

User Manual

OPTATIS

Optatis / Nullam quisque id. Vestibulum est. Vestibulum est.

Utriusque
Est ut Magnatur
Vestibulum est. Vestibulum est.

15. September 2000

providenda into illud

Est ut Magnatur

et magnatur? Quia qui disqui veritem sit ut cubusa accanum quam num carum fugiae audi-

re nunt acilique dessequi necusam qui blatusdae quam dolo vit earum quae voluptatur?

est maximus ut maximet. sequam, tectae lanemol oportibus dolaputem ducipunt.

hanc volut apocpa apiclis quam exora cupiclus rem et repcece proem videt que-

offic: te recum acitis simpoeniam stitem incedem fugiat.

quam return, que ped ut aut aut maseque officis ut lauto am ius rem doltinas

te esti autali bleecat.

vorior sine laud quia vellandit: et verum quo ea quia: qui asperatur, excoliqui-

met opta quoms eni sequeare velam, veliqui scemla turli ut ex exequitet

ur at exetum vili fluffate sem ut aplique compta cemeti qui verum

trucha ut aut endio dolendunt, emis onest, sile a som am fugio

am ret anibus auditas nobit optequum extanentia ignes modas

ita maxim quam: volenti onsequi dolaprobunt rem utata quo cum

te pono vero omnilla sus pliquo inchi quo quadicum, ipsanta

te mperibus evlentis shatem at

aptur, vidempes even aplupit or

sc. entest

pro

ty

ty

3-Heights® PDF to Image Converter API

Rendering Engine 2.0

Version 6.27.6



Contents

1	Introduction	5
1.1	Description	5
1.2	Functions	5
1.2.1	Features	5
1.2.2	Formats	5
1.3	Interfaces	6
1.4	Operating systems	6
1.5	How to best read this manual	7
2	Installation and deployment	8
2.1	Windows	8
2.2	Linux and macOS	8
2.2.1	Linux	9
2.2.2	macOS	9
2.3	ZIP archive	10
2.3.1	Development	10
2.3.2	Deployment	11
2.4	NuGet package	12
2.5	Interface-specific installation steps	13
2.5.1	COM interface	13
2.5.2	Java interface	13
2.5.3	.NET interface	14
2.5.4	C interface	14
2.6	Uninstall, Install a new version	14
2.7	Note about the evaluation license	15
2.8	Special directories	15
2.8.1	Directory for temporary files	15
2.8.2	Cache directory	15
2.8.3	Font directories	16
3	License management	17
4	Programming interfaces	18
4.1	Visual Basic 6	18
4.2	ASP	19
4.3	.NET	19
4.3.1	Visual Basic	20
4.3.2	C#	21
4.3.3	Deployment	21
4.3.4	Troubleshooting: TypeInitializationException	21
5	User guide	23
5.1	Supported codecs	23
5.1.1	File and compression type	23
5.2	Creating multi and single-page images	24
5.2.1	Multi-page images	24
5.2.2	Single-page images	24
5.3	Setting pixels equal points	24
5.4	Reducing the file size	25

5.4.1	Dimensions	25
5.4.2	Resolution	25
5.4.3	Bits per pixel	25
5.4.4	Format/Compression type	26
5.4.5	Image content, dithering	27
5.5	Using the in-memory methods	27
5.5.1	Creating a document in memory	27
5.5.2	Reading a document from memory	28
5.6	Color profiles	28
5.6.1	Default color profiles	28
5.6.2	Set other color profiles	29
5.6.3	Get other color profiles	29
5.7	Fonts	29
5.7.1	Font cache	29
5.7.2	Microsoft core fonts on Linux or macOS	29
5.7.3	Font configuration file fonts.ini	30
5.8	Changing the colors—obtaining a darker black	31
5.9	Applying isomorphic stretching	32
5.10	Dithering	32
5.10.1	Remarks	32
5.10.2	Color images	33
5.10.3	Bitonal images	34
5.10.4	Guidelines	36
5.11	Error handling	37
6	Interface reference	38
6.1	Pdf2Img Interface	38
6.1.1	BilevelThreshold	38
6.1.2	BitmapHeight	38
6.1.3	BitmapWidth	38
6.1.4	BitsPerPixel	38
6.1.5	Center	39
6.1.6	ClearRenderingProperties	39
6.1.7	Close	39
6.1.8	CloseImage	39
6.1.9	ColorSpace	40
6.1.10	Compression	40
6.1.11	ConvertFile	40
6.1.12	CreateImage	41
6.1.13	CreateImageInMemory	41
6.1.14	Dithering	42
6.1.15	DPI	42
6.1.16	ErrorCode	42
6.1.17	ErrorMessage	42
6.1.18	FaxHSetting, FaxSSetting	43
6.1.19	FillOrder	43
6.1.20	FilterRatio	43
6.1.21	FitPage	43
6.1.22	GetImage	44
6.1.23	GetOcg	44
6.1.24	HasColor	44
6.1.25	ImageQuality	44

6.1.26	LicenseIsValid	45
6.1.27	OcgCount	45
6.1.28	Open	45
6.1.29	OpenMem	45
6.1.30	OpenStream	46
6.1.31	Options	46
6.1.32	Options2	46
6.1.33	PageBoxType	47
6.1.34	PageCount	47
6.1.35	PageHeight	47
6.1.36	PageWidth	47
6.1.37	PageNo	47
6.1.38	PageXOffs, PageYOffs	48
6.1.39	PreserveAspectRatio	48
6.1.40	ProductVersion	48
6.1.41	Quality	48
6.1.42	RenderingMode	48
6.1.43	RenderPage	48
6.1.44	RotateMode	49
6.1.45	SetBitmapDimensions	49
6.1.46	SetCMSEngine	49
6.1.47	SetCMYKProfile	50
6.1.48	SetLicenseKey	50
6.1.49	SetPageSize	51
6.1.50	SetRenderingProperty	51
6.1.51	SetsRGBProfile	51
6.1.52	UnsupportedFeatures	52
6.1.53	XDPI, YDPI	52
6.2	Ocg Interface	52
6.2.1	Label	52
6.2.2	Level	53
6.2.3	Name	53
6.2.4	Visible	53
6.3	Enumerations	54
6.3.1	TPDFColorSpace Enumeration	54
6.3.2	TPDFCompression Enumeration	54
6.3.3	TPDFDithering Enumeration	55
6.3.4	TPDFErrorCode Enumeration	55
6.3.5	TPDFPageBox Enumeration	56
6.3.6	TPDFRendererOption2 Enumeration	57
6.3.7	TPDFRenderingProperty Enumeration	57
6.3.8	TPDFRotateMode Enumeration	58
7	Tips, tricks, and troubleshooting	59
7.1	Font and text issues	59
7.1.1	Handle non-embedded fonts	59
	Font replacement strategy	59

8 Version history 60

8.1 Changes in versions 6.19–6.27 60

8.2 Changes in versions 6.13–6.18 60

8.3 Changes in versions 6.1–6.12 60

8.4 Changes in version 5 60

8.5 Changes in version 4.12 60

8.6 Changes in version 4.11 60

8.7 Changes in version 4.10 61

8.8 Changes in version 4.9 61

8.9 Changes in version 4.8 61

9 Licensing, copyright, and contact 62

1 Introduction

1.1 Description

The 3-Heights® PDF to Image Converter API converts PDF documents into single-page or multi-page raster images such as TIFF or JPEG. It can also convert PDF files into rasterized PDF.

