User Manual



3-Heights®
PDF Optimizer API

Version 6.27.4

pdftools

Contents

1 Introduction 5
1.1 Description 5
1.2 Functions 5
1.2.1 Features 6
1.2.2 Formats 7
1.2.3 Conformance 7
1.3 Interfaces 7
1.4 Operating systems 8
1.5 How to best read this manual 8

2 Installation and deployment 9
2.1 Windows 9
2.2 Linux and macOS 9
2.2.1 Linux 10
2.2.2 macOS 10
2.3 ZIP archive 10
2.3.1 Development 11
2.3.2 Deployment 12
2.4 NuGet package 13
2.5 Interface-specific installation steps 14
2.5.1 COM interface 14
2.5.2 Java interface 14
2.5.3 .NET interface 15
2.5.4 C interface 15
2.6 Uninstall, Install a new version 15
2.7 Color profiles 15
2.7.1 Default color profiles 16
2.7.2 Get other color profiles 16
2.8 Note about the evaluation license 16

3 License management 17
3.1 License features 17

4 Programming interfaces 18
4.1 Visual Basic 6 18
4.2 ASP - VBScript 19
4.3 .NET 19
4.3.1 Visual Basic 19
4.3.2 C# 20
4.3.3 Deployment 21
4.3.4 Troubleshooting: TypeInitializationException 21

5 User manual 23
5.1 Overview of the API 23
5.2 Optimizing PDF documents 23
5.2.1 Identifying target application area 23
Web 23
Printing 24
Archiving 24

Scanned documents . . . 25
Special requirements . . . 25
5.2.2 Using optimization profiles . . . 25
5.3 Optimizing images . . . 25
5.3.1 Supported image compression types . . . 26
No compression (raw) . . . 26
DCT (JPEG) . . . 26
Flate (ZIP) . . . 26
LZW . . . 27
CCITT Fax Group 3 and 4 . . . 27
JBIG2 . . . 27
JPEG2000 . . . 28
5.3.2 Relevant factors for file size . . . 28
5.3.3 Provided features for optimizing images . . . 29
5.3.4 MRC optimization for images . . . 30
Stage 1: Cutting out pictures . . . 31
Stage 2: Separating into layers . . . 31
Stage 3: Reconstructing the image . . . 32
5.4 Optimizing fonts . . . 32
5.5 Extracting resources . . . 33
5.6 Error handling . . . 33

6 Interface reference . . . 35
6.1 `PDFOptimizer` Interface . . . 35
6.1.1 `AutoLinearize` . . . 35
6.1.2 `BitonalCompression` . . . 35
6.1.3 `BitonalCompressions` . . . 36
6.1.4 `BitonalResolutionDPI` . . . 36
6.1.5 `BitonalThresholdDPI` . . . 36
6.1.6 `ClipImages` . . . 37
6.1.7 `Close` . . . 37
6.1.8 `ColorCompression` . . . 37
6.1.9 `ColorConversion` . . . 38
6.1.10 `ColorResolutionDPI` . . . 38
6.1.11 `ColorThresholdDPI` . . . 38
6.1.12 `ContinuousCompressions` . . . 39
6.1.13 `CompressionQuality` . . . 39
6.1.14 `ConvertToCFF` . . . 39
6.1.15 `DitheringMode` . . . 39
6.1.16 `ErrorCode` . . . 40
6.1.17 `ErrorMessage` . . . 40
6.1.18 `ExtractFonts` . . . 40
6.1.19 `ExtractImages` . . . 41
6.1.20 `FlattenSignatureFields` . . . 41
6.1.21 `ForceCompressionTypes` . . . 41
6.1.22 `ForceRecompression` . . . 42
6.1.23 `GetInfoEntry` . . . 42
6.1.24 `GetPdf` . . . 42
6.1.25 `ImageStratConserv` . . . 42
6.1.26 `ImageQuality` . . . 43
6.1.27 `IndexedCompressions` . . . 43
6.1.28 `LicenseIsValid` . . . 43

6.1.29 Linearize ... 44
6.1.30 LinearizeFile ... 44
6.1.31 ListFonts ... 45
6.1.32 ListImages ... 46
6.1.33 MergeEmbeddedFonts ... 47
6.1.34 MonochromeCompression ... 47
6.1.35 MonochromeResolutionDPI ... 47
6.1.36 MonochromeThresholdDPI ... 47
6.1.37 MrcLayerCompression ... 48
6.1.38 MrcLayerQuality ... 48
6.1.39 MrcLayerResolutionDPI ... 48
6.1.40 MrcMaskCompression ... 49
6.1.41 MrcPictCompression ... 49
6.1.42 MrcRecognizePictures ... 49
6.1.43 Open ... 49
6.1.44 OpenMem ... 50
6.1.45 OpenStream ... 50
6.1.46 OptimizeResources ... 51
6.1.47 PageCount ... 51
6.1.48 ProductVersion ... 51
6.1.49 Profile ... 51
6.1.50 ReduceColorComplexity ... 52
6.1.51 RemoveImages ... 52
6.1.52 RemoveNonSymbolicFonts ... 52
6.1.53 RemoveRedundantObjects ... 53
6.1.54 RemoveStandardFonts ... 53
6.1.55 ResolutionDPI ... 54
6.1.56 SaveAs ... 54
6.1.57 SaveInMemory ... 55
6.1.58 SaveAsStream ... 55
6.1.59 SetCMSEngine ... 56
6.1.60 SetInfoEntry ... 56
6.1.61 SetLicenseKey ... 57
6.1.62 SetVersion ... 57
6.1.63 Strip ... 58
6.1.64 SubsetFonts ... 58
6.1.65 ThresholdDPI ... 58
6.1.66 UnembedFont ... 59
6.2 Enumerations ... 59
6.2.1 TPDFColorConversion Enumeration ... 59
6.2.2 TPDFComprAttempt Enumeration ... 59
6.2.3 TPDFCompression Enumeration ... 60
6.2.4 TPDFErrorCode Enumeration ... 61
6.2.5 TPDFOptimizationProfile Enumeration ... 61
6.2.6 TPDFPermission Enumeration ... 65
6.2.7 TPDFStripType Enumeration ... 66

7 Version history ... 68
7.1 Changes in versions 6.19–6.27 ... 68
7.2 Changes in versions 6.13–6.18 ... 68
7.3 Changes in versions 6.1–6.12 ... 68
7.4 Changes in version 5 ... 68

7.5 Changes in version 4.12 69
7.6 Changes in version 4.11 69
7.7 Changes in version 4.10 69
7.8 Changes in version 4.9 69
7.9 Changes in version 4.8 70

8 Licensing, copyright, and contact 71

1 Introduction

1.1 Description

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

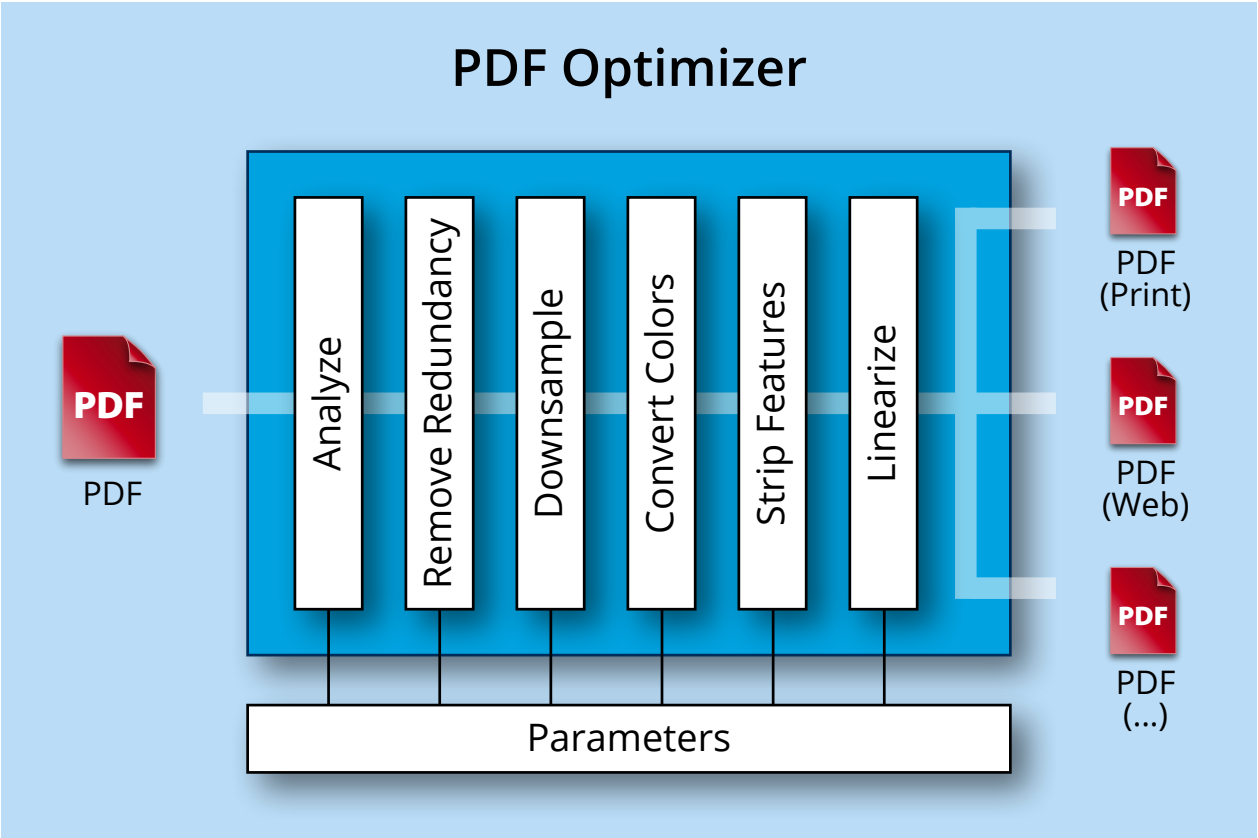
Many processes produce PDF documents that may not be optimized for their specific target application. For example, the file size may deteriorate download times, or the same font embedded multiple times may impede printing. In most cases, there is no advantage gained when trying to convert a PDF to some 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 API 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 API 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 API 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
    - Process memory-resident files
    - 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
    - Extract number of pages
    - 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 Interfaces

The following interfaces are available:

- C
- Java
- .NET Framework
- .NET Core[1]
- COM

[1] Limited supported OS versions. Operating systems

1.4 Operating systems

The 3-Heights® PDF Optimizer API 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.

1.5 How to best read this manual

If you are reading this manual for the first time and would like to evaluate the software, the following steps are suggested:

1. Read the Introduction chapter to verify this product meets your requirements.
2. Identify what interface your programming language uses.
3. Read and follow the instructions in Installation and deployment.
4. In Programming interfaces, find your programming language. Please note that not every language is covered in this manual.
   For most programming languages, there is sample code available. To start, it is generally best to refer to these samples rather than writing code from scratch.
5. (Optional) Read the User manual for general information about the API. Read the Interface reference for specific information about the functions of the API.

