, 28172, 3.99%, img9.tif
```

See also `-xf` option for extracting fonts.

6.2.19 `-lk` Set license key

Set license key `-lk ‹key›`

Pass a license key to the application at runtime, instead of using one that is installed on the system.

```
pdfoptimize -lk X-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX ...
```

This is required in an OEM scenario only.

6.2.20 -m Merge embedded font programs

Merge embedded font programs `-m`
License feature: `Optimize`

Font programs can be merged if they originate from the same font, e.g. they are of the same type, have the same name and encoding. Merging of Type1 (PostScript) and TrueType fonts is supported.

This option is affected when setting a profile (`-pr`).

6.2.21 -ml Compression type for MRC layers

Compression type for MRC layers `-ml ‹compr›`
License feature: `Optimize`

This option affects only MRC foreground and background layers. The option is followed by a single numerical value indicating the compression type to use for MRC foreground and background layers. For possible values, see Color/Monochrome compression. The default is 8 (JPEG2000 compression).

See also MRC optimization for images.

This option is affected when setting a profile (`-pr`).

6.2.22 -mlq Image quality for MRC layers

Image quality for MRC layers `-mlq ‹q›`
License feature: `Optimize`

This option affects only MRC foreground and background layers. The option is followed by a numerical value between 0 and 100 to be used as the image quality for MRC foreground and background layers when using a lossy compression for these layers. The default is 10.

See also MRC optimization for images.

This option is affected when setting a profile (`-pr`).

6.2.23 -mlr Resolution in DPI for MRC layers

Resolution in DPI for MRC layers `-mlr ‹dpi›`
License feature: `Optimize`

This option affects only MRC foreground and background layers. The option is followed by a numerical value that indicates the target resolution in DPI of MRC layers after downsampling. The default is 70.

See also MRC optimization for images.

This option is affected when setting a profile (`-pr`).

6.2.24 -mm Compression type for the MRC mask

Compression type for the MRC mask `-mm ‹compr›`
License feature: `Optimize`

This option affects only MRC masks. The option is followed by a single numerical value indicating the compression type to use for MRC masks. For possible values, see Bitonal compression. The default is 6 (CCITT Fax Group 4 compression).

See also MRC optimization for images.

This option is affected when setting a profile (-pr).

6.2.25 -mp Compression type for MRC cut-out pictures

Compression type for MRC cut-out pictures -mp ‹compr›
License feature: Optimize

This option affects only cut-out images and entire monochrome (grayscale) images when doing MRC optimization. The option is followed by a single numerical value indicating the compression type to use for MRC cut-out pictures. For possible values, see Color/Monochrome compression. The default is 1 (JPEG compression).

See also MRC optimization for images.

This option is affected when setting a profile (-pr).

6.2.26 -o Owner password

Owner password -o ‹owner›

The owner password is required to change the security settings of the document. To apply permission flags, an owner password must be set. Permission flags are set with the -p switch.

Example: Encrypt a document and set the owner password to ‹owner›.