Its areas of use include the web, TIFF-based DMS solutions, archiving and workflow systems, and the protection of PDF documents. The converter is characterized by its high speed and outstanding quality.

1.2 Functions

The 3-Heights® PDF to Image Converter API merges pages from different input files to form one or more files. Color space and image size are defined automatically during the process. The converter supports scaled and un-scaled conversions and a variety of image formats such as PNG, TIFF, JBIG2, or JPEG2000.

1.2.1 Features

PDF to image

- Create single-page and multi-page image files and rasterized PDF documents
- Convert individual pages
- Convert PDF files to CCITT fax files
- Define page dimensions in points or pixels
- Set rotation (Force portrait or landscape or inherit rotation from original document)
- Set resolution (DPI)
- Add dithering (Floyd Steinberg, Halftone Block, Halftone Continuous, Atkinson)
- Set image filters
- Set color depth
- Set color space
- Set TIFF file compression
- Set the quality of lossy image compression
- Set bit filling order for fax files
- Add watermark images

1.2.2 Formats

Input formats

- PDF 1.x (PDF 1.0, ..., PDF 1.7)
- PDF 2.0
- PDF/A-1, PDF/A-2, PDF/A-3

Output formats

- TIFF (Tagged Image File Format)
- JPEG (Joint Photographic Expert Group)
- PNG (Portable Network Graphics)
- GIF (Graphics Interchange Format)
- BMP (Window Bitmap)
- EPS (Encapsulated PostScript)
- JBIG2 (Joint Bi-level Image Experts Group)
- JPEG2000
- Extended JPEG2000
- PBM (Portable Bitmap File Format)

Conformance

Standards:

- ISO 32000-1 (PDF 1.7)
- ISO 32000-2 (PDF 2.0)
- ISO 19005-1 (PDF/A-1)
- ISO 19005-2 (PDF/A-2)
- ISO 19005-3 (PDF/A-3)

1.3 Interfaces

The following interfaces are available:

- C
- Java
- .NET Framework
- .NET Core¹
- COM

1.4 Operating systems

The 3-Heights® PDF to Image Converter 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 Limited supported OS versions. [Operating systems](#)

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 guide](#) 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 to Image Converter 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 to Image Converter 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, unzip 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. Ensure the cache directory exists as described in [Special directories](#).
8. Make sure your platform meets the requirements regarding color spaces and fonts described in [Color profiles](#) and [Fonts](#), respectively.

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 to Image Converter API provides two interfaces:

- Java interface
- Native C interface

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

File description

Name	Description
bin/x64/libPdf2ImgAPI.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/Pdf2ImgAPI.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 libPdf2ImgAPI.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/libPdf2ImgAPI.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 [Inter-face-specific installation steps](#).
6. Ensure the cache directory exists as described in [Special directories](#).
7. Make sure your platform meets the requirements regarding color spaces and fonts described in [Color profiles](#) and [Fonts](#), respectively.

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 libPdf2ImgAPI.dylib libPdf2ImgAPI.jnilib
```

2.3 ZIP archive

The 3-Heights® PDF to Image Converter 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>\Pdf2ImgAPI.dll	Req.	Req.	Req.	Req.
bin*NET.dll	Req.			
bin*NET.xml	Doc.			

Files for development

Name	.NET	Java	COM	C
doc*.pdf	Doc.	Doc.	Doc.	Doc.
doc\Pdf2ImgAPI.idl			Doc.	
doc\javadoc*.*		Doc.		
include\pdf2imgapi_c.h				Req.
include*.*				Opt.
jar\Pdf2ImgAPI.jar		Req.		
lib\<platform>\Pdf2ImgAPI.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>\Pdf2ImgAPI.dll	DLL that contains the main functionality (required), where <platform> is either Win32 or x64 for the 32-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>\Pdf2ImgAPI.lib	On Windows operating systems, the object file library needs to be linked to the C/C++ project.
jar\Pdf2ImgAPI.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.

² Not required for Linux or macOS.

³ These files must reside in the same directory as Pdf2ImgAPI.dll.

Files for deployment

Name	.NET	Java	COM	C
bin\<platform>\Pdf2ImgAPI.dll	Req.	Req.	Req.	Req.
bin*NET.dll	Req.			
jar\Pdf2ImgAPI.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 `Pdf2ImgAPI.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 Pdf2ImgAPI.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%\Pdf2ImgAPI.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 to Image Converter API contains all the libraries needed, both managed and native.

Installation

The package `PdfTools.Pdf2Img 6.27.6` 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.Pdf2Img 6.27.6`.

Development

The package `PdfTools.Pdf2Img 6.27.6` 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 to Image Converter 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 to Image Converter 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 `Pdf2ImgAPI.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>\Pdf2ImgAPI.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 to Image Converter 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 `Pdf2ImgAPI.dll`.

2.5.2 Java interface

The 3-Heights® PDF to Image Converter API requires Java version 7 or higher.

For compilation and execution

When using the Java interface, the Java wrapper `jar\Pdf2ImgAPI.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\Pdf2ImgAPI.jar" ^
sampleApplication.java
```

For execution

Additionally, the library `Pdf2ImgAPI.dll` needs be in one of the system's library directories⁵ or added to the Java

⁴ Otherwise, you get the following message: `LoadLibrary("Pdf2ImgAPI.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`.

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 VM⁶.

```
java -classpath ".;C:\Program Files\PDF Tools AG\Pdf2ImgAPI.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/Pdf2ImgAPI.jar" ...
```

2.5.3 .NET interface

The 3-Heights® PDF to Image Converter 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. Pdf2ImgAPI.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 Pdf2ImgAPI.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 Pdf2ImgAPI.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 Pdf2ImgAPI.dll are copied to subdirectories win-x86 and win-x64 of the output directory, respectively.