2 Installation and deployment

2.1 Windows

The 3-Heights® PDF Optimizer API comes as a ZIP archive or as a NuGet package.

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 API". 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.
   The product comes as a ZIP archive containing all files, or as a NuGet package containing all files for development in .NET.
   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 ZIP archive as well as the NuGet package contain both the 32-bit and the 64-bit version of the product.
3. 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 (see also ZIP archive):

Subdirectory	Description
`bin`	Runtime executable binaries
`doc`	Documentation
`include`	Header files to include in your C/C++ project
`jar`	Java archive files for Java components
`lib`	Object file library to include in your C/C++ project
`samples`	Sample programs in various programming languages

4. The usage of the NuGet package is described in section NuGet package.
5. (Optional) Register your license key using the License management.
6. Identify the interface you are using. Perform the specific installation steps for that interface described in Interface-specific installation steps.
7. Make sure your platform meets the requirements regarding color spaces described in Color profiles.

2.2 Linux and macOS

This section describes installation steps required on Linux or macOS.

The Linux and macOS version of the 3-Heights® PDF Optimizer API provides two interfaces:

- Java interface
- Native C interface

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

File description

Name	Description
bin/x64/libPdfOptimizeAPI.so	Shared library that contains the main functionality. The file's extension differs on macOS, (.dylib instead of .so).
doc/*.*	Documentation
include/*.h	Header files to include in your C/C++ project
jar/POLA.jar	Java API archive
samples	Example code

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 libPdfOptimizeAPI.so
```

   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 shared library from one of the standard library directories, e.g.

```
ln -s /opt/pdf-tools.com/bin/x64/libPdfOptimizeAPI.so /usr/lib
```

4. Optionally, register your license key using the license manager.
5. Identify the interface you are using. Perform the specific installation steps for that interface described in Interface-specific installation steps.
6. Make sure your platform meets the requirements regarding color spaces described in Color profiles.

2.2.2 macOS

The shared library must have the extension .jnilib for use with Java. Create a file link for this purpose by using the following command:

```
ln libPdfOptimizeAPI.dylib libPdfOptimizeAPI.jnilib
```

2.3 ZIP archive

The 3-Heights® PDF Optimizer API provides four different interfaces. The installation and deployment of the software depend on the interface you are using. The table below shows the supported interfaces and some of the programming languages that can be used.

Interface	Programming languages
.NET	The MS software platform .NET can be used with any .NET capable programming language such as: ■ C# ■ VB .NET ■ J# ■ others For a convenient way to use this interface, see NuGet package.
Java	The Java interface is available on all platforms.
COM	The component object model (COM) interface can be used with any COM-capable programming language, such as: ■ MS Visual Basic ■ MS Office Products such as Access or Excel (VBA) ■ C++ ■ VBScript ■ others This interface is available in the Windows version only.
C	The native C interface is for use with C and C++. This interface is available on all platforms.

2.3.1 Development

The software development kit (SDK) contains all files that are used for developing the software. The role of each file in each of the four different interfaces is shown in table Files for development. The files are split in four categories:

Req. The file is required for this interface.

Opt. The file is optional. See also the File description table to identify the files are required for your application.

Doc. The file is for documentation only.

Empty field An empty field indicates this file is not used for this particular interface.

Files for development

Name	.NET	Java	COM	C
bin\‹platform›\PdfOptimizeAPI.dll	Req.	Req.	Req.	Req.
bin*NET.dll	Req.			
bin*NET.xml	Doc.			
doc*.pdf	Doc.	Doc.	Doc.	Doc.
doc\PdfOptimizeAPI.idl			Doc.	
doc\javadoc*.*		Doc.		

Files for development

Name	.NET	Java	COM	C
include\pdfoptimizeapi_c.h				Req.
include*.*				Opt.
jar\POLA.jar		Req.		
lib\‹platform›\PdfOptimizeAPI.lib				Req.[2]
samples*.*	Doc.	Doc.	Doc.	Doc.

The purpose of the most important distributed files is described in the File description table.

File description

Name	Description
bin\‹platform›\PdfOptimizeAPI.dll	DLL that contains the main functionality (required), where ‹platform› is either Win32 or x64 for the 23-bit or the 64-bit library, respectively.
bin*NET.dll	.NET assemblies are required when using the .NET interface. The files bin*NET.xml contain the corresponding XML documentation for MS Visual Studio.
doc*.*	Documentation
include*.*	Files to include in your C / C++ project
lib\‹platform›\PdfOptimizeAPI.lib	On Windows operating systems, the object file library needs to be linked to the C/C++ project.
jar\POLA.jar	Java API archive
samples*.*	Sample programs in different programming languages

2.3.2 Deployment

For the deployment of the software, only a subset of the files are required. The table below shows the files that are required (Req.), optional (Opt.) or not used (empty field) for the four different interfaces.

Files for deployment

Name	.NET	Java	COM	C
bin\‹platform›\PdfOptimizeAPI.dll	Req.	Req.	Req.	Req.

[2] Not required for Linux or macOS.
[3] These files must reside in the same directory as PdfOptimizeAPI.dll.

Files for deployment

`bin\*NET.dll`	Req.
`jar\POLA.jar`	Req.

The deployment of an application works as described below:

1. Identify the required files from your developed application (this may also include color profiles).
2. Identify all files that are required by your developed application.
3. Include all these files in an installation routine such as an MSI file or a simple batch script.
4. Perform any interface-specific actions (e.g. registering when using the COM interface).

Example: This is a very simple example of how a COM application written in Visual Basic 6 could be deployed.

1. The developed and compiled application consists of the file `application.exe`. Color profiles are not used.
2. The application uses the COM interface and is distributed on Windows only.
   - The main DLL `PdfOptimizeAPI.dll` must be distributed.
3. All files are copied to the target location using a batch script. This script contains the following commands:

```
copy application.exe %targetlocation%\.
copy PdfOptimizeAPI.dll %targetlocation%\.
```

4. For COM, the main DLL needs to be registered in silent mode (`/s`) on the target system. This step requires Power-User privileges and is added to the batch script.

```
regsvr32 /s %targetlocation%\PdfOptimizeAPI.dll.
```

2.4 NuGet package

NuGet is a package manager that lets you integrate libraries for software development in .NET. The NuGet package for the 3-Heights® PDF Optimizer API contains all the libraries needed, both managed and native.

Installation

The package `PdfTools.PdfOptimize 6.27.4` is available on nuget.org. Right-click on your .NET project in Visual Studio and select “Manage NuGet Packages...”. Finally, select the package source “nuget.org” and navigate to the package `PdfTools.PdfOptimize 6.27.4`.

Development

The package `PdfTools.PdfOptimize 6.27.4` contains .NET libraries with versions .NET Standard 1.1, .NET Standard 2.0, and .NET Framework 2.0, and native libraries for Windows, macOS, and Linux.

The required native libraries are loaded automatically. All project platforms are supported, including “AnyCPU”.

To use the software, you must first install a license key for the 3-Heights® PDF Optimizer API. To do this, you have to download the product kit and use the license manager in it. See also License management.

Note: This NuGet package is only supported on a subset of the operating systems supported by .NET Core. See also Operating systems.

2.5 Interface-specific installation steps

2.5.1 COM interface

Registration

Before you can use the 3-Heights® PDF Optimizer API component in your COM application program, you have to register the component using the `regsvr32.exe` program that is provided with the Windows operating system. The following command shows how to register the PdfOptimizeAPI.dll. In Windows Vista and later, the command needs to be executed from an administrator shell.

```
regsvr32 "C:\Program Files\PDF Tools AG\bin\‹platform›\PdfOptimizeAPI.dll"
```

Where `‹platform›` is `Win32` for the 32-bit and `x64` for the 64-bit version.

If you are using a 64-bit operating system and would like to register the 32-bit version of the 3-Heights® PDF Optimizer API, you need to use the `regsvr32` from the directory `%SystemRoot%\SysWOW64` instead of `%SystemRoot%\System32`.[4]

If the registration process succeeds, a corresponding dialog window is displayed. The registration can also be done silently (e.g. for deployment) using the switch `/s`.

Other files

The other DLLs do not need to be registered, but for simplicity, it is suggested that they reside in the same directory as the `PdfOptimizeAPI.dll`.

2.5.2 Java interface

The 3-Heights® PDF Optimizer API requires Java version 7 or higher.

For compilation and execution

When using the Java interface, the Java wrapper `jar\POLA.jar` needs to be on the CLASSPATH. You can do this by either adding it to the environment variable CLASSPATH, or by specifying it using the switch `-classpath`:

```
javac -classpath ".;C:\Program Files\PDF Tools AG\jar\POLA.jar" ^
 sampleApplication.java
```

For execution

Additionally, the library `PdfOptimizeAPI.dll` needs be in one of the system's library directories[5] or added to the Java system property `java.library.path`. You can add the library by either adding it dynamically at program startup before using the API, or by specifying it using the switch `-Djava.library.path` when starting the Java VM. Choose the correct subdirectory (`x64` or `Win32` on Windows) depending on the platform of the Java VM[6].

[4] Otherwise, you get the following message: `LoadLibrary("PdfOptimizeAPI.dll") failed - The specified module could not be found.`

[5] On Windows defined by the environment variable PATH, and on Linux defined by LD_LIBRARY_PATH.

[6] If the wrong data model is used, there is an error message similar to this: "`Can't load IA 32-bit .dll on a AMD 64-bit platform`"

```
java -classpath ".;C:\Program Files\PDF Tools AG\POLA.jar" ^
 "-Djava.library.path=C:\Program Files\PDF Tools AG\bin\x64" sampleApplication
```

On Linux or macOS, the path separator usually is a colon and hence the above changes to something like:

```
... -classpath ".:/path/to/POLA.jar" ...
```

2.5.3 .NET interface

The 3-Heights® PDF Optimizer API does not provide a pure .NET solution. Instead, it consists of a native library and .NET assemblies, which call the native library. This has to be accounted for when installing and deploying the tool.

It is recommended that you use the NuGet package. This ensures the correct handling of both the .NET assemblies and the native library.

Alternatively, the files in the ZIP archive can be used directly in a Visual Studio project targeting .NET Framework 2.0 or later. To achieve this, proceed as follows:

The .NET assemblies (`*NET.dll`) are added as references to the project; they are needed at compile time. `PdfOptimizeAPI.dll` is not a .NET assembly, but a native library. It is not added as a reference to the project. Instead, it is loaded during execution of the application.

For the operating system to find and successfully load the native library `PdfOptimizeAPI.dll`, it must match the executing application's bitness (32-bit versus 64-bit) and it must reside in either of the following directories:

- In the same directory as the application that uses the library
- In a subdirectory `win-x86` or `win-x64` for 32-bit or 64-bit applications, respectively
- In a directory that is listed in the PATH environment variable

In Visual Studio, when using the platforms "x86" or "x64", you can do this by adding the 32-bit or 64-bit `PdfOptimizeAPI.dll`, respectively, as an "existing item" to the project, and setting its property "Copy to output directory" to true. When using the "AnyCPU" platform, make sure, by some other means, that both the 32-bit and the 64-bit `PdfOptimizeAPI.dll` are copied to subdirectories `win-x86` and `win-x46` of the output directory, respectively.

2.5.4 C interface

- The header file `pdfoptimizeapi_c.h` needs to be included in the C/C++ program.
- On Windows operating systems, the library `PdfOptimizeAPI.lib` needs to be linked to the project.
- The dynamic link library `PdfOptimizeAPI.dll` needs to be in a path of executables (e.g. on the environment variable %PATH%).

2.6 Uninstall, Install a new version

If you have used the ZIP file for the installation, undo all the steps done during installation, e.g. de-register using `regsvr32.exe /u`, delete all files, etc.

Installing a new version does not require you to previously uninstall the old version. The files of the old version can directly be overwritten with the new version.

2.7 Color profiles

The color conversion feature of the 3-Heights® PDF Optimizer API 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 API 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 API.

2.7.1 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 `PdfOptimizeAPI.dll` resides.

Linux and macOS

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

2.7.2 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

2.8 Note about the evaluation license

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

3 License management

The 3-Heights® PDF Optimizer API requires a valid license in order to run correctly. If no license key is set or the license is not valid, then most of the interface elements documented in Interface reference fail with an error code and error message indicating the reason.

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

3.1 License features

The functionality of the 3-Heights® PDF Optimizer API 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.

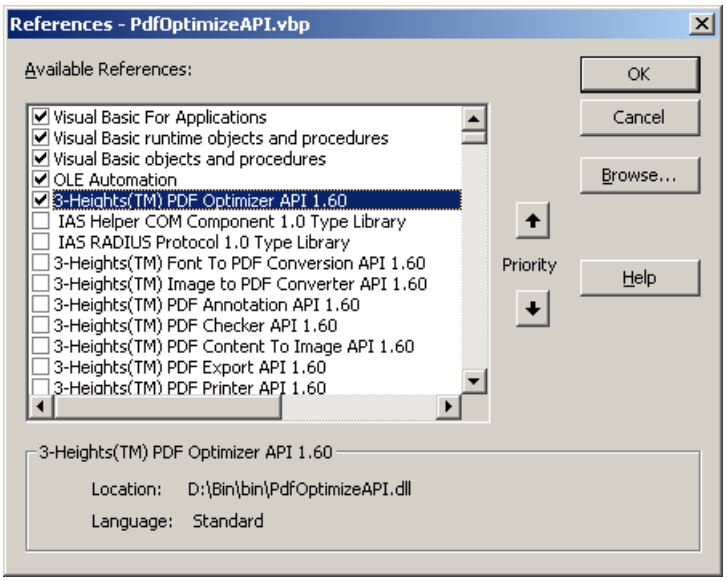
4 Programming interfaces

4.1 Visual Basic 6

After installing the 3-Heights® PDF Optimizer API and registering the COM interface (see Installation and deployment), you find a Visual Basic 6 example `PdfOptimizeAPI.vbp` in the directory `samples/VB/`. You can either use this sample as a base for an application, or you can start from scratch.

If you start from scratch, perform these steps:

1. Create a new Standard-Exe Visual Basic 6 project. Then include the 3-Heights® PDF Optimizer API component to your project.



2. Draw a new Command Button and optionally, rename it as appropriate.
3. Double-click the command button and insert the few lines of code below. All that you need to change is the path of the file name.

Example: Simple Visual Basic sample

```
Private Sub Command1_Click()
    Dim Opt As New PDFOPTIMIZEAPILib.PdfOptimize
    ' Open and analyze the input file.
    Opt.Open "C:\pdf\input.pdf"
    ' Optimize output
    Opt.ColorConversion = eConvRGB
    Opt.BitonalCompressions = eComprAttemptGroup4
    Opt.ContinuousCompressions = eComprAttemptJPEG + eComprAttemptJPEG2000
    Opt.ImageQuality = 75
    Opt.ResolutionDPI = 150
    Opt.ThresholdDPI = 225
    Opt.Linearize = True
    Opt.RemoveRedundantObjects = True
    Opt.SaveAs "C:\out.pdf", "owner", ePermPrint + ePermFillForms
    ' Terminate
    Opt.Close
```

```
End Sub
```

4.2 ASP - VBScript

```
<%@ Language=VBScript %>
<%
  option explicit

  dim pdfOpt
  set pdfOpt = Server.CreateObject("PDFOPTIMIZEAPI.PDFOptimizer")

  if not pdfOpt.Open("https://www.pdf-tools.com /public/downloads/manuals/
pola.pdf", "") then
      Response.Write "<p>"
      Response.Write "Could not open input file." & "<br>"

  else
  pdfOpt.RemoveRedundantObjects = True
  pdfOpt.Linearize = True
  if not pdfOpt.SaveAs("C:\temp\optimized.pdf", vbNullString, vbNullString,
   -1)
  then
     Response.Write "<p>"
     Response.Write "Could not save optimized file." & "<br>"
  else
     Response.Write "<p>"
     Response.Write "Optimized output file created <br>"
     Response.Write "</p>"
  end if
end if
%>
```

4.3 .NET

There should be at least one .NET sample for MS Visual Studio available in the ZIP archive of the Windows version of the 3-Heights® PDF Optimizer API. The easiest to quickly start is to refer to this sample.

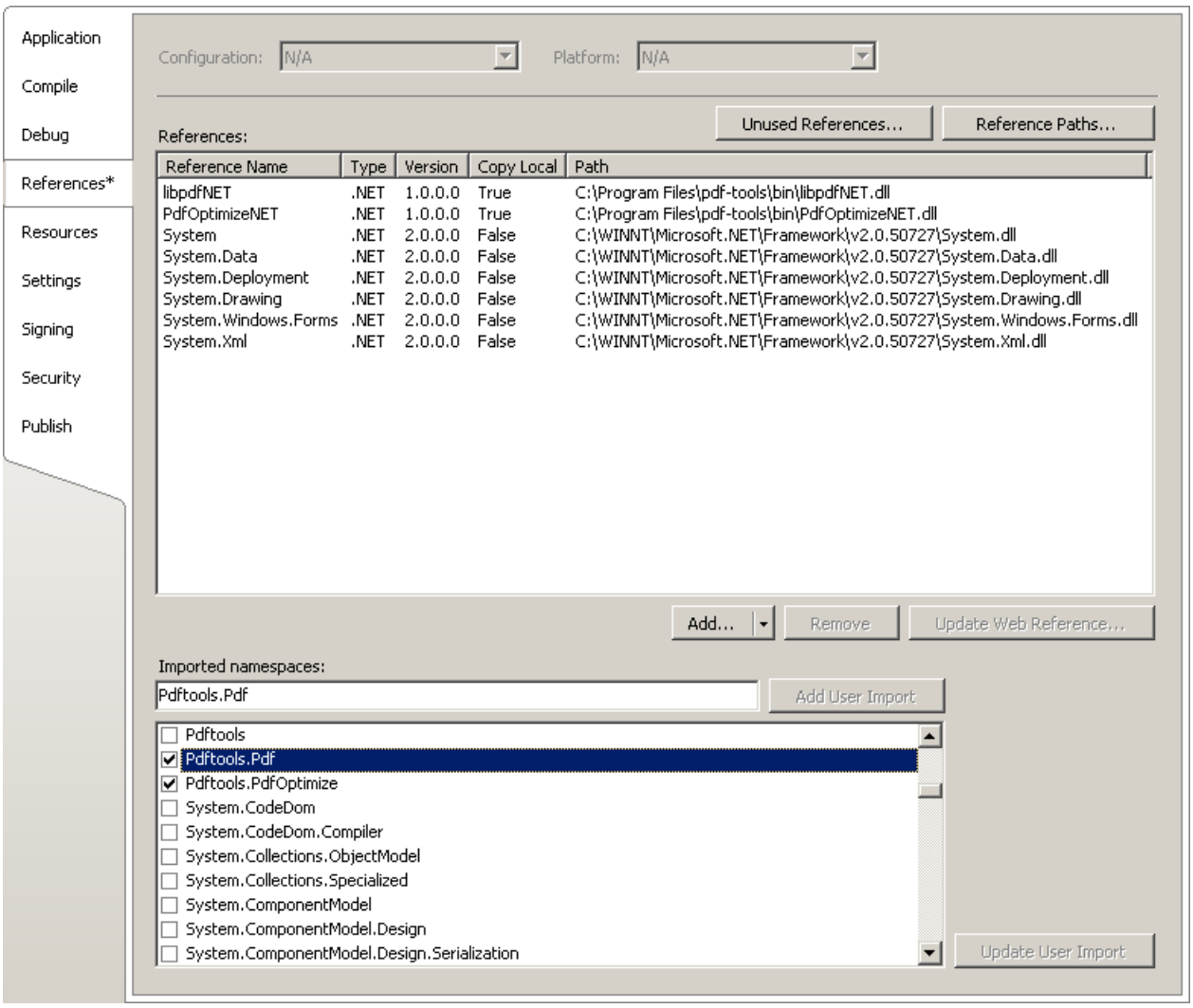
To create a new project from scratch, perform the following steps:

1. Start Visual Studio and create a new C# or VB project.
2. Add references to the NuGet package `PdfTools.PdfOptimize 6.27.4`, as described in NuGet package.
3. Import namespaces (Note: This step is optional, but useful.)
4. Write your code.

Steps 3 and 4 are shown separately for C# and Visual Basic.

4.3.1 Visual Basic

3. Double-click "My Project" to view its properties. On the left hand side, select the menu "References". The .NET assemblies you added before should show up in the upper window. In the lower window, import the namespaces `Pdftools.Pdf`, `Pdftools.PdfRenderer` , and `Pdftools.PdfOptimize`.
   You should now have settings similar as in the screenshot below:



4. The .NET interface can now be used as shown below:

Example:

```
Dim opt As New Pdftools.PdfOptimize.Optimizer
opt.Open(...)
...
opt.Close()
```

4.3.2 C#

3. Add the following namespaces:

Example:

```
using Pdftools.Pdf;
using Pdftools.PdfRenderer;
using Pdftools.PdfOptimize;
```

4. The .NET interface can now be used as shown below:

Example:

```
using (Optimizer opt = new Optimizer())
{
    opt.Open(...);
    ...
    opt.Close();
}
```

4.3.3 Deployment

This is a guideline on how to distribute a .NET project that uses the 3-Heights® PDF Optimizer API:

1. The project must be compiled using Microsoft Visual Studio. See also .NET interface.
2. For deployment, all items in the project's output directory (e.g. `bin\Release`) must be copied to the target computer. This includes the 3-Heights® PDF Optimizer API's .NET assemblies (`*NET.dll`), as well as the native library (`PdfOptimizeAPI.dll`) in its 32 bit or 64 bit version or both. The native library can alternatively be copied to a directory listed in the PATH environment variable, e.g. `%SystemRoot%\System32`.
3. It is crucial that the native library `PdfOptimizeAPI.dll` is found at execution time, and that the native library's format (32 bit versus 64 bit) matches the operating system.
4. The output directory may contain multiple versions of the native library, e.g. for Windows 32 bit, Windows 64 bit, MacOS 64 bit, and Linux 64 bit. Only the versions that match the target computer's operating system need be deployed.
5. If required by the application, optional DLLs must be copied to the same folder. See Deployment for a list and description of optional DLLs.

4.3.4 Troubleshooting: TypeInitializationException

The most common issue when using the .NET interface is that the correct native DLL `PdfOptimizeAPI.dll` is not found at execution time. This normally manifests when the constructor is called for the first time and an exception of type `System.TypeInitializationException` is thrown.

This exception can have two possible causes, which you distinguish by the inner exception (property `InnerException`):

`System.DllNotFoundException` Unable to load DLL `PdfOptimizeAPI.dll`: The specified module could not be found.

`System.BadImageFormatException` An attempt was made to load a program with an incorrect format.

The following sections describe in more detail how to resolve these issues.

Troubleshooting: DllNotFoundException

This means that the native DLL `PdfOptimizeAPI.dll` could not be found at execution time.

Resolve this by performing one of these actions:

- Use the NuGet package.
- Add `PdfOptimizeAPI.dll` as an existing item to your project and set its property "Copy to output directory" to "Copy if newer", or
- Add the directory where `PdfOptimizeAPI.dll` resides to the environment variable `%Path%`, or
- Manually copy `PdfOptimizeAPI.dll` to the output directory of your project.

Troubleshooting: BadImageFormatException

The exception means that the native DLL `PdfOptimizeAPI.dll` has the incorrect "bitness" (i.e. platform 32 vs. 64 bit). There are two versions of `PdfOptimizeAPI.dll` available in the ZIP archive: one is 32-bit (directory `bin\Win32`) and the other 64-bit (directory `bin\x64`). It is crucial that the platform of the native DLL matches the platform of the application's process.

(Using the NuGet package normally ensures that the matching native DLL is loaded at execution time.)

The platform of the application's process is defined by the project's platform configuration for which there are three possibilities:

AnyCPU This means that the application runs as a 32-bit process on 32-bit Windows and as 64-bit process on 64-bit Windows. When using AnyCPU, then the correct native DLL must be used, depending on the Windows platform. You can perform this either when installing the application by installing the matching native DLL, or at application start-up by determining the application's platform and ensuring the matching native DLL is loaded. The latter can be achieved by placing both the 32 bit and the 64 bit native DLL in subdirectories `win-x86` and `win-x64` of the application's directory, respectively.

x86 This means that the application always runs as 32-bit process, regardless of the platform of the Windows installation. The 32-bit DLL runs on all systems.

x64 This means that the application always runs as 64-bit process. As a consequence, the application will not run on a 32-bit Windows system.

5 User manual

5.1 Overview of the API

The 3-Heights® PDF Optimizer API requires a PDF document as an input document. The input document is processed (optimized) and the result is stored into an output PDF document. (When only Extracting resources then no output PDF document need be produced.)

Basically, the API is used as follows:

1. Create a `PDFOptimizer` object.
2. Configure optimization settings by setting various properties of this object.
3. Use the object (potentially several times) to process a PDF document.
4. Destroy the object.

Details about how to configure the optimizer is described in Optimizing PDF documents and subsequent sections, and in the Interface reference.

The optimization process consists of three steps:

1. Open document The document is opened from a file (`Open`) or from memory (`OpenMem`). Encrypted documents are automatically decrypted. For password-protected encryption, a user password must be provided.

2. Analyze document The document is analyzed automatically.

3. Optimize and save as new document The optimizer goes through the whole input document and constructs an output document. While doing so, the optimizer transforms objects of the input document according to the optimizer's settings. Often not all the settings are relevant. For example, if an input document contains color images only, the settings for monochrome and bitonal images are not used.

The output document is constructed either as a file (`SaveAs`) or as a block of memory (`SaveInMemory`). The output file name must be different from the input file name. The input document, regardless of whether it a file or a block of memory, is never changed.

5.2 Optimizing PDF documents

5.2.1 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 (`ThresholdDPI`, `ResolutionDPI`)
- Clipping images to their visible parts (`ClipImages`)

- Applying compressions algorithms with high compression ratios (BitonalCompressions, ContinuousCompressions, IndexedCompressions)
- Collapsing redundant objects (RemoveRedundantObjects)
- Removing unused resources (OptimizeResources)
- Removing irrelevant information such as article threads, metadata, alternate images, document structure information, etc. (Strip)
- Merging and sub-setting embedded font programs (MergeEmbeddedFonts and SubsetFonts)
- Depending on the PDF documents to be optimized, font programs of embedded standard fonts can even be removed (RemoveStandardFonts).

Additionally, PDF documents can be linearized (Linearize). 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. (ColorConversion)

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). (ColorConversion)

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. (MergeEmbeddedFonts and SubsetFonts)

There are still further ways to decrease the file size:

- Clipping images to their visible parts (ClipImages)
- Compressing uncompressed images, e.g. with a lossless compression type (BitonalCompressions, ContinuousCompressions, IndexedCompressions, see also Supported image compression types)
- Collapsing redundant objects (RemoveRedundantObjects)
- Removing unused resources (OptimizeResources)
- Removing irrelevant information for printing such as thumbnails, article threads, document structure information, etc. (Strip)

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 API 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 API supports MRC, see also MRC optimization for images.

Special requirements

As an example for a specific requirement, the 3-Heights® PDF Optimizer API supports the restriction of compression types for images in a document to a specific list of types. (See `ForceCompressionTypes`, `BitonalCompressions`, `ContinuousCompressions`, and `IndexedCompressions`.)

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 API provides both in the `SaveAs` method.

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.)

5.2.2 Using optimization profiles

The 3-Heights® PDF Optimizer API 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. See `TPDFOptimizationProfile` for all available profiles and their configuration values. The configuration is done by setting this profile (`Profile`) followed by adjusting individual settings that differ from the profile.

Example: Adapt the "web" profile to not downsample bitonal images

```
Dim Opt As New PDFOPTIMIZEAPILib.PDFOptimizer
Opt.Profile =
  PDFOPTIMIZEAPILib.TPDFOptimizationProfile.eOptimizationProfileWeb
Opt.BitonalThresholdDPI = -1
'Do some optimization e.g. by calling Opt.Open(...) and Opt.SaveAs(...)
```

5.3 Optimizing images

For the 3-Heights® PDF Optimizer API, the target compression type of an image is specified by means of the `BitonalCompressions`, `ContinuousCompressions`, and `IndexedCompressions` properties. Values of `TPDFComprAttempt` must be used to set these properties. Several values can be combined by means of a bitwise or operation.

5.3.1 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)

5.3.2 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.

5.3.3 Provided features for optimizing images

The 3-Heights® PDF Optimizer API 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 API, 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 API 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 API 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 API 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 API cannot be used to "un-compress" embedded images.

5.3.4 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 `TPDFOptimizationProfile`.

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 API 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 API, 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.

This first stage is optional and can be switched off using the `MrcRecognizePictures` property. Even when switched off, monochrome (grayscale) images are still treated as 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.

5.4 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 API Tool can merge these fonts into one font (if they can be merged).

5.5 Extracting resources

The 3-Heights® PDF Optimizer API can extract resources such as images or fonts and save them to files in the file system. This is achieved as follows:

1. Configure the optimizer to extract images and fonts by setting `ExtractImages` and `ExtractFonts`, respectively, to `True`.
2. Set the current working directory to the directory into which the files of extracted resources are saved. The procedure to set the current directory depends on the programming language and the operating system.
3. Open the PDF from which to extract resources by calling `Open` or `OpenMem`.
4. Execute the extraction by calling `SaveAs` or `SaveInMemory`. It is not possible to extract resources without producing an optimized output PDF, either as a file or a memory block.

Example: Extracting resources in C#

```
string inputPath = @"C:\Path\to\input.pdf";
string outputPath = @"C:\Path\to\output.pdf";
string extractionPath = @"C:\Path\to\extraction\directory";
using (Optimizer opt = new Optimizer())
{
    opt.ExtractFonts = true;
    opt.ExtractImages = true;

    System.IO.Directory.SetCurrentDirectory(extractionPath);

    if (!opt.Open(inputPath, ""))
        ; // Handle error

    if (!opt.SaveAs(outputPath, "", "", PDFPermission.ePermNoEncryption))
        ; // Handle error

    if (!opt.Close())
        ; // Handle error
}
```

These resources are extracted unaltered from the PDF document. In particular, this means:

- Fonts are not converted to installable fonts, i.e. extracted fonts cannot be installed and used on the operating system. (Copyright prevents extracted fonts from being installed.)
- Images are extracted from the resources without the context of the page. This means they do not inherit the resolution of the image on the page in the PDF document. (The same image could be used multiple times in the document at different resolutions). Also images on the PDF page could possibly be clipped (i.e. not the complete image is visible), or stretched or rotated, etc. All these PDF operators affecting the visual appearance of the image on the page are disregarded.

5.6 Error handling

Most methods of the 3-Heights® PDF Optimizer API can either succeed or fail depending on user input, the state of the PDF Optimizer API, or the state of the underlying system. It is important to detect and handle these errors to get accurate information about the nature and source of the issue at hand.

Methods communicate their level of success or failure using their return value. The return values to be interpreted as failures are documented in the Interface reference. To identify the error on a programmatic level, check the

ErrorCode property. The ErrorMessage property provides a human readable error message, which describes the error.

Example:

```
public Boolean Open(string file, string password)
{
  if (!opt.Open(file, password))
  {
    if (opt.ErrorCode == PDFErrorCode.PDF_E_PASSWORD)
    {
      password = InputBox.Show("Password incorrect. Enter correct password:");
      return Open(file, password);
    }
    else
    {
      MessageBox.Show(String.Format(
        "Error {0}: {1}", doc.ErrorCode, doc.ErrorMessage));
      return false;
    }
  }
  [...]
}
```

6 Interface reference

Note: This manual describes the COM interface only. Other interfaces (C, Java, .NET) work similarly, i.e. they have calls with similar names and the call sequence to be used is the same as with COM.

Note: Methods and properties marked with a license feature are available if the listed feature is included in the license. See also License features.

6.1 PDFOptimizer Interface

6.1.1 AutoLinearize

Property (get, set): `Boolean AutoLinearize`
Default: `False`
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 API 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 `Linearize` for more information for linearized PDFs.

Note: If this property is set to `True`, then the value given to `Linearize` is ignored.

6.1.2 BitonalCompression

[Deprecated] Property (get, set): `TPDFCompression BitonalCompression`

Deprecated in Version 4.6. Use `BitonalCompressions`.

6.1.3 BitonalCompressions

Property (get, set): TPDFCompressionAttempt BitonalCompressions
Default: eComprNone

Get or set the compression types for bitonal images. See also TPDFComprAttempt enumeration.

Several values can be combined with bitwise or operators. The following values are allowed:

- eComprAttemptNone
- eComprAttemptRaw
- eComprAttemptFlate
- eComprAttemptLZW
- eComprAttemptGroup3
- eComprAttemptGroup4
- eComprAttemptSource
- eComprAttemptJBIG2

Other values are ignored.

During optimization, all set compression types are tried and the one resulting in the least memory footprint is taken.

Typically, CCITT Group 4 or JBIG2 is used for bitonal compression. Due to the simpler algorithm, CCITT Group 4 has the advantage of being faster. JBIG2 can achieve compression ratios that are up to twice as high as CCITT Group 4 at the cost of longer computation time.

If BitonalCompressions is set to eComprAttemptNone, then bitonal images are excluded from processing. They are neither re-sampled (BitonalResolutionDPI, BitonalThresholdDPI) or clipped (ClipImages).

This property is affected when setting a Profile.

6.1.4 BitonalResolutionDPI

Property (get, set): Float BitonalResolutionDPI
Default: 200
License feature: Optimize

Get or set the target resolution in dots per inch (DPI) after downsampling images for bitonal images. See also ResolutionDPI.

Downsampling bitonal images is disabled if BitonalCompressions is set to eComprAttemptNone.

This property is affected when setting a Profile.

6.1.5 BitonalThresholdDPI

Property (get, set): Float BitonalThresholdDPI
Default: -1
License feature: Optimize

Get or set the threshold resolution in dots per inch (DPI) to selectively activate downsampling for bitonal images. The `-1` value deactivates downsampling for bitonal images. See also `ThresholdDPI`.

Downsampling bitonal images is disabled if `BitonalCompressions` is set to `eComprAttemptNone`.

This property is affected when setting a `Profile`.

6.1.6 ClipImages

Property (get, set): `Boolean ClipImages`
Default: `False`
License feature: `Optimize`

Get or set the option to clip images. When enabled, then invisible parts of images are clipped (cropped). While this does not affect visual parts of images, it may have a minor visual impact because clipped images are re-compressed. Pre-blended images are not clipped.

This property is affected when setting a `Profile`.

Enabling this property also enables the `OptimizeResources` property.

Regardless of this property's value, image clipping is disabled for those image types whose compression is set to `eComprAttemptNone`. See `BitonalCompressions` for bitonal images, `ContinuousCompressions` for color and grayscale images, and `IndexedCompressions` for indexed (palleted) images.

6.1.7 Close

Method: `Boolean Close()`

Close an opened input file. If the document is already closed, the method does nothing.

Returns:

`True` The file was closed successfully.

`False` Otherwise.

6.1.8 ColorCompression

[Deprecated] Property (get, set): `TPDFCompression ColorCompression`

Deprecated in Version 4.6. Use `ContinuousCompressions`.

6.1.9 ColorConversion

Property (get, set): `TPDFColorConversion ColorConversion`
Default: `eConvNone`
License feature: `Color`

Get or set the color conversion. Color conversion is applied to images. Image can be not converted (`eConvNone`), converted to RGB (`eConvRGB`), to CMYK or grayscale. Color key masked images are not color converted. Pre-blended images can be converted from RGB to grayscale if `ForceRecompression` is set to `True`.

This property is affected when setting a `Profile`.

Color conversion is mostly used for specific application areas. For example, in the printing industry, the CMYK color space is used, since it represents the colors that printer devices commonly support. If colors are exclusively used for the monitor, the RGB color space should be used.

Regardless of this property's state, color conversion is not done for image types whose compression is set to `eComprAttemptNone`. See `ContinuousCompressions` for color and grayscale images, and `IndexedCompressions` for indexed images.

6.1.10 ColorResolutionDPI

Property (get, set): `Float ColorResolutionDPI`
Default: `150`
License feature: `Optimize`

Get or set the target resolution in dots per inch (DPI) after downsampling images for color images. See also `ResolutionDPI`.

This property is affected when setting a `Profile`.

Downsampling color images is disabled if `ContinuousCompressions` is set to `eComprAttemptNone`.

6.1.11 ColorThresholdDPI

Property (get, set): `Float ColorThresholdDPI`
Default: `-1`
License feature: `Optimize`

Get or set the threshold resolution in dots per inch (DPI) to selectively activate downsampling for color images. The `-1` value deactivates downsampling for color images. See also `ThresholdDPI`.

This property is affected when setting a `Profile`.

Downsampling color images is disabled if `ContinuousCompressions` is set to `eComprAttemptNone`.

6.1.12 ContinuousCompressions

Property (get, set): TPDFComprAttempt ContinuousCompressions
Default: eComprAttemptNone

Get or set the compression types to be tried for continuous images, i.e. RGB, CMYK, and grayscale images. See also TPDFComprAttempt.

Several values can be combined with bitwise or operators. The following values are allowed:

- eComprAttemptNone
- eComprAttemptRaw
- eComprAttemptJPEG
- eComprAttemptFlate
- eComprAttemptJPEG2000
- eComprAttemptSource
- eComprAttemptMRC

Other values are ignored.

During optimization, all set compression types are tried and the one resulting in the least memory footprint is taken.

If ContinuousCompressions is set to eComprAttemptNone then color and grayscale images are excluded from processing. They are not downsampled (ColorResolutionDPI, ColorThresholdDPI, MonochromeResolutionDPI, MonochromeThresholdDPI), not color converted (ColorConversion), not clipped (ClipImages), and not reduced in color complexity (ReduceColorComplexity).

This property is affected when setting a Profile.

6.1.13 CompressionQuality

[Deprecated] Property (get, set): Single CompressionQuality

Deprecated. Use ImageQuality instead.

6.1.14 ConvertToCFF

Property (get, set): Boolean ConvertToCFF
Default: False
License feature: Optimize

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

This property is affected when setting a Profile.

6.1.15 DitheringMode

Property (get, set): TPDFDitheringMode DitheringMode
Default: eDitherNone

This option enables or disables dithering when downsampling bitonal images.

The only values supported are `eDitherNone` and `eDitherFloydSteinberg`.

This property is affected when setting a `Profile`.

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.

6.1.16 ErrorCode

Property (get): `TPDFErrorCode ErrorCode`

This property can be accessed to receive the latest error code. This value should only be read if a function call on the PDF Optimizer API has returned a value, which signales a failure of the function (see Error handling). See also enumeration `TPDFErrorCode`. Pdftools error codes are listed in the header file `bseerror.h`. Please note that only few of them are relevant for the 3-Heights® PDF Optimizer API.

6.1.17 ErrorMessage

Property (get): `String ErrorMessage`

Return the error message text associated with the last error (see property `ErrorCode`). This message can be used to inform the user about the error that has occurred. This value should only be read if a function call on the PDF Optimizer API has returned a value, which signals a failure of the function (see Error handling)

Note: Reading this property if no error has occurred can yield `Nothing` if no message is available.

.

6.1.18 ExtractFonts

Property (get, set): `Boolean ExtractFonts`
Default: `False`
License feature: `Optimize`

Get or set whether to extract embedded fonts. Depending on the font type, the extracted font has one of the following three formats: `fnt‹objno›.ttf`, `fnt‹objno›.pfb` or `fnt‹objno›.cff`, where `‹objno›` is the number of the PDF object of the font.

6.1.19 ExtractImages

Property (get, set): Boolean ExtractImages
Default: False

Get or set whether to extract images. Depending on the compression, the extracted image has one of the following formats: img‹objno›.tif or img‹objno›.jpg, where objno is the number of the PDF object of the image.

6.1.20 FlattenSignatureFields

Property (get, set): Boolean FlattenSignatureFields
Default: False
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 API removes all signatures, including parts (a) and (b).

When the FlattenSignatureFields property is set to True, 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.

6.1.21 ForceCompressionTypes

Property (get, set): Boolean ForceCompressionTypes
Default: False
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 ContinuousCompressions, BitonalCompressions, or IndexedCompressions. Use this option if you want to allow only the given compression types for images in the output PDF.

> This property is affected when setting a Profile.

6.1.22 ForceRecompression

Property (get, set): Boolean ForceRecompression
Default: False
License feature: Optimize

If set, images whose compression setting differs from eComprAttemptNone (see BitonalCompressions, ContinuousCompressions, IndexedCompressions) are always re-compressed. If not set (default), images are only re-compressed if the resulting image is smaller than the original, i.e. requires less bytes to store in the file.

This property is affected when setting a Profile.

6.1.23 GetInfoEntry

Method: String GetInfoEntry(String Key)

Get a string from the document's info dictionary or metadata.

Parameter:

Key [String] The key for which to get the value.

Returns:

The requested value or Nothing if the entry does not exist.

The value is extracted from the input document unless SetInfoEntry has been called previously with the same Key.

6.1.24 GetPdf

Method: Variant GetPdf()

Get the output file from memory. See also method SaveInMemory.

Returns:

A byte array containing the output PDF. In certain programming languages, such as Visual Basic 6, the type of the byte array must explicitly be Variant.

6.1.25 ImageStratConserv

[Deprecated] Property (get, set): Boolean ImageStratConserv
Default: False

Deprecated in Version 4.7.

Enables or disables a more conservative strategy for processing images. When enabled, the compression types set in BitonalCompressions, ContinuousCompressions, and IndexedCompressions are only tried if the image has either been clipped (ClipImages), downsampled (ResolutionDPI, ThresholdDPI), or if it has undergone a color conversion (ColorConversion). Otherwise, the original input image is taken as is. See also Provided features for optimizing images.

This property is affected when setting a Profile.

6.1.26 ImageQuality

Property (get, set): Single ImageQuality
Default: 75

Get or set the quality index of lossy compression types. This value ranges from 1 to 100 and is applied to JPEG and JPEG2000 compression. For JPEG2000, a quality index of 100 means lossless compression. JPEG compression is always lossy.

This property is affected when setting a Profile.

6.1.27 IndexedCompressions

Property (get, set): TPDFComprAttempt IndexedCompressions
Default: eComprAttemptFlate

Get or set the compression types for images that have an indexed ("palette") color space. See also TPDFComprAttempt.

Several values can be combined with bitwise or operators. The following values are allowed:

- eComprAttemptNone
- eComprAttemptRaw
- eComprAttemptFlate
- eComprAttemptLZW
- eComprAttemptSource

Other values are ignored.

During optimization, all set compression types are tried and the one resulting in the least memory footprint is taken.

If IndexedCompressions is set to eComprAttemptNone, then indexed images are excluded from processing. They are not clipped (ClipImages), not color converted (ColorConversion), and reduced in color complexity (ReduceColorComplexity).

This property is affected when setting a Profile.

6.1.28 LicenseIsValid

Property (get): Boolean LicenseIsValid
Static

Check if the license is valid.

6.1.29 Linearize

Property (get, set): `Boolean Linearize`
Default: `False`
License feature: `Optimize`

Note: This property is ignored when `AutoLinearize` is set to `True`.

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

Get or set whether to linearize the PDF output file, i.e. optimize file for fast web access.

The 3-Heights® PDF Optimizer API does not support linearization of PDF 2.0 documents. For such documents, processing fails. To automatically disable linearization for PDF 2.0, use `AutoLinearize`.

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.

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.

6.1.30 LinearizeFile

Method: `Boolean LinearizeFile(String FileName, String Password, String OutFileName, String UserPw, String OwnerPw, String PermissionFlags)`
License feature: `Optimize`

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

Linearize a PDF file and save the result as a new PDF file, which is optimized for fast web view. This is a standalone function and cannot be combined with any other functions or properties.

Parameters:

`FileName` [`String`] The input PDF file name, i.e. the name of document that is read.

Password [String] (optional) The user or owner password of the input file name. A password must be provided if the input file is protected by a user password, otherwise an empty string can be passed as argument.

OutFileName [String] The output PDF file name, i.e. the name of the linearized document that is written.

UserPw [String] (optional) The user password of the output PDF file.

OwnerPw [String] (optional) The owner password of the output PDF file.

PermissionFlags [String] (optional) The permission flags if the document is encrypted and secured by an owner password.

Additional information about UserPw, OwnerPw and PermissionFlags can be found in the SaveAs method.

Returns:

True A linearized PDF file was successfully created.

False Otherwise.

6.1.31 ListFonts

Method: Boolean ListFonts(String FileName)
License feature: Optimize

List all fonts included in the document and write them as a list to a text file

Parameter:

FileName [String] The file name of the output text file to which the information should be stored. The list contains the header line "FontName, FontType, Encoding, IsCID, IsEmbedded, IsSubsetted, Filename".

The meanings of these columns are:

FontName Name of the font, such as Arial-BoldMT or TimesNewRomanPS-BoldMT.

FontType Font type, such as TrueType or Type1

Encoding Encoding of the font, such as WinAnsiEncoding or MacRomanEncoding.

IsCID Font is CID (character identifier) keyed. This value is either CID or Non-CID.

IsEmbedded Font program for this font is embedded in the PDF document. This value is either Embedded or Non-Embedded.

FileName The file name of the font program. This is the name under which the font is saved to file in case the font is extracted and saved. Only embedded fonts can be extracted. The file name consists of the prefix fnt, the object number and the file type which is one of .ttf, .pfb or .cff.

Example: fnt38.ttf

Returns:

`True` The font information was successfully extracted and written to the output text file.

`False` Otherwise.

6.1.32 ListImages

Method: `Boolean ListImages(String FileName)`

List all images included in the document and write them as a list to a text file.

Parameter:

`FileName` `[String]` The file name of the output text file, to which the information should be stored. The list contains the header line:

```
PageNumber, ObjectNumber, Width, Height, BitsPerComponent, ColorSpace, Resolution, Filter, ImageSize, CompressedSize, CompressionRatio, FileName
```

The meanings of these columns are:

PageNumber Page number in the PDF on which the image occurs.

ObjectNumber Number of the PDF object that contains this image.

Width Width of the image in dots (pixels).

Height Height of the image in dots (pixels).

BitsPerComponent Number of bits that are used per component. This value is 1 for bitonal images and 8 for grayscale and color images.

ColorSpace The color space can be one of `DeviceGray`, `DeviceRGB`, `DeviceCMYK`, `ICCBased`, `Indexed`.

Resolution The ratio of number of pixels divided by the length of the image on the page.

Example: An image is 300 dots (pixel) wide and takes 1 inch (2.54cm) on the page in the PDF. This image has a resolution of 300 DPI (dots per inch). If the same image is stretched to 2 inch, its resolution is 150 DPI.

Filter The compression filter, for example: `FlateDecode`, `DCTDecode`, `CCITTFaxDecode`.

ImageSize Size in bytes of the uncompressed image.

CompressedSize Size in bytes of the compressed image.

CompressionRatio Ratio compressed size divided by uncompressed size.

FileName File name of the image. This is the name under which the image is saved to file in case the image is extracted and saved. The file name consists of the prefix `img`, followed by the PDF object number, and the extension which is one of `.jpg` or `.tif`, depending on the extracted image type. Example: `img19.jpg`, `img21.tif`.

Returns:

`True` The image information was successfully extracted and written to the output text file.

`False` Otherwise.

6.1.33 MergeEmbeddedFonts

Property (get, set): `Boolean MergeEmbeddedFonts`
Default: `False`
License feature: `Optimize`

Merge embedded font programs. 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 property is affected when setting a `Profile`.

6.1.34 MonochromeCompression

[Deprecated] Property (get, set): `TPDFCompression MonochromeCompression`

Deprecated in Version 4.6, use `ContinuousCompressions`.

6.1.35 MonochromeResolutionDPI

Property (get, set): `Float MonochromeResolutionDPI`
Default: `150`
License feature: `Optimize`

Get or set target resolution in dots per inch (DPI) after downsampling images for monochrome (grayscale) images. See also `ResolutionDPI`.

Note that downsampling monochrome images is disabled if `ContinuousCompressions` is set to `eComprAttemptNone`.

This property is affected when setting a `Profile`.

6.1.36 MonochromeThresholdDPI

Property (get, set): `Float MonochromeThresholdDPI`
Default: `-1`
License feature: `Optimize`

Get or set the threshold resolution in dots per inch (DPI) to selectively activate downsampling for monochrome (grayscale) images. The `-1` value deactivates downsampling for monochrome images. See also `ThresholdDPI`.

Note that downsampling monochrome images is disabled if `ContinuousCompressions` is set to `eComprAttemptNone`.

This property is affected when setting a `Profile`.

6.1.37 MrcLayerCompression

Property (get, set): `TPDFCompression MrcLayerCompression`
Default: `eComprJPEG2000`
License feature: `Optimize`

Get or set the compression type for MRC foreground and background layers. See `TPDFCompression` for possible values. See also MRC optimization for images.

This property is affected when setting a `Profile`.

6.1.38 MrcLayerQuality

Property (get, set): `Short MrcLayerQuality`
Default: `10`
License feature: `Optimize`

Get or set the image quality for MRC foreground and background layers when using a lossy compression type. This is a value between `0` and `100`.

See also Supported image compression types, Relevant factors for file size, and MRC optimization for images.

This property is affected when setting a `Profile`.

6.1.39 MrcLayerResolutionDPI

Property (get, set): `Float MrcLayerResolutionDPI`
Default: `70`
License feature: `Optimize`

Get or set the target resolution in DPI for downsampling MRC foreground and background layers. If set to `-1` then no downsampling is performed. See also Provided features for optimizing images and MRC optimization for images.

This property is affected when setting a `Profile`.

6.1.40 MrcMaskCompression

Property (get, set): `TPDFCompression MrcMaskCompression`
Default: `eComprGroup4`
License feature: `Optimize`

Get or set the compression type for MRC masks. See `TPDFCompression` for possible values. See also MRC optimization for images.

This property is affected when setting a `Profile`.

6.1.41 MrcPictCompression

Property (get, set): `TPDFCompression MrcPictCompression`
Default: `eComprJPEG`
License feature: `Optimize`

Get or set the compression type for MRC cut-out pictures. See `TPDFCompression` for possible values. When performing MRC, this is the compression applied to monochrome (grayscale) images because they are treated as cut-out pictures. See also MRC optimization for images.

This property is affected when setting a `Profile`.

6.1.42 MrcRecognizePictures

Property (get, set): `Boolean MrcRecognizePictures`
Default: `True`
License feature: `Optimize`

Get or set the option to recognize photographic regions when doing MRC. When set to `False`, then Stage 1: Cutting out pictures is omitted, i.e., no regions of photographic content are identified.

This property is affected when setting a `Profile`.

Regardless of this property's setting, monochrome (grayscale) images are always treated as entire photographic regions (cut-out pictures) by the MRC algorithm. See also MRC optimization for images.

6.1.43 Open

Method: `Boolean Open(String Filename, String Password)`

Open a PDF file, i.e. make the objects contained in the document accessible. If another document is already open, it is closed first.

Parameters:

`Filename` [`String`] The file name and optionally, the file path, drive or server string according to the operating systems file name specification rules.

Password [String] (optional) The user or the owner password of the encrypted PDF document. If this parameter is left out, an empty string is used as a default.

Returns:

True The file could be successfully opened.

False The file does not exist, it is corrupt, or the password is not valid. Use the ErrorCode and ErrorMessage properties for additional information.

6.1.44 OpenMem

Method: Boolean OpenMem(Variant MemBlock, String Password)

Open a PDF file, i.e. make the objects contained in the document accessible. If a document is already open, it is closed first.

Parameters:

MemBlock [Variant] The memory block containing the PDF file given as a one-dimensional byte array.

Password [String] (optional) The user or the owner password of the encrypted PDF document. If this parameter is left out, an empty string is used as a default.

Returns:

True The document could be successfully opened.

False The document could not be opened, it is corrupt, or the password is not valid.

6.1.45 OpenStream

Method: Boolean OpenStream(Variant Stream, String Password)

Open a PDF file, i.e. make the objects contained in the document accessible. If a document is already open, it is closed first.

Parameters:

Stream [Variant] The stream providing the PDF file. The stream must support random access.

Password [String] (optional) The user or the owner password of the encrypted PDF document. If this parameter is left out, an empty string is used as a default.

Returns:

`True` The document could be successfully opened.

`False` The document could not be opened, it is corrupt, or the password is not valid.

6.1.46 OptimizeResources

Property (get, set): `Boolean OptimizeResources`
Default: `False`
License feature: `Optimize`

Get or set whether resources should be optimized. If set, unused resources such as images, fonts, and color spaces are removed. Also content streams are re-built.

This property is affected when setting a `Profile`.

6.1.47 PageCount

Property (get): `Long PageCount`

Get the number of total pages of the document. If no document is opened, it returns `0`.

6.1.48 ProductVersion

Property (get): `String ProductVersion`

Get the version of the 3-Heights® PDF Optimizer API in the format "A.C.D.E".

6.1.49 Profile

Property (set): `TPDFOptimizationProfile Profile`
Default: `eOptimizationProfileDefault`

With this property, one of the predefined optimization profiles can be set. If a profile is set, then all the properties listed in `TPDFOptimizationProfile` (table Profile settings) are set to their respective values. Properties not listed in this table are left unchanged.

One way of quickly arriving at a specific setting is to first set the `Profile` and then adapt the configuration by setting some of the individual properties.

Example: In C#.