```
pdfoptimize -o owner input.pdf output.pdf
```

6.2.27 -oc Clip images

Clip images -oc
License feature: Optimize

Images in PDF documents can be clipped. This means that only part of the image is visible, while the rest is hidden. The -oc option detects these images, reduces their size the area that is actually displayed, and replaces the original image by the reduced image. Pre-blended images are not clipped.

Setting -oc activates the -od option.

This option is affected when setting a profile (-pr).

6.2.28 -od Optimize resources

Optimize resources -od
License feature: Optimize

Optimize the resources of the PDF such as images, color spaces, or fonts. If set, unused resources are removed. Also content streams are re-built.

This option is affected when setting a profile (-pr).

6.2.29 -ol Linearize only

Linearize only -ol
License feature: Optimize

Note: With this option enabled, non-Latin characters in the input and output file name are not supported.

Do not apply any optimizations, but linearize the file. This can be significantly faster than the -ow option. See -ow for more information.

When this option is set, then all other options are disabled except -o, -u, -pw, and -p.

6.2.30 -or Remove redundant objects

Remove redundant objects -or
License feature: Optimize

This option removes redundant PDF objects. It identifies duplicates objects in the input document and collapses them into single objects in the output document.

This option is affected when setting a profile (-pr).

6.2.31 -ow Optimize for the web

Optimize for the web -ow
License feature: Optimize

Note: This option has no effect when combined with -owa.

Note: With this option enabled, non-Latin characters in the output file name are not supported.

Linearize the PDF output file, i.e. optimize file for fast web access.

The 3-Heights® PDF Optimizer Shell does not support linearization of PDF 2.0 documents. For such documents, processing fails. In order to automatically disable linearization for PDF 2.0 use -owa.

A linearized document has a slightly larger file size than a non-linearized file and provides the following main features:

- When a document is opened in a PDF viewer of a web browser, the first page can be viewed without downloading the entire PDF file. In contrast, a non-linearized PDF file must be downloaded completely before the first page can be displayed.

- When another page is requested by the user, that page is displayed as quickly as possible and incrementally as data arrives, without downloading the entire PDF file.

The above applies only if the PDF viewer supports fast viewing of linearized PDFs.

> **Note:** To use a linearized PDF file, the PDF must reside as a "file" on the web server. It must not be streamed.

When enabling this option, then no PDF objects are stored in object streams in the output PDF. For certain input documents, this can lead to a significant increase of file size.

6.2.32 `-owa` Optimize for the Web automatically

Optimize for the Web automatically `-owa`
License feature: `Optimize`

> **Note:** With this option enabled, non-Latin characters in the output file name are not supported.

Automatically decide whether to linearize the PDF output file for fast web access.

Applying linearization can lead to a large increase in file size for certain documents. Enabling this option lets the 3-Heights® PDF Optimizer Shell automatically apply linearization or refrain from doing so based on the estimated file size increase.

With this option enabled, PDF 2.0 documents are automatically excluded from linearization.

See also `-ow` for more information for linearized PDFs.

> **Note:** When `-owa` is given, then the `-ow` option has no effect.

6.2.33 `-p` Permission flags

Permission flags `-p ‹flags›`

This option sets the permission flags. It is only usable when producing encrypted documents. In other words, at least an owner password must be set with `-o`, and additionally a user password can be set with `-u`. When omitting the `-p` option, then all permissions are granted. The permissions that can be granted are listed below.

Permission flags

Flag	Description
p	Allow printing (low resolution)
m	Allow change of the document

Permission flags

c	Allow content copying or extraction
o	Allow commenting
f	Allow filling of form fields
s	Allow content extraction for accessibility
a	Allow document assembly
d	Allow high quality printing
i	Set the same permissions as in the input file
0	Allow nothing (no permissions are granted)

The actual ‹flags› given to this option is a string that contains one or several of the permission flags above.

Note: The i and 0 values cannot be combined with any other permission flags.

Example: The following command sets the owner password to "owner" and the permission flags to "allow printing in low resolution" and "allow form filling".

```
pdfoptimize -o owner -p pf input.pdf output.pdf
```

Example: "High quality printing" requires the standard printing flag to be set too.

```
pdfoptimize -o owner -p pd input.pdf output.pdf
```

Example: Create a document with the same permission settings as present in the input document.

```
pdfoptimize -o owner -p i input.pdf output.pdf
```

For further information about the permission flags, see PDF Reference 1.7 Section 3.5.2.

6.2.34 -pr Set an optimization profile

Set an optimization profile -pr ‹profile›
License feature: Optimize

With this option, one of the predefined optimization profiles can be set. Profile settings tabulates the configuration values for all profiles. If a profile is set, then all the tabulated configuration parameters are set to their respective values. Configuration parameters not listed in this table are left unchanged.

Parameter:

`‹profile›` The profile. Currently, the following profiles can be selected:

`web` Optimization for the Internet.

`print` Optimization for print.

`archive` Optimization for archiving purposes.

`max` Optimization for maximal memory size reduction.

`mrc` MRC (Mixed Raster Content) optimization of images. See MRC optimization for images.

One way of quickly arriving at a specific setting is to set a profile and adapt individual configuration parameters.

Example: Use the "web" profile, but inhibit the downsampling of bitonal images.