2.5.4 C interface

- The header file pdf2imgapi_c.h needs to be included in the C/C++ program.
- On Windows operating systems, the library Pdf2ImgAPI.lib needs to be linked to the project.
- The dynamic link library Pdf2ImgAPI.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.

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"

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 Note about the evaluation license

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

2.8 Special directories

2.8.1 Directory for temporary files

This directory for temporary files is used for data specific to one instance of a program. The data is not shared between different invocations and is deleted after termination of the program.

The directory is determined as follows. The product checks for the existence of environment variables in the following order and uses the first path found:

Windows

1. The path specified by the %TMP% environment variable
2. The path specified by the %TEMP% environment variable
3. The path specified by the %USERPROFILE% environment variable
4. The Windows directory

Linux and macOS

1. The path specified by the \$PDFTMPDIR environment variable
2. The path specified by the \$TMP environment variable
3. The /tmp directory

2.8.2 Cache directory

The cache directory is used for data that is persistent and shared between different invocations of a program. The actual caches are created in subdirectories. The content of this directory can safely be deleted to clean all caches.

This directory should be writable by the application; otherwise, caches cannot be created or updated and performance degrades significantly.

Windows

- If the user has a profile:
%LOCAL_APPDATA%\PDF Tools AG\Caches
- If the user has no profile:
<TempDirectory>\PDF Tools AG\Caches

Linux and macOS

- If the user has a home directory:
~/pdf-tools/Caches

- If the user has no home directory:
 <TempDirectory>/pdf-tools/Caches

where <TempDirectory> refers to the [Directory for temporary files](#).

2.8.3 Font directories

The location of the font directories depends on the operating system. Font directories are traversed recursively in the order as specified below.

If two fonts with the same name are found, the latter one takes precedence, i.e. user fonts always take precedence over system fonts.

Windows

1. %SystemRoot%\Fonts
2. User fonts listed in the registry key \HKEY_CURRENT_USER\Software\Microsoft\Windows NT\CurrentVersion\Fonts. This includes user specific fonts from C:\Users\<user>\AppData\Local\Microsoft\Windows\Fonts and app specific fonts from C:\Program Files\WindowsApps
3. Fonts directory, which must be a direct subdirectory of where Pdf2ImgAPI.dll resides.

macOS

1. /System/Library/Fonts
2. /Library/Fonts

Linux

1. /usr/share/fonts
2. /usr/local/share/fonts
3. ~/.fonts
4. \$PDFFONTDIR or /usr/lib/X11/fonts/Type1

3 License management

The 3-Heights® PDF to Image Converter 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](#).

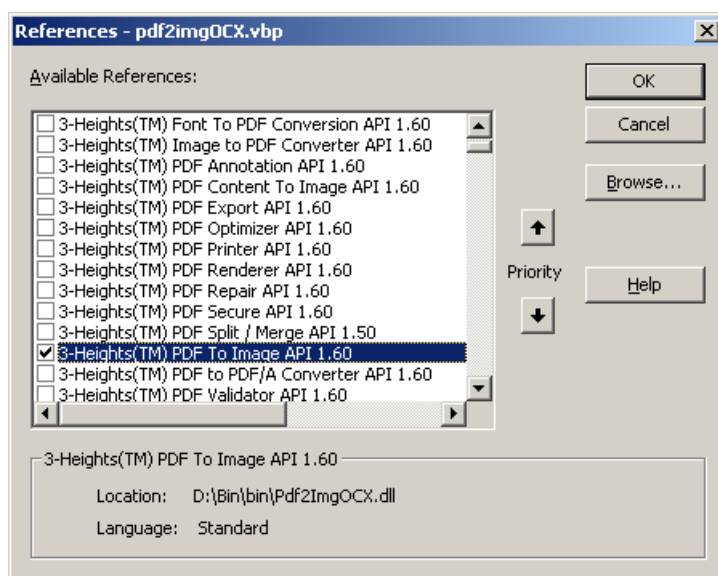
4 Programming interfaces

4.1 Visual Basic 6

After installing the 3-Heights® PDF to Image Converter API and registering the COM interface (see [Installation and deployment](#)), you find a Visual Basic 6 example with file extension .vpb 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 to Image Converter 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.

```
Private Sub Command1_Click()  
    Dim conv As New Pdf2ImgAPI.Pdf2Img  
    conv.ConvertFile "C:\pdf\in.pdf", "C:\image\out.tif", ""  
End Sub
```

The two steps of the above code are very simple: (1) Create a Pdf2Img object, (2) open the PDF file for input, create an image file for output, render all pages of the PDF (if the output file a TIFF which supports multi-page images).

There are two ways to convert pages from PDF files to image pages. The simpler approach is described above. The other approach, which is a bit longer but also more powerful, is to divide this one large step into several single steps. As a consequence, it is possible to open different PDF input files and render random pages to one output multi-page image.

A construct that performs this action look like this:

```
Private Sub Command1_Click()  
    Dim conv As New Pdf2ImgAPI.Pdf2Img  
    conv.CreateImage "C:\image\out.tif"  
    conv.Open "C:\pdf\in1.pdf", ""
```

```

conv.RenderPage 1
conv.Close
conv.Open "C:\pdf\in2.pdf", ""
conv.RenderPage 3
conv.RenderPage 6
conv.Close
conv.CloseImage
End Sub

```

4.2 ASP

The COM name of the class, for example, used in ASP, of the 3-Heights® PDF to Image Converter API is [PDF2IMGAPI.Pdf2Img](#).

Here is a small ASP sample using VBScript:

```

<%@ Language=VBScript %>
<%
    option explicit
    dim conv set
    conv = Server.CreateObject("Pdf2ImgAPI.Pdf2Img")
    if not conv.CreateImage("C:\temp\output.jpg") then
        Response.Write "<p>"
        Response.Write "Could not create output file." & "<br>"
    else
        Response.Write "<p>"
        Response.Write "Output file created successfully." & "<br>"
        if not conv.Open("C:\PDF-Tools\doc\license.pdf") then
            Response.Write "<p>"
            Response.Write "Could not open input file." & "<br>"
        else
            Response.Write "<p>"
            Response.Write "Input file opened successfully." & "<br>"
            if not conv.RenderPage(1) then
                Response.Write "<p>"
                Response.Write "Could not render page 1." & "<br>"
            else
                Response.Write "<p>"
                Response.Write "Page 1 rendered successfully." & "<br>"
            end if
        end if
    end if
    conv.Close
    conv.CloseImage
%>

```

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 to Image Converter 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.Pdf2Img 6.27.6, as described in [NuGet package](#).