```
using (Optimizer opt = new Optimizer())
```

```
{
    opt.Profile = PDFOptimizationProfile.eOptimizationProfileWeb;
    opt.Linearize = false;
    // ... perform some optimization now
}
```

6.1.50 ReduceColorComplexity

Property (get, set): Boolean ReduceColorComplexity
Default: False
License feature: Color

This property 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 property is affected when setting a Profile.

- 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 eComprAttemptNone. See ContinuousCompressions and IndexedCompressions.

6.1.51 RemoveImages

Property (get, set): Boolean RemoveImages
Default: False
License feature: Optimize

When enabling this property, images and stencil masks are substituted by empty form XObjects. Inline images are left untouched. Enabling this option disables all other image-related processing.

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

6.1.52 RemoveNonSymbolicFonts

[Deprecated] Property (get, set): Boolean RemoveNonSymbolicFonts
Default: False

This property has no effect.

6.1.53 RemoveRedundantObjects

Property (get, set): `Boolean RemoveRedundantObjects`
Default: `False`
License feature: `Optimize`

Get or set whether redundant objects should be removed. If this property is set to `True`, duplicate objects are removed to reduce the file size.

This property is affected when setting a `Profile`.

6.1.54 RemoveStandardFonts

Property (get, set): `Boolean RemoveStandardFonts`
Default: `False`
License feature: `Optimize`

Get or set whether to remove 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 property is affected when setting a `Profile`.

A PDF Viewer must be able to display these 14 PDF standard fonts correctly. Therefore, enabling this property 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 API. 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

6.1.55 ResolutionDPI

Property (get, set): `Single ResolutionDPI`
Default: (Different defaults apply to different image types)
License feature: `Optimize`

Get or set the resolution in DPI (dots per inch) after re-sampling images, image masks and image's soft masks.

This property affects all three image compression types (`BitonalResolutionDPI`, `ColorResolutionDPI`, `MonochromeResolutionDPI`).

This property is affected when setting a `Profile`.

A typical value for the resolution when optimizing for the web is `150` DPI. For printing typically no re-sampling is applied (see property `ThresholdDPI`). Pre-blended images, images with a color key mask, mask, and soft mask images are not re-sampled.

When getting `ResolutionDPI`, the property returns `ColorResolutionDPI`.

Note that downsampling images is disabled for indexed images and for those images whose compression type is set to `eComprAttemptNone`. See `BitonalCompressions`, `ContinuousCompressions`.

6.1.56 SaveAs

Method: `Boolean SaveAs(String FileName, String UserPw, String OwnerPw, TPDFPermission PermissionFlags)`

Save the currently opened document.

Parameters:

`FileName` `[String]` The file name and optionally the file path, drive or server string according to the operating systems file name specification rules.

`UserPw` `[String]` (optional) Set the user password of the PDF document. If this parameter is omitted, the default password is used. Use `""` to set no password.

OwnerPw [String] (optional) Set the owner password of the PDF document. If this parameter is omitted, the default password is used. Use "" to set no password.

PermissionFlags [TPDFPermission] (optional) The permission flags.

By default no encryption is used (-1). The permissions that can be granted are listed at the enumeration TPDFPermission. To not encrypt the output document, set PermissionFlags to ePermNoEncryption, user and owner password to "". In order to allow high quality printing, flags ePermPrint and ePermDigitalPrint need to be set.

Returns:

True The opened document could successfully be saved to file.

False Otherwise. One of the following occurred[7]:

- The output file cannot be created.
- PDF_E_FILECREATE: Failed to create the file.

6.1.57 SaveInMemory

Method: Boolean SaveInMemory()

Save the output PDF in memory. After the Close call, it can be accessed using the GetPdf method.

Returns:

True The document could be saved in memory successfully.

False Otherwise.

6.1.58 SaveAsStream

Method: Boolean SaveAsStream(Variant Stream, String UserPw, String OwnerPw)

Parameter:

Stream [Variant] The stream the output file is written to. The stream must support read, write, and random access.

All other parameters and the return value are identical to the SaveAs method.

[7] This is not a complete list. If SaveAs returns False, it is recommended to abort the processing of the file and log the error code and error message.

6.1.59 SetCMSEngine

Method: `Boolean SetCMSEngine(String CMSEngine)`
License feature: `Color`

Set the Color Management System (CMS) Engine. The following strings are supported:

`"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` 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):

```
0.996078, 0.996078, 0.996078 ; White
0.000000, 0.686275, 0.937255 ; C
0.925490, 0.149020, 0.560784 ; M
1.000000, 0.949020, 0.066667 ; Y
0.215686, 0.203922, 0.207843 ; K
0.243137, 0.247059, 0.584314 ; CM
0.000000, 0.658824, 0.349020 ; CY
0.066667, 0.176471, 0.215686 ; CK
0.929412, 0.196078, 0.215686 ; MY
0.215686, 0.101961, 0.141176 ; MK
0.200000, 0.196078, 0.125490 ; YK
0.266667, 0.266667, 0.274510 ; CMY
0.133333, 0.098039, 0.160784 ; CMK
0.074510, 0.180392, 0.133333 ; CYK
0.215686, 0.121569, 0.113725 ; MYK
0.125490, 0.121569, 0.121569 ; 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.

6.1.60 SetInfoEntry

Method: `Boolean SetInfoEntry(String Key, String Value)`

Set a key-value pair in the document info dictionary. Values of predefined keys are also stored in the XMP metadata.

Popular entries specified in the PDF Reference 1.7 and accepted by most PDF viewers are `"Title"`, `"Author"`, `"Subject"`, `"Creator"` (sometimes referred to as Application), and `"Producer"` (sometimes referred to as PDF Creator).

Parameters:

`Key` `[String]` The key as a string.

`Value` `[String]` The value as a string.

Example in C#:

```
opt.SetInfoEntry("Producer", "Me, myself, and I");
```

6.1.61 SetLicenseKey

Method: `Boolean SetLicenseKey(String LicenseKey)`

Sets the license key.

6.1.62 SetVersion

Method: `Boolean SetVersion(String PDFVersion)`

Set the minimum PDF version of the created PDF output file. Supported values for the string are `"1.1"` to `"1.7"`[8]. There are three parameters that influence the version of the PDF output file:

- The value set using this method
- The PDF version of the input file
- Other optimizer settings (e.g. 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: Input PDF is version 1.5 and the following settings are applied

```
SetVersion("1.4")
```

The output file is PDF version 1.5.

Example: Input PDF is version 1.4 or lower and the following settings are applied

```
SetVersion("1.4")
```

The output file is PDF version 1.4.

Example: Input PDF is version 1.3 and the following settings are applied

```
ColorCompression = eComprJPEG2000
```

[8] PDF 1.4 corresponds to Acrobat version 5, PDF 1.5 to Acrobat version 6, etc.

```
SetVersion("1.4")
```

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.

6.1.63 Strip

Property (get, set): `TPDFStripType Strip`
Default: `0`
License feature: `Optimize`

Get or set the stripping mode. This mode can be configured to remove unneeded data of a PDF document such as threads, metadata, the PieceInfo, the StructTreeRoot entry, embedded Thumbs and the SpiderInfo entry. Also this mode is used to indicate whether to flatten form fields, links, and other annotations. Multiple values of `TPDFStripType` can be combined with the bitwise or operator.

This property is affected when setting a `Profile`.

6.1.64 SubsetFonts

Property (get, set): `Boolean SubsetFonts`
Default: `false`
License feature: `Optimize`

This property influences two optimizations related to subsetted fonts:

- Subset embedded fonts.
- Merge embedded font programs of different subsets of the same font, provided they can be merged.

Sub-setting refers to removing those glyphs in a font that are not actually used in any text contained in the PDF.

This property is affected when setting a `Profile`.

6.1.65 ThresholdDPI

Property (get, set): `Single ThresholdDPI`
Default: `-1`
License feature: `Optimize`

Set the threshold in DPI (dots per inch) to selectively activate re-sampling. Only images with a resolution above the threshold DPI will be re-sampled.

The value `-1` deactivates re-sampling.

This property affects all three image compression types (`BitonalThresholdDPI`, `ColorThresholdDPI`, `MonochromeThresholdDPI`).

This property is affected when setting a `Profile`.

A typical value for the threshold when optimizing for the web is 210 DPI. For printing typically no re-sampling is applied.

When getting ThresholdDPI, the property returns ColorThresholdDPI.

Note that downsampling images is disabled for indexed images and for those images whose compression type is set to eComprAttemptNone. See BitonalCompressions, ContinuousCompressions.

6.1.66 UnembedFont

Method: Boolean UnembedFont(String FontName)
License feature: Optimize

Remove the embedded font program for the font given in FontName.

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

6.2 Enumerations

> **Note:** Depending on the interface, enumerations may have TPDF as prefix (COM, C), PDF as prefix (.NET), or no prefix at all (Java).

6.2.1 TPDFColorConversion Enumeration

TPDFColorConversion table

TPDFColorConversion	Description	License feature
Const eConvNone	None	Optimize, Color
Const eConvRGB	Red Green Blue	Optimize, Color
Const eConvCMYK	Cyan Magenta Yellow Key	Optimize, Color
Const eConvGray	Gray	Optimize, Color

6.2.2 TPDFComprAttempt Enumeration

In contrast to TPDFCompression, TPDFComprAttempt is meant to be used as a bit-field, i.e. values can be composed with the bitwise or operator (Or in Visual Basic). Use this enumeration to compose values for the BitonalCompressions, ContinuousCompressions, and IndexedCompressions properties.

TPDFComprAttempt table

TPDFComprAttempt	Description	License feature
eComprAttemptNone	Exclude from processing	
eComprAttemptRaw	No compression (raw)	Optimize, Color
eComprAttemptJPEG	DCT (JPEG)	Optimize, Color
eComprAttemptFlate	Flate (ZIP)	Optimize, Color
eComprAttemptLZW	LZW	Optimize, Color
eComprAttemptGroup3	CCITT Fax Group 3 (CCITT Fax Group 3 and 4)	Optimize, Color
eComprAttemptGroup3_2D	CCITT Fax Group 3 2D (CCITT Fax Group 3 and 4)	Optimize, Color
eComprAttemptGroup4	CCITT Fax Group 4 (CCITT Fax Group 3 and 4)	Optimize, Color
eComprAttemptJBIG2	JBIG2 (Supported in PDF 1.4 or later)	Optimize, Color
eComprAttemptJPEG2000	JPEG2000 (Supported in PDF 1.5 or later, not supported in PDF/A-1)	Optimize, Color
eComprAttemptMRC	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.	Optimize
eComprAttemptSource	In contrast to the values 0-8, this is not a single compression format. Instead, this tells 3-Heights® PDF Optimizer API to use the same compression as the original input image.	Optimize, Color

6.2.3 TPDFCompression Enumeration

Compression types as occurring in PDF.

Note: Not all image formats/color depths support all compression types. See also Supported image compression types.

TPDFColorConversion table

TPDFColorConversion	Description
eComprRaw	No compression
eComprJPEG	Joint Photographic Expert Group
eComprFlate	Flate compression

TPDFColorConversion table

eComprLZW	Lempel-Ziv-Welch
eComprGroup3	CCITT Fax Group 3
eComprGroup3_2D	CCITT Fax Group 3 2D
eComprGroup4	CCITT Fax Group 4
eComprJBIG2	Joint Bi-level Image Experts Group
eComprJPEG2000	JPEG2000
eComprUnknown	Unknown compression

6.2.4 TPDFErrorCode Enumeration

All TPDFErrorCode enumerations start with a prefix, such as PDF_, followed by a single letter which is one of S, E, W or I, an underscore, and a descriptive text.

The single letter gives an indication of the severity of the error. These are: Success, Error, Warning, and Information. In general, an error is returned if an operation could not be completed, e.g. no valid output file was created. A warning is returned if the operation was completed, but problems occurred in the process.