```
pdfoptimize -pr web -dbt -1 input.pdf output.pdf
```

6.2.35 `-pw` Read an encrypted PDF file

Read an encrypted PDF file `-pw ‹password›`

A PDF document that has a user password (the password to open the document) can only be processed when either the user or the owner password is provided. The password can be provided using the option `-pw` followed by the password.

Example: The input PDF document is encrypted with a user password. Either the user or the owner password of the input PDF is "mypassword". The command to process such an encrypted file is:

```
pdfoptimize -pw mypassword input.pdf output.pdf
```

When a PDF is encrypted with a user password and the password is not provided or is incorrect, the 3-Heights® PDF Optimizer Shell cannot read and process the file. Instead it generates the following error message:

```
Password wasn't correct.
```

6.2.36 `-q` Compression quality

Compression quality `-q ‹quality›`
License feature: `Optimize`

Set the compression quality index for lossy compression methods. This option only applies to JPEG, JPEG2000, and JBIG2 images. A lower value results in a smaller file size, but the images are of poorer visual quality. A higher value results in better visual quality, but also a larger file size.

This option is affected when setting a profile (`-pr`).

The supported values range from 1 (lowest) to 100 (highest). The default is 75. For image compressions that support lossless compression (JPEG2000), a value of 100 corresponds to lossless compression. Any other value represents lossy compression. For JBIG2, compression is always lossless irrespective of the quality index set. JPEG compression is always lossy.

Example: The following selects the "web" optimization profile and sets the quality index to 50. All images types that support the quality parameter are recompressed with this quality index.