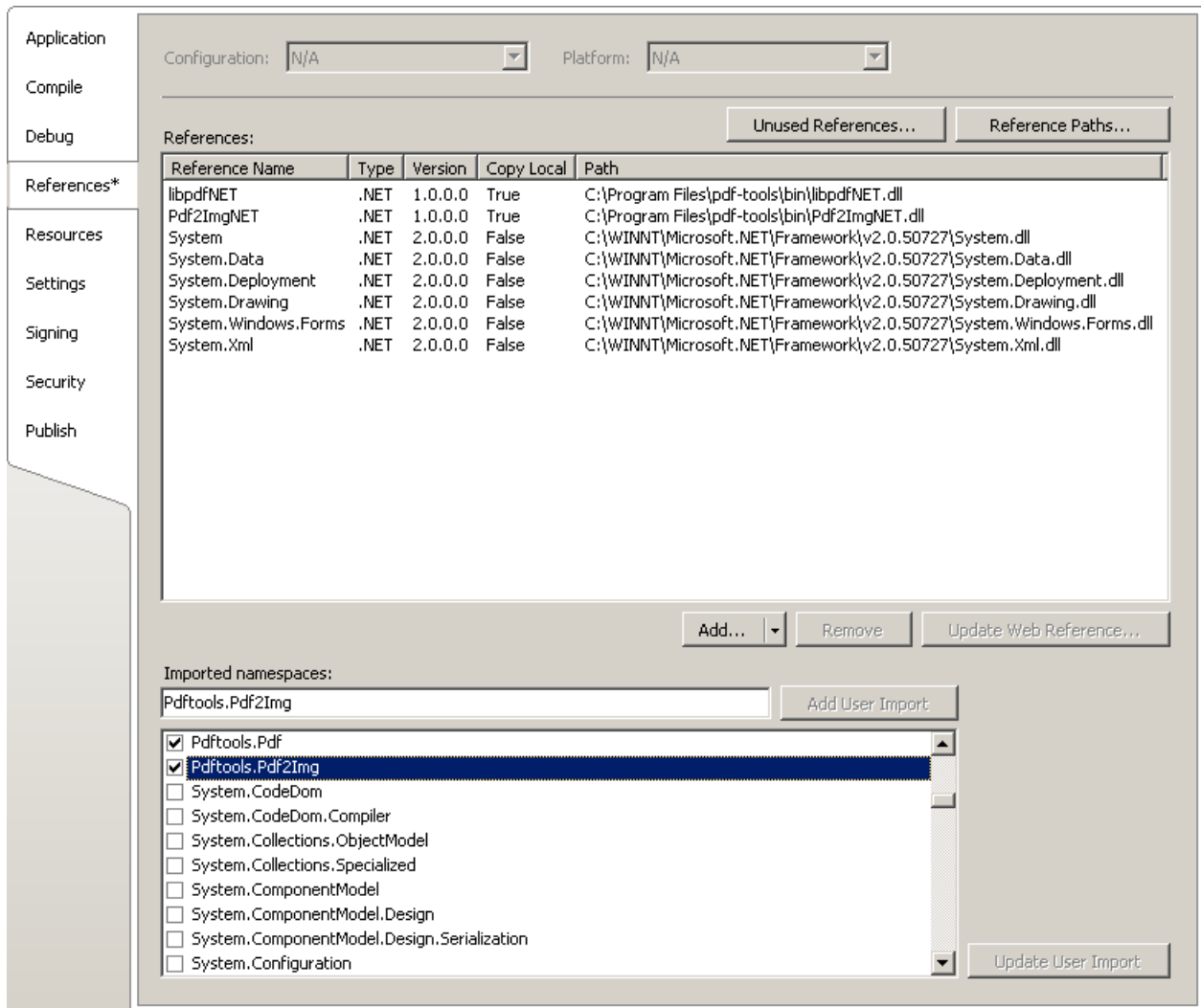
3. Import namespaces (Note: This step is optional, but useful.)
4. Write your code.

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.Pdf2Img](#).

You should now have settings similar as in the screenshot below:



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

Example:

```
Dim conv As New PdfTools.Pdf2Img.Converter
conv.Open(...)
...
```

4.3.2 C#

3. Add the following namespaces:

Example:

```
using Pdftools.Pdf;
using Pdftools.PdfRenderer;
using Pdftools.Pdf2Img;
```

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

Example:

```
using (Converter conv = new Converter())
{
    conv.Open(...);
    ...
}
```

4.3.3 Deployment

This is a guideline on how to distribute a .NET project that uses the 3-Heights® PDF to Image Converter 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 to Image Converter API's .NET assemblies (`*.NET.dll`), as well as the native library (`Pdf2ImgAPI.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 `Pdf2ImgAPI.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 `Pdf2ImgAPI.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 `Pdf2ImgAPI.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 Pdf2ImgAPI.dll could not be found at execution time.

Resolve this by performing one of these actions:

- Use the [NuGet package](#).
- Add Pdf2ImgAPI.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 Pdf2ImgAPI.dll resides to the environment variable %Path%, or
- Manually copy Pdf2ImgAPI.dll to the output directory of your project.

Troubleshooting: BadImageFormatException

The exception means that the native DLL Pdf2ImgAPI.dll has the incorrect "bitness" (i.e. platform 32 vs. 64 bit). There are two versions of Pdf2ImgAPI.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 guide

5.1 Supported codecs

The following table lists the capabilities that the different codecs that are supported by the 3-Heights® PDF to Image Converter API. Other capabilities not listed here are not supported.

Codec capabilities

Codec	Bits per Pixel	Gray	Indexed	Quality	Compression
TIFF	1,2,3,4,8,24 ⁷	Yes	Yes	Yes	Raw, Flate, LZW(default), JPEG, Group3, Group3_2D, Group4
JPEG	8, 24	Yes	No	Yes	JPEG (lossy only)
BMP	1, 2, 4, 8, 24 ⁷	Yes	Yes	No	Raw
GIF	2-8	Yes	Yes	No	LZW
PNG	1-8, 24	Yes	Yes	No	Flate
JBIG2	1	Yes	No	No	JBIG2 (lossless only)
JPEG2000	8, 24	Yes	Yes	Yes	JPEG2000 (lossless: Q = 100) ⁸
PBM	1-8, 24	Yes	No	No	Raw
EPS	1, 2, 4, 8, 24 ⁷	Yes	No	No	Raw

Codec The **Compression/Decompression** type.

Bits per pixel The supported values for bits per pixel. 1 = bitonal, 8 = 256 colors/grayscales, 24 = True Color

Gray This format supports grayscale.