A list of all error codes is available in the C API header file bseerror.h, the javadoc documentation of com.pdftools.NativeLibrary.ERRORCODE, and the .NET documentation of Pdftools.Pdf.PDFErrorCode. Note that only a few are relevant for the 3-Heights® PDF Optimizer API, most of which are listed here:

TPDFErrorCode table

TPDFErrorCode	Description
PDF_S_SUCCESS	The operation was completed successfully.
LIC_E_NOTSET, LIC_E_NOTFOUND, ...	Various license management related errors.
PDF_E_FILEOPEN	Failed to open the file.
PDF_E_FILECREATE	Failed to create the file.
PDF_OPT_E_ANNOTAPPEAR	Cannot create appearance for annotation.
PDF_OPT_W_RMSIGANNOT	A signature annotation was removed.
PDF_W_NOENCRYPTION	The file is PDF/A and must not be encrypted.

6.2.5 TPDFOptimizationProfile Enumeration

TPDFOptimizationProfile

TPDFOptimizationProfile	Description
`eOptimizationProfileDefault`	Minimal optimization. This is the default profile.
`eOptimizationProfileWeb`	Optimization for the Internet
`eOptimizationProfilePrint`	Optimization for print
`eOptimizationProfileArchive`	Optimization for archiving purposes
`eOptimizationProfileMax`	Optimization for maximum memory size reduction
`eOptimizationProfileMRC`	MRC (Mixed Raster Content) optimization of images. See also MRC optimization for images.

Note: When setting a profile with the `Profile` property, then all the properties listed in Profile settings are set to their respective value. Values not listed are left unchanged.
Also note that gray 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 `eOptmizationProfileDefault` profile.

Profile settings

	eOptimizationProfileDefault	eOptimizationProfileWeb	eOptimizationProfilePrint	eOptimizationProfileArchive	eOptimizationProfileMax	eOptimizationProfileMRC
BitonalCompressions:						
eComprAttemptNone	✓					✓
eComprAttemptGroup4		✓	✓	✓	✓	
eComprAttemptJBIG2		✓		✓	✓	
eComprAttemptSource		✓	✓	✓	✓	
ContinuousCompressions:						
eComprAttemptNone	✓					
eComprAttemptJPEG		✓	✓	✓	✓	
eComprAttemptFlate		✓	✓	✓	✓	
eComprAttemptJPEG2000		✓		✓	✓	
eComprAttemptMRC						✓
eComprAttemptSource		✓	✓	✓	✓	
IndexedCompressions:						
eComprAttemptNone	✓					✓
eComprAttemptFlate		✓	✓	✓	✓	
eComprAttemptLZW					✓	
eComprAttemptSource		✓	✓	✓	✓	
BitonalResolutionDPI	(200)[9]	200	(200)[9]	(200)[9]	160	(200)[9]
BitonalThresholdDPI	-1	280	-1	-1	220	-1
MonochromeResolutionDPI	(150)[9]	150	(150)[9]	(150)[9]	130	(150)[9]
MonochromeThresholdDPI	-1	210	-1	-1	180	-1
ColorResolutionDPI	(150)[9]	150	(150)[9]	(150)[9]	130	(150)[9]

Profile settings

	eOptimizationProfileDefault	eOptimizationProfileWeb	eOptimizationProfilePrint	eOptimizationProfileArchive	eOptimizationProfileMax	eOptimizationProfileMRC
ColorThresholdDPI	-1	210	-1	-1	180	-1
ImageQuality	75	75	80	80	70	75
ColorConversion	None	RGB	CMYK	None	RGB	RGB
ClipImages	False	True	True	True	True	True
ReduceColorComplexity	False	True	True	True	True	False
ForceRecompression	False					
ForceCompressionTypes	False					
DitheringMode	eDitheringNone					
MrcLayerCompression	eComprJPEG2000					
MrcLayerResolutionDPI	70					
MrcMaskCompression	eComprGroup4					
MrcLayerQuality	20					
MrcRecognizePictures	True					
MrcPictCompression	eComprJPEG					
ConvertToCFF	False	True	True	True	True	False
MergeEmbeddedFonts	False	True	True	True	True	True
RemoveStandardFonts	False	False	False	False	True	False
SubsetFonts	False	True	True	True	True	True
OptimizeResources	False	True	True	True	True	True
Linearize	False	False	False	False	False	False
AutoLinearize	False	True	False	False	False	False
RemoveRedundantObjects	False	True	True	True	True	True

Profile settings

	eOptimizationProfileDefault	eOptimizationProfileWeb	eOptimizationProfilePrint	eOptimizationProfileArchive	eOptimizationProfileMax	eOptimizationProfileMRC
Strip:						
eStripThreads		✓	✓		✓	✓
eStripMetadata		✓			✓	
eStripPieceInfo		✓	✓		✓	✓
eStripStructTree		✓	✓		✓	✓
eStripThumb		✓	✓	✓	✓	✓
eStripSpider		✓	✓		✓	✓
eStripAlternates		✓		✓	✓	
eStripOutputIntents					✓	
eStripAnnots					✓	
eStripForms					✓	
eStripLinks						

6.2.6 TPDFPermission Enumeration

An enumeration for permission flags. If a flag is set, the permission is granted.

TPDFPermission table

TPDFPermissionFlag	Description
ePermNoEncryption	Do not apply encryption. This enumeration value cannot be combined with other values. When using this enumeration, set both passwords to an empty string or Nothing.

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

TPDFPermission table

ePermSameAsInput	Use the same permissions as present in the input file. This enumeration value cannot be combined with other values.
ePermPrint	Low resolution printing
ePermModify	Changing the document
ePermCopy	Content copying or extraction
ePermAnnotate	Annotations
ePermFillForms	Filling of form fields
ePermSupportDisabilities	Support for disabilities
ePermAssemble	Document assembly
ePermDigitalPrint	High resolution printing
ePermAll	Grant all permissions

Changing permissions or combining multiple permissions is done using a bitwise "or" operator.

> **Note:** The special values ePermSameAsInput and ePermNoEncryption cannot be combined with any other values.

Changing the current permissions in Visual Basic should be done like this:

Allow Printing

```
Permission = Permission Or ePermPrint
```

Prohibit Printing

```
Permission = Permission And Not ePermPrint
```

To disable encryption, you should overwrite all flags:

```
Permission = ePermNoEncryption
```

6.2.7 TPDFStripType Enumeration

TPDFStripType table

TPDFStripType	Description
eStripThreads	Strip article threads
eStripMetadata	Strip metadata

TPDFStripType table

eStripPieceInfo	Strip page piece info (private application data)
eStripStructTree	Strip document structure tree (incl. Mark-up)
eStripThumb	Strip thumbnails
eStripSpider	Strip spider (web capture) info
eStripAlternates	Strip alternate images
eStripForms	Strip and flatten form fields
eStripLinks	Strip and flatten link annotations
eStripAnnots	Strip and flatten annotations except form fields and links
eStripFormsAnnots	Strip and flatten form fields, links, and annotations. This implies eStripForms, eStripLinks, eStripAnnots
eStripInvisibleAnnots	Strip invisible annotations
eStripOutputIntents	Strip the document output intents
eStripAll	Strip everything (all of the above)

7 Version history

Some of the documented changes below may be preceded by a marker that specifies the interface technologies the change applies to. For example, [C, Java] applies to the C and the Java interface.

7.1 Changes in versions 6.19–6.27

- **Update** license agreement to version 2.9

7.2 Changes in versions 6.13–6.18

- [.NET, C, Java] **New** methods `OpenStream` and `SaveAsStream` to facilitate the use of streams as input and output.

7.3 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".
- [Java] **Changed** minimal supported Java language version to 7 [previously 6].
- [PHP] **Removed** all versions of the PHP interface.
- [.NET] **New** availability of this product as NuGet package for Windows, macOS and Linux.
- [.NET] **New** support for .NET Core versions 1.0 and higher. The support is restricted to a subset of the operating systems supported by .NET Core, see Operating systems.
- [.NET] **Changed** platform support for NuGet packages: The platform "AnyCPU" is now supported for .NET Framework projects.

7.4 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 `eOptimizationProfileMRC`: Now includes color conversion to RGB.
- **Changed** behavior: Invisible annotations are not removed anymore by default.
- **New** enum value `eStripInvisibleAnnots` to remove invisible annotations.
- **New** additional supported operating system: Windows Server 2019.
- [PHP] **New** extension PHP 7.3 (non thread safe) for Linux.
- [.NET, C, COM, Java, PHP] **New** method `GetInfoEntry` for extracting Info-dictionary entries.

7.5 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.
- [.NET, C, COM, Java, PHP] **New** property `AutoLinearize` to automatically choose whether to linearize the output document or not.
- [.NET, C, COM, Java, PHP] **Changed** property `Profile`: In the `TPDFOptimizationProfile.eOptimizationProfileWeb` optimization profile, linearization (`Linearize` property) is substituted by auto linearization (`AutoLinearize` property).
- **Changed** values of the error codes `PDF_E_RICHTEXT` and `PDF_W_RICHTEXT`.

7.6 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.
- [.NET, C, COM, Java, PHP] **Changed** enum `TPDFPermission`: Added a new value `ePermSameAsInput` to adopt the encryption parameters from the input document.
- [.NET, C, COM, Java, PHP] **New** property `ErrorMessage`.

7.7 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.
- [C] **Clarified** Error handling of `TPdfStreamDescriptor` functions.
- [.NET, C, COM, Java, PHP] **Changed** enum `TPDFOptimizationProfile`: Added a new profile `eProfileArchive` to optimize for archiving purposes.

7.8 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.
- [C] **Changed** return value `pfGetLength` of `TPDFStreamDescriptor` to `pos_t`[10].
- [PHP] **New** Interface for Windows and Linux. Supported versions are PHP 5.6 & 7.0 (Non Thread Safe). The `PdfOptimizeAPI` PHP Interface is contained in the 3-Heights® PDF Tools PHP5.6 Extension and the 3-Heights® PDF Tools PHP7.0 Extension.
- [C] **Changed** 32-bit binaries on Windows that link to the API need to be recompiled due to a change of the used mangling scheme.
- [.NET, C, COM, Java] **New** property `FlattenSignatureFields`: Allows to flatten the visual appearance of digital signatures. (During optimization, digital signatures are always removed.)
- [.NET, C, COM, Java] **Changed** property `Profile`: Setting this property to `eOptimizationProfileMax` sets the property `ColorConversion` to `eConvRGB`.
- [.NET, C, COM, Java] **New** property `RemoveImages`: Remove images by replacing them with empty XObjects.

7.9 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.
- [.NET, C, COM, Java] **New** property `ReduceColorComplexity`: Reduce color complexity for images.
- [.NET, C, COM, Java] **Changed** property `Profile`: The profiles `eOptimizationProfileWeb, eOptimizationProfilePrint, andeOptimizationProfileMax` now include the property `ReduceColorComplexity`.
- [.NET, C, COM, Java] **New** property `ProductVersion` to identify the product version.
- [.NET] **Deprecated** method `GetLicenseIsValid`.
- [.NET] **New** property `LicenseIsValid`.

[10] This has no effect on neither the .NET, Java, nor COM API

8 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.

Licensing and copyright The 3-Heights® PDF Optimizer API is copyrighted. This user manual is also copyright protected; It may be copied and distributed provided that it remains unchanged including the copyright notice.

Contact

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