```
pdfoptimize -pr web -q 50 input.pdf output.pdf
```

6.2.37 -rc Reduce image color complexity

Reduce image color complexity `-rc`
License feature: `Color`

This option is used to enable color complexity reduction of images. (See also Provided features for optimizing images.)

If enabled, then images with device color spaces (DeviceRGB, DeviceCMYK, or DeviceGray) and indexed images with a device color space as base color space are analyzed and if possible, converted as follows:

This option is affected when setting a profile (-pr).

- An image with DeviceRGB or DeviceCMYK color space in which all pixels are gray is converted to a grayscale image with DeviceGray color space.
- An image that contains only black and white pixels is converted into a bitonal image.
- An image in which all the pixels have the same color is downsampled to one pixel.

Furthermore, image masks and soft masks are optimized as follows:

- A soft mask that contains only black and white pixels is converted to a mask.
- A (soft) mask that is opaque is removed.

Color complexity reduction is disabled for those images whose compression type is set to -1. See -fc and -fi.

6.2.38 -ri Remove images

Remove images `-ri`
License feature: `Optimize`

When enabling this option, then images and stencil masks are substituted by empty form XObjects. Inline images are left untouched. All other image processing options are thus irrelevant.

Warning: Enabling this option usually alters the visual appearance of the document significantly.

6.2.39 -rf Remove embedded font program

Remove embedded font program `-rf ‹font›`
License feature: `Optimize`

This option makes the 3-Heights® PDF Optimizer Shell remove the embedded font program for the given ‹font›. This option can be specified several times to remove several font programs.

> **Warning:** The output document may not display correctly on certain systems.

6.2.40 -rs Remove embedded standard fonts

Remove embedded standard fonts -rs
License feature: Optimize

This option enables the removal of the font programs of all embedded standard fonts such as Arial, Courier, CourierNew, Helvetica, Symbol, Times, TimesNewRoman, and ZapfDingbats. (A complete list is given below.) The fonts are replaced with one of the 14 PDF standard fonts, all of which have no associated font program. Un-embedding a font decreases the file size.

This option is affected when setting a profile (-pr).

A PDF Viewer must be able to display these 14 PDF standard fonts correctly. Therefore, using this option usually should not visually alter the PDF when it is displayed.

Un-embedding the font works based on the font's Unicode information. In other words, the un-embedded font's characters are mapped to those of the original font with the same Unicode. Therefore, only fonts with Unicode information are un-embedded by the 3-Heights® PDF Optimizer Shell. However, if a font's Unicode information is not correct, un-embedding may lead to visual differences. Whether or not a font's Unicode information is correct can be verified by extracting text that uses the font. Suitable tools for this purpose are, for instance, the 3-Heights® PDF Extract Tool or an interactive PDF viewer.

If the extracted text is meaningful, the font's Unicode information is correct and unembedding of the font does not to visual differences.

List of candidate font base names for removing

- Arial
- Arial,Bold
- Arial,BoldItalic
- Arial,Italic
- Arial-Bold
- Arial-BoldItalic
- Arial-BoldItalicMT
- Arial-BoldMT
- Arial-Italic
- Arial-ItalicMT
- ArialMT
- Courier
- Courier,Bold
- Courier,BoldItalic
- Courier,BoldOblique
- Courier,Italic
- Courier,Oblique
- Courier-Bold
- Courier-BoldOblique
- Courier-Oblique
- CourierNew
- CourierNew,Bold
- CourierNew,BoldItalic
- CourierNew,Italic
- CourierNew-Bold
- CourierNew-BoldItalic
- CourierNew-Italic
- CourierNewPS-BoldItalicNT
- CourierNewPS-BoldMT
- CourierNewPS-ItalicMT
- CourierNewPSMT
- Helvetica
- Helvetica,Bold
- Helvetica,BoldItalic
- Helvetica,BoldOblique
- Helvetica,Italic
- Helvetica,Oblique
- Helvetica-Bold
- Helvetica-BoldItalic
- Helvetica-BoldOblique
- Helvetica-Italic
- Helvetica-Oblique
- Symbol
- SymbolMT
- Times,Bold
- Times,BoldItalic
- Times,Italic
- Times-Bold
- Times-BoldItalic
- Times-Italic
- Times-Roman
- TimesNewRoman
- TimesNewRoman,Bold
- TimesNewRoman,BoldItalic
- TimesNewRoman,Italic
- TimesNewRoman-Bold
- TimesNewRoman-BoldItalic
- TimesNewRoman-Italic
- TimesNewRomanPS
- TimesNewRomanPS-Bold

- TimesNewRomanPS-BoldItalic
- TimesNewRomanPS-BoldItalicMT
- TimesNewRomanPS-BoldMT
- TimesNewRomanPS-Italic
- TimesNewRomanPS-ItalicMT
- TimesNewRomanPSMT
- ZapfDingbats

6.2.41 -s Subset fonts

Subset fonts -s
License feature: Optimize

Embedded fonts can be subsetted. Subsetting refers to only storing those character glyphs of the font that are actually used. Unused character glyphs are removed. The advantage is that the file size can be reduced this way (in particular for Asian fonts).

This option is affected when setting a profile (-pr).

The downside is that if text is to be edited, only the characters of the subsetted font can be used.

6.2.42 Strip the file

Remove certain elements of the PDF file. See PDF Reference 1.7 for more details on these elements.

This option is affected when setting a profile (-pr).

The following parts of a PDF can be stripped:

Strip article threads -sa

Flatten and strip form fields and annotations -sf

Flatten and strip form fields -sff

Flatten and strip link annotations -sfl

Flatten and strip other annotations -sfa

Strip invisible annotations -sia

Strip alternate images (variant representations of the base image) -si

Strip the document's output intents -so

Strip meta data -sm

Strip page piece info (private application data) -sp

Strip document structure tree (incl. markup) -ss

Strip embedded thumbnails -st

Strip spider (web capture) info -sw

Strip everything (all of the above) -se

-sf implies -sfa, -sff, and -sfl. Likewise, -se implies all other strip options listed.

6.2.43 -sfs Flatten appearances of signature fields

Flatten appearances of signature fields -sfs
License feature: Optimize

A signature in a PDF consist of two parts:

a. The invisible digital signature in the PDF.
b. The visual appearance that was attributed to the signature.

Part (a) can be used by a viewing application to verify that a document has not changed since it has been signed and report this to the user. Part (b) is merely a "decorative" element on the page without further significance.

When optimizing a PDF, the PDF is altered and hence the digital signature is broken. Therefore, the 3-Heights® PDF Optimizer Shell removes all signatures, including parts (a) and (b).

If the `sfs` option is set, then digital signatures (parts (a)) are still removed, but their visual appearances (parts (b)) are flattened. In other words, the visual appearance is retained and drawn as non-editable graphic onto the page.

> **Note:** The resulting PDF can be misleading as it visually appears to be signed, but it has no digital signature and hence, a viewer application does not report any broken signature. In most cases, such a behavior is undesirable.

6.2.44 `-u` User password

User password `-u ‹user›`

Set the user password of the document. If a document that has a user password is opened for any purpose (such as viewing, printing, editing), either the user or the owner password must be provided.

A user who knows the user password is able to open and read the document. A user who knows the owner password is able to open, read, and modify (e.g. change passwords) the document. A PDF document can have none, either, or both passwords.

Example: Encrypt a document with a user and an owner password