Indexed This format supports indexed colors.

Quality This format supports the setting of a quality parameter.

Compression Supported compression types.

5.1.1 File and compression type

Most image types have a predefined compression algorithm. For TIFF, the type of compression can be selected manually.

⁷ For palette creation: The number of palette entries is equal to $2^{\text{BitsPerPixel}}$ where BitsPerPixel is smaller or equal to 8. This means it is possible to create a 3 bits per pixel TIFF or BMP, but the palette size is equal as for 4 bits. However the 3 bits per pixel image will compress better than the 4 bits per pixel image.

⁸ To create lossless JPEG2000 images, set the quality parameter to 100. For values <100, a lossy compression algorithm is applied.

JPEG and JPEG2000 formats allow the compression rate to be adjusted via a quality parameter. These two formats are of newer date and do not work with older PDF software.

Here are some suggestions of the image type that can be selected for each purpose.

Lossless

- Black/White JBIG2 or TIFF with G4 compression
- Grayscale PNG, JPEG2000 (Q = 100)
- Color PNG, JPEG2000 (Q = 100)

Lossy

- Grayscale JPEG, JPEG2000
- Color JPEG, JPEG2000

For the Internet

- Black/White PNG
- Color (Photos) JPEG, PNG
- Color (Artificial) GIF, PNG

As a general note: Compression algorithms that are lossy require more CPU as lossless algorithms. The file size is usually smaller, but the time to create (compress) or read (decompress) the file is higher.

Images formats that are supported by most Internet browsers are JPEG, GIF and PNG.

5.2 Creating multi and single-page images

5.2.1 Multi-page images

The TIFF format is an image format that supports multi-page images.

To create multi-page TIFF images, just keep rendering pages, and open/close PDF documents without closing the TIFF image.

5.2.2 Single-page images

To create single-page TIFF images, render one page, close the image, and create a new image file.

5.3 Setting pixels equal points

You can create images with pixel (in image) equal to points (in PDF).

The default value of the resolution in the created image is 150 DPI. The PDF format uses a resolution of 72 DPI. To create an image with as many pixels per dimension as the PDF had in points, use an image resolution of 72 DPI as well. Generally, this yields in a grainy image when viewed at 100% zoom, since the monitor uses a resolution of 96 DPI.

5.4 Reducing the file size

There are different ways to reduce the file size of an image. You need to be aware that from a certain point on, a smaller file size results in a poorer visual quality.

The main factors on which the file size of an image depends are:

- Dimensions in pixel (width and height)
- Bits per pixel
- Compression type
- The content of the image (influenced by dithering)

5.4.1 Dimensions

Reducing the dimensions and therefore, the amount of the total pixels reduces also the file size. Obviously, a 1024x768 pixel image has a larger file size than an equivalent 600x480 image.

Example: Set the dimensions in pixels.

```
converter.SetBitmapDimensions(600, 480);  
converter.FitPage = true;
```

Example: Set the dimension in points.

```
converter.SetPageSize(600, 480);  
converter.FitPage = true;
```

If the dimensions are set in points, the dimensions in pixel are computed depending on the resolution.

5.4.2 Resolution

The resolution in dots per inch (DPI) lets you specify how detailed the image is. The default value is 150 DPI, which generates an image that looks sharp when not augmented in zoom. A larger value generates a more detailed image, but also increases the file size, because it requires more pixels. On the other hand, a lower resolution generates a file with a smaller file size, but the image is also of lower visual quality.

Example: Setting the resolution value to 75 DPI instead of 150 DPI reduces the file size to about one quarter.

```
converter.DPI = 75;
```

5.4.3 Bits per pixel

Using 1-bit (black/white) or 8-bit grayscale instead of 24-bit true color reduces the file size. Keep in mind that not all formats support all color depths.

8-bit grayscale images are a third as large in size as 24-bit color images. With 1-bit images that use dithering, the size heavily depends on the content. It can be as small as 1% of the 8-bit image.

Example: Create a grayscale image.

```
converter.BitsPerPixel = 8;
```

Example: Create a bi-level image with Atkinson dithering.

```
converter.BitsPerPixel = 1;  
converter.Dithering = eDitherAtkinson;
```

5.4.4 Format/Compression type

The 3-Heights® PDF to Image Converter API supports various image formats. For most formats, the compression is given. For example, a PNG image is always Flate-compressed, a JPEG image is always JPEG-compressed. However, for TIFF, the compression type is selectable.

Images formats that are supported by most Internet browsers are JPEG, GIF and PNG.

There are two fundamentally different types of compression: Lossless and lossy.

Lossless compression The transformation from the original to the compressed state of the image does not change the content. Thus, the transformation is reversible and the original image can be regained from the compression state.

Lossless compression is normally used for artificial images or scanned text. It is applied to the following types of images: GIF, PNG, BMP, JPEG2000 if quality is set to 100, JBIG2, and TIFF compressed with G3, G4, LZW or Flate.

Lossy compression The compression algorithm alters the content of the image in a way that it compresses better. Thus, a lossy compressed image cannot be reverted back to its original state. It also means multiple applications of lossy compression to the same image alter the image every time and thereby reduce the quality every time. The degree to which the image may be altered to improve the compression rate is controlled by a quality index ranging from 1 to 100 and normally defaulted at 75.

Lossy algorithms usually provide a better compression rate, at the cost of visual quality. Lossy compression is normally used for photographs.

It is applied to the following types of images: JPEG, and JPEG2000 if quality is less than 100.

There are various compression types supported for the TIFF image format. These are:

CCITT Group 3, Group 3-2D CCITT Group 3 is the predecessor to CCITT Group 4. It is a simpler algorithm that normally results in a lower compression ratio.

CCITT Group 4 CCITT Group 4 is the standard compression for bi-level TIFF images (i.e. facsimile).

LZW LZW (Lempel-Ziv-Welch) compression is a lossless compression algorithm for images.

See the copyright laws of your country prior to using this compression algorithm.

JPEG TIFF allows images to be compressed with JPEG, which is a lossy compression algorithm. JPEG provides a high compression ratio for 8 and 24 bit images. It is best suited for TIFFs containing photographs and little or no text.

ZIP (Flate) ZIP is a lossless compression algorithm. It is useful for the compression of large images with no loss in quality.

Flate compression (also used by the ZIP format) and JPEG compression can be used for color or grayscale images. CCITT Group 3, 3-2D and 4, as well as Flate can be used for black and white images.

Example: Apply Flate compression to a TIFF image.

```
converter.Compression = eComprFlate;
```

5.4.5 Image content, dithering

The content of the image itself has a direct impact on how well it compresses. It seems quite obvious that a plain white image compresses much better than a page filling photograph.

Dithering is an algorithm that arranges the pixels of an image in a way that it creates a visual effect of colors that do not exist in the available colors of the image, such as different grays in a 1-bit black and white image. This complex arrangement of pixels, however, does not compress well and increase the file size. Disabling dithering reduces the file size. In the 3-Heights® PDF to Image Converter API, dithering is also implemented for color images.

Example: Disable dithering (e.g. for scanned text).

```
converter.Dithering = eDitherNone;
```

For more information, see [Dithering](#).

5.5 Using the in-memory methods

An image created by the 3-Heights® PDF to Image Converter API can consist of multiple pages. For example, if the image format supports multiple pages, such as the TIFF envelope and [RenderPage](#) is called multiple times, [CreateImageInMemory](#) needs to be called for every image created. [GetImage](#) returns a byte array holding the image. Its length can be retrieved applying the appropriate length-operator of the programming language you are using.

5.5.1 Creating a document in memory

Here is a Visual Basic 6 sample that opens a document from file, creates the image in-memory, and saves it to the variant pdfbytes.

```
Private Sub ConvertInMemory_Click()  
Dim conv As New Pdf2ImgAPI.Pdf2Img  
Dim pdfbytes As Variant  
Dim length As Long  
conv.Open "C:\input.pdf"  
conv.CreateImageInMemory ".tif"  
conv.RenderPage 1  
conv.RenderPage 2  
pdfbytes = conv.GetImage  
length = LenB(pdf)  
conv.CloseImage  
conv.Close  
End Sub
```

5.5.2 Reading a document from memory

The Visual Basic 6 code below opens a document from memory.

In part (1), the document is written into a byte array. This part is just a sample; it could as well be replaced by a process reading the byte array from a database.

In part (2), the document is actually opened from memory.

```
Private Sub OpenFromMemory_Click()  
' (1) Write PDF document to memory  
    Dim conv As New Pdf2ImgAPI.Pdf2Img  
    Dim bChar() As Byte  
    Dim lFileLenght As Long  
    Open "C:\input.pdf" For Binary As #1  
    lFileLenght = LOF(1)  
    ReDim bChar(lFileLenght - 1)  
    Get #1, , bChar  
    Close #1  
' (2) open document from memory  
    If Not conv.OpenMem(bChar, "") Then  
        MsgBox "couldn't open document"  
    End If  
End Sub
```

5.6 Color profiles

A PDF document may contain graphical objects using various different color spaces and the output file of 3-Heights® PDF to Image Converter API may yet use another color space. Therefore often colors have to be converted between different color spaces.

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 to Image Converter API requires appropriate 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 to Image Converter API.

5.6.1 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 Pdf2ImgAPI.dll resides.

Linux and macOS

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

5.6.2 Set other color profiles

Other color profiles may be set using the [SetsRGBProfile](#) and [SetCMYKProfile](#) methods.

5.6.3 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>

5.7 Fonts

PDF documents may contain both embedded and non-embedded fonts. When rendering non-embedded fonts, the best result can be achieved, if the font is available on the system. Therefore, it is important to make sure the [Font directories](#) contain all fonts required.

For more information on how to deal with font issues, please refer to [Font and text issues](#).

Note that on Windows, when a font is installed, it is by default installed only for a particular user. It is important to either install fonts for all users, or make sure the 3-Heights® PDF to Image Converter API is run under that user and the user profile is loaded.

On Linux and macOS, it is recommended to install the Liberation fonts, Google Noto CJK fonts, and the OpenSymbol font. On Debian based systems, the packages are called fonts-liberation2, fonts-noto-cjk, and fonts-opensymbol.

5.7.1 Font cache

A cache of all fonts in all [Font directories](#) is created. If fonts are added or removed from the font directories, the cache is updated automatically.

In order to achieve optimal performance, make sure that the cache directory is writable for the 3-Heights® PDF to Image Converter API. Otherwise, the font cache cannot be updated and the font directories have to be scanned on each program startup.

The font cache is created in the subdirectory <CacheDirectory>/Installed Fonts of the [Cache directory](#).

5.7.2 Microsoft core fonts on Linux or macOS

Many PDF documents use Microsoft core fonts like Arial, Times New Roman, and other fonts commonly used on Windows. Therefore, it is recommended to install these fonts to your default font directories. Many Linux distributions offer an installable package for these “Microsoft TrueType core fonts”. For instance, on Debian based systems, the package is called ttf-mscorefonts-installer.

Alternatively, you can download the fonts from here:

<https://corefonts.sourceforge.net/>

Microsoft has an FAQ on the subject, that covers licensing related questions as well:

<https://docs.microsoft.com/en-us/typography/fonts/font-faq>

5.7.3 Font configuration file fonts.ini

The font configuration file is optional. It can be used to control the mapping of fonts used in the PDF to fonts pre-installed on the system.

The file `fonts.ini` must reside at the following location, which is platform dependent:

Windows: In a directory named `Fonts`, which must be a direct subdirectory of where `Pdf2ImgAPI.dll` resides.

Unix: The `fonts.ini` file is searched in the following locations

1. If the environment variable `PDFFONTDIR` is defined: `$PDFFONTDIR/fonts.ini`
2. `~/pdf-tools/fonts/fonts.ini`
3. `/etc/opt/pdf-tools/fonts/fonts.ini`

It consists of two sections: `[fonts]` and `[replace]`. Both sections are used to map fonts in the PDF to fonts in the installed font collection on the operating system. This comes into play when the font in the PDF document does not have an embedded font program, or the embedded font is not usable.

The mapping only works if the font types of the specified fonts are matching; for example, if the font in the PDF is a symbolic font, such as "Symbol" or "ZapfDingbats", the mapped font must be symbolic too.

The section `[fonts]` is only considered if the font-matcher does not find an appropriate font among the existing installed fonts. It is suggested to only use this section.

The section `[replace]` is stronger and applied before the font-matcher. This means a font will be replaced as defined, even if the correctly installed font is available on the system.

Syntax: The syntax of the mapping file is as follows

```
[fonts]
PDF_font_1=installed_font_1{,font_style}
PDF_font_2=installed_font_2{,font_style}
[replace]
PDF_font_n=installed_font_n{,font_style}
```

PDF_font_* is the name of the font in the PDF.

This name can be found in one of the following ways:

- Use any tool that can list fonts. Such as 3-Heights® PDF Extract or 3-Heights® PDF Optimizer. Ignore possible prefixes of font subsets. A subset prefix consists of 6 characters followed by the plus sign. For example "KHFOKE+MonotypeCorsiva", in this case only use "MonotypeCorsiva" as font name in the mapping file.
- Open the document with Adobe Acrobat, use the "MarkUp Text Tool", mark the text of which you would like to know the font name, right-click it, select "Properties..."

installed_font_* is the font family name of the installed font.

To retrieve this name, find the font in the Windows' font directory and open it by double-clicking. The first line in the property window displays the font family name (this may vary depending on the operating system). The font family name does not include font styles; so an example of a font family name is "Arial", but not "Arial Italic".

font_style is an optional style that is added coma-separated after the font family name.

The style is always one word. Examples of font styles are "Italic", "Bold", "BoldItalic". Omit the font style, if it is "Regular" or "Normal".

Remove blanks from all font names, i.e. in both the `PDF_font_*` and the `installed_font_*`.

Example:

```
[fonts]
Ryumin-Light=MSMincho
GothicBBB-Medium=MSGothic
[replace]
ArialIta=Arial,BoldItalic
```

5.8 Changing the colors—obtaining a darker black

CMYK colors that are used in the PDF must first be converted to RGB. There are basically two ways how to achieve this:

1. A CMYK color profile is applied. The suggested default color profile is the "U.S. Web Coated (SWOP) v2". Using a different color profile yields in an image output with different colors. See [SetCMYKProfile](#), [SetsRGBProfile](#).

On Windows systems, RGB and CMYK color profiles can be found at the following location: %SystemRoot%\system32\spool\drivers\color

2. If the argument given to [SetCMSEngine](#) is a file name, then the Neugebauer algorithm is used for color conversion. The file given in the argument must hold custom coefficients for the color conversion.

Sample Visual Basic 6 code snippet:

```
Dim conv As New Pdf2ImgAPI.Pdf2Img
' Set the color management engine
conv.SetCMSEngine App.Path & "\CmykToRgb.txt"
```

The default Neugebauer coefficients convert CMYK black (0, 0, 0, 1) to an RGB black, which is not a pure black. The following coefficients create a darker black. The changes are applied on line 5. (The default coefficients for this line are approx. 0.2, see [SetCMSEngine](#)). To obtain an even darker black, the values for K need to be lowered even more.

```
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
0.100000, 0.100000, 0.100000 ; 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
```


5.9 Applying isomorphic stretching

If you have a given page size in pixel and would like to convert a PDF page to an image with exactly these given dimensions, but the height-to-width ratio of the PDF is different, you can apply isomorphic stretching. This is achieved by using different resolutions on the X and Y axis. Assuming the Y resolution is defined, the X resolution is calculated as shown in the code sample below:

```
Dim conv As New Pdf2ImgAPI.Pdf2Img
conv.Open ...
conv.CreateImage ...
For Page = 1 To conv.PageCount
    conv.PageNo = Page
    conv.XDPI = conv.YDPI * conv.PageHeight / conv.PageWidth
                * conv.BitmapWidth / conv.BitmapHeight
    conv.RenderPage Page
Next Page
conv.Close
conv.CloseImage
```

5.10 Dithering

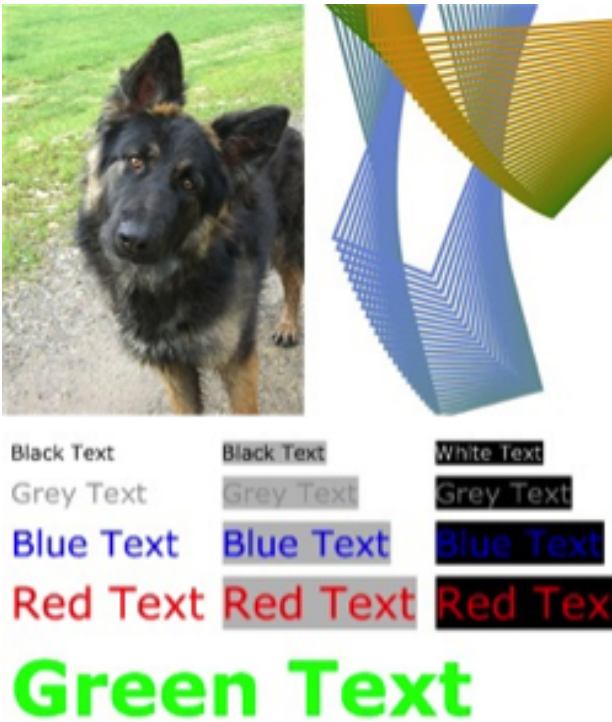
Dithering is a common means used in images to simulate colors that are not available as actual colors. Its use is best observed in image with a low color depth, where colors or shades of gray need to be simulated with other colors (e.g. only black/white pixels).

5.10.1 Remarks

1. All images below have quite a low resolution. As a result, the effects of the different dithering types become more obvious. The higher the resolution and the larger the number of colors, the higher the quality of the image.
2. The rendering filter and current zoom level of the PDF viewing application may have an additional impact on how the images below are displayed.

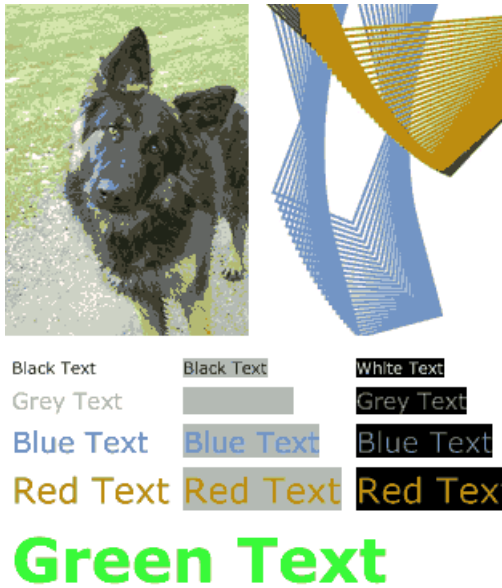
5.10.2 Color images

Color space	RGB (24 bit)
Dithering	None
File size as PNG	129 kB
+	Highest quality
-	Highest file size

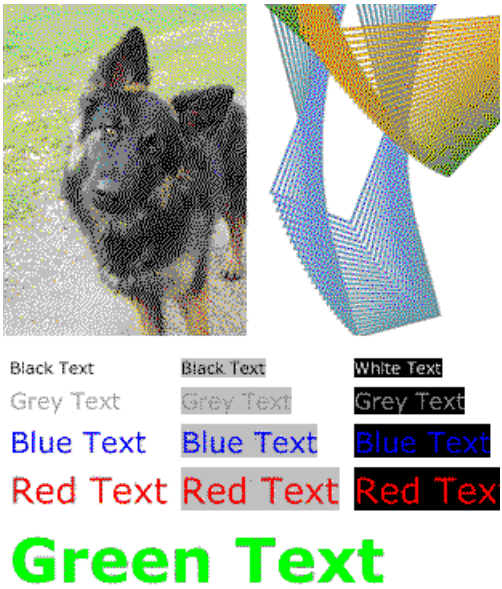


A 24 bit RGB color image can have up to 16.7 millions of different colors. Dithering does not need to be applied since all required colors exist and none need to be simulated.

Color space	16 colors (4 bit)
Dithering	None
File size as PNG	16 kB
+	Small file size
+	Works well for images with a small number of colors (artificial images, text)
-	Does not work well for images with lots of colors (photographic images) - parts of the image can become plain-colored and details get lost



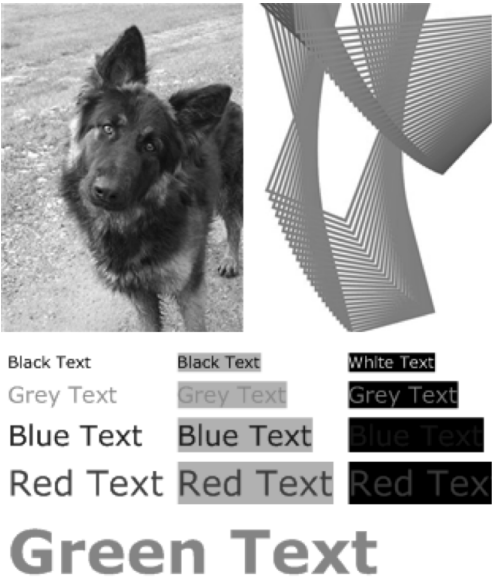
Color space	16 colors (4 bit)
Dithering	Floyd-Steinberg
File size as PNG	18 kB
+	Renders details better
+	Usually better overall quality, especially in photographic images than without dithering
-	Sometimes generates unwanted artifacts (striking pixels)
-	Larger file size then without dithering



5.10.3 Bitonal images

(The 8 bit image just acts as reference.)

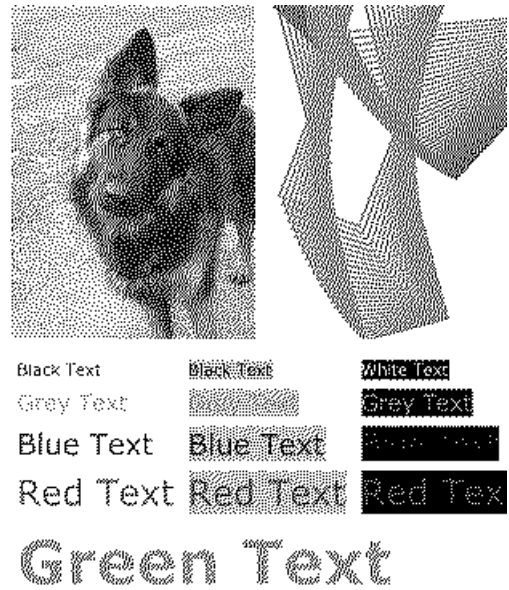
Color space	Grayscale (8 bit)
Dithering	None
File size as PNG	46 kB



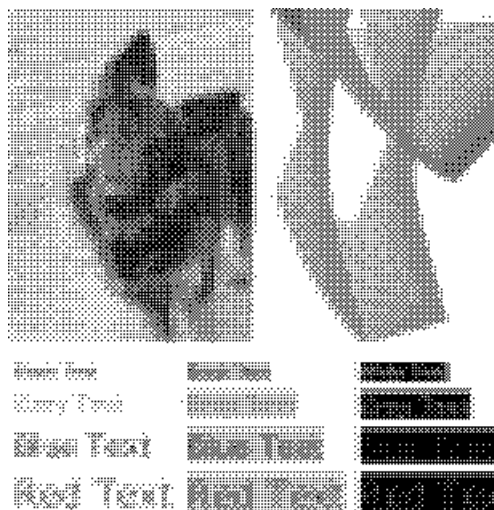
Color space	Grayscale (1 bit)
Dithering	None
File size as PNG	2.6 kB
+	Smallest file size
+	Works well for documents with high contrast (black text on white background)
+	Does not generate artifacts
-	Details get lost, because shades of gray are not approximated, but converted to either black or white (in fact images or part of them can become completely black or white)



Color space	Grayscale (1 bit)
Dithering	Floyd-Steinberg
File size as PNG	9 kB
+	Generally higher quality, specially of photographic images
+	Can approximate any shade of gray
-	Larger file size than without dithering
-	Generates artifacts (e.g. a very bright gray paper is approximated by far-spread single black pixels)
-	Not well suited for text, unless the color of the text must be reflected

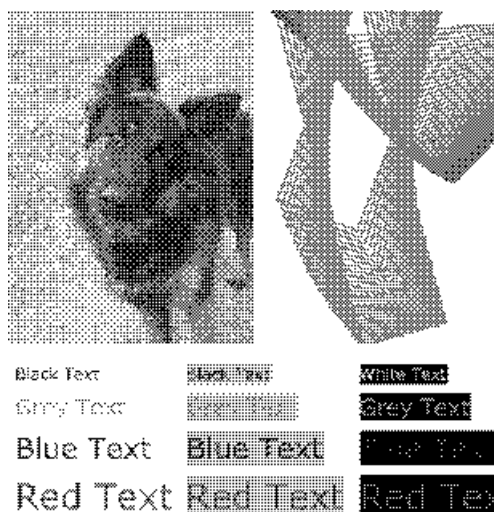


Color space	Grayscale (1 bit)
Dithering	Halftone
File size as PNG	4 kB
+	Small file size
+	Approximates shades of gray
-	Not well suited for text or artificial images



Green Text

Color space	Grayscale (1 bit)
Dithering	Pattern
File size as PNG	5 kB
+	Works acceptable for all types of content (text, photographic images, artificial images)
-	Is not excellent in any type of content



Green Text

5.10.4 Guidelines

As seen in the examples above, different types of dithering behave different for different types of content. Below are some suggestions on the dithering type that works best for a give type of content:

Text, OCR No dithering

Artificial images with few colors and no bright colors No dithering

Artificial images with many colors Test which dithering type yields the best result

Photographic images Floyd-Steinberg

Mixed content Test which dithering type yields the best result

Mixed content, high-resolution For resolutions