```
pdfoptimize -u userpassword -o ownerpassword input.pdf output.pdf
```

6.2.45 `-v` Verbose mode

Verbose mode `-v`

This option enables the verbose mode. In the verbose mode, the individual steps performed by the 3-Heights® PDF Optimizer Shell are displayed.

6.2.46 `-xf` Extract fonts

Extract fonts `-xf`
License feature: `Optimize`

This option enables the extraction of embedded fonts into to files. Only embedded fonts can be extracted. Non-embedded fonts are not affected. Due to copyright reasons, the extracts fonts are not installable. The generated files are stored in the current directory and are named as following:

- A TrueType font file is named: `fnt‹objno›.ttf`
- A Type 1 font file is named: `fnt‹objno›.pfb`
- A CFF font file is named: `fnt‹objno›.cff`

In the above, `‹objno›` corresponds to the object number of the font in the PDF document.

6.2.47 `-xi` Extract images

Extract images `-xi`

This option extracts the images from a PDF document and automatically stores them as TIFF or JPEG.

The images are stored in the current directory and are named as following:

- `img‹objno›.jpg` for images with JPEG compression
- `img‹objno›.tif` for any other type of image

In the above, `‹objno›` corresponds to the object number of the image in the PDF document. This number can also be retrieved with the option `-li`.

6.3 Return codes

All return codes other than `0` indicate an error in the processing.

Return codes

Value	Description
`0`	Success.
`1`	Couldn't open input file.
`2`	PDF output file could not be created.
`3`	Error with given options, e.g. too many parameters.
`4`	Linearization failed.
`10`	License error, e.g. invalid license key.

7 Tips, tricks, and troubleshooting

You should use an optimization profile `-pr` and adapt it to your needs.

7.1 Output file too large

It is important to understand what kind of content there is in the document. There is no point in trying to optimizing fonts when the document contains scanned images only. Document properties such as embedded fonts and images can be listed using the corresponding listing functions (`-li`, `-lf`).

It is not possible to compress a document arbitrarily without loss of information.

7.1.1 Images

- Try setting a lower threshold and a lower DPI for the images.

  Example: Use the "web" profile and rescale all images with a DPI greater than 72 DPI to 50 DPI

```
pdfoptimize -pr web -dt 72 -dr 50 input.pdf output.pdf
```

- Try reducing the quality of the JPEG and JPEG2000 images by setting `-q`.

  Example: Use the "web" profile and set the quality index to 60

```
pdfoptimize -pr web -q 60 input.pdf output.pdf
```

7.1.2 Fonts

- Apply sub-setting to fonts using `-s`. This means all glyphs of characters that are unused are removed from the font.
- Merge font programs using `-m`. Multiple occurrances of the same glyph in compatible font programs are then merged into one glyph.
- Remove non-symbolic embedded fonts using `-rs`. Keep in mind that the appearance when rendering a PDF document with non-embedded non-PDF standard fonts is unpredictable.

Note: While sub-setting and merging font programs is enabled in most optimization profiles, removing standard fonts is not. (See Profile settings.)

7.2 Output file larger than input file

- The 3-Heights® PDF Optimizer Shell also repairs corrupt documents to a certain extent. This means if relevant data is missing, it is recovered. This could possibly lead to a larger file size.
- If linearization is applied, there is information added to the document. This information contains hints for the browser plugin, and allows it to specifically download only those objects relevant for displaying a certain page. The linearization information can increase the file size by about 1 to 10%.
- The input file may contain compressed object streams. These are currently not re-constructed in the output document.

7.3 Selected compression type not applied

- Not all compression types can be applied to all types of images. Check the tables Bitonal compression, Color/Monochrome compression, and Indexed compression types.
- The optimization is only applied if it reduces the file size. Therefore, an image usually is not re-compressed with a new compression that uses more disk space than the original compression. This behavior can be switched off with `-ff`.

7.4 Output document not encrypted

To encrypt the output document, set an owner password using the `-o` switch and permission flags using the `-p` switch.

Example: Set the owner password to “mypassword” and do not grant any permissions:

```
pdfoptimize -o mypassword -p 0 input.pdf output.pdf
```

It is not possible to inherit the owner or user password or the permission flags from the input document.

8 Version history

8.1 Changes in versions 6.19–6.27

- **Update** license agreement to version 2.9

8.2 Changes in versions 6.13–6.18

No functional changes.

8.3 Changes in versions 6.1–6.12

- **Improved** processing of images with indexed color space. Down-sampling is now supported if color conversion is activated.
- **Changed** behavior for compressing images: Compression types are selected based on the output image's type, not on the input image's type. (The image type can change, e.g., if color conversion is activated.)
- **Changed** optimization profiles: In profiles "Web", "Print", "Archive", and "Max", the "Flate" compression is added to images of type "continuous".

8.4 Changes in version 5

- MRC improvements
  - **Improved** MRC background and foreground layer coloring.
  - **Changed** default value for MRC layer compression quality to `20` [previously `10`].
  - **Changed** MRC behavior: monochrome (gray-scale) images are treated as entire photographic regions (cut-out pictures).
  - **Improved** MRC pre-processing: conversion to RGB is now possible.
  - **Changed** optimization profile `mrc`: Now includes color conversion to RGB.
- **Changed** behavior: Invisible annotations are not removed anymore by default.
- **New** option `-sia` to remove invisible annotations.
- **New** additional supported operating system: Windows Server 2019.

8.5 Changes in version 4.12

- **Introduced** license features `Optimize` and `Color`.
- **Improved** MRC functionality supports images with differing resolution in horizontal and vertical direction.
- **Improved** file size reduction by using object streams for annotations and form fields by default.
- **Improved** runtime for certain document types.
- **New** support for encryption according to PDF 2.0 (revision 6, replaces deprecated revision 5).
- **New** HTTP proxy setting in the GUI license manager.
- **New** option `-owa` to automatically choose whether to linearize the output document or not.
- **Changed** option `-pr`: In the `web` optimization profile, linearization (`-ow`) is substituted by auto linearization (`-owa`).

8.6 Changes in version 4.11

- **New** support for the creation of appearance streams for free text annotations that contain rich text content.
- **New** support for reading and writing PDF 2.0 documents.
- **New** support for the creation of output files larger than 10GB (not PDF/A-1).
- **Improved** font subsetting of CFF and OpenType fonts.
- **Improved** repair of corrupt image streams.
- **New** treatment of the `DocumentID`. In contrast to the `InstanceID` the `DocumentID` of the output document is inherited from the input document.
- **Changed** option `-p`: Added a new value `i` to adopt the encryption parameters from the input document.

8.7 Changes in version 4.10

- **New** support for down-sampling of non-shared Masks and SMasks.
- **Improved** removal of redundant objects: More types of dictionaries are included.
- **New** support for writing PDF objects into object streams. Most objects that are contained in object streams in the input document are now also stored in object streams in the output document. When enabling linearization, however, no objects are stored in object streams.
- **New** feature: If linearization and the removal of document structure information are both disabled, then any document structure elements, if present in the input document, are stored in objects streams in the output document.
- **Improved** robustness against corrupt input PDF documents.
- **Improved** annotation appearance generation for polyline, squiggly, and stamp annotations.
- **Changed** option `-pr`: Added a new profile `archive` to optimize for archiving purposes.

8.8 Changes in version 4.9

- **Improved** support for and robustness against corrupt input PDF documents.
- **Improved** repair of embedded font programs that are corrupt.
- **New** support for OpenType font collections in installed font collection.
- **Improved** metadata generation for standard PDF properties.
- **Changed** option `-pr`: The `max` profile includes color conversion to RGB.
- **New** option `-ri`: Remove images by replacing them with empty XObjects.
- **New** option `-sfs`: Allows to flatten the visual appearance of digital signatures. (During optimization, digital signatures are always removed.)

8.9 Changes in version 4.8

- **Improved** content stream optimization.
- **Improved** creation of annotation appearances to use less memory and processing time.
- **Added** repair functionality for TrueType font programs whose glyphs are not ordered correctly.
- **New** option `-rc`: Reduce color complexity for images.
- **Changed** option `-pr`: The profiles `web`, `print`, and `max` now include the option `-rc`.

9 Licensing, copyright, and contact

Pdftools (PDF Tools AG) is a world leader in PDF software, delivering reliable PDF products to international customers in all market segments.

Pdftools provides server-based software products designed specifically for developers, integrators, consultants, customizing specialists, and IT departments. Thousands of companies worldwide use our products directly and hundreds of thousands of users benefit from the technology indirectly via a global network of OEM partners. The tools can be easily embedded into application programs and are available for a multitude of operating system platforms.

Contact

PDF Tools AG
Brown-Boveri-Strasse 5
8050 Zürich
Switzerland
https://www.pdf-tools.com
pdfsales@pdf-tools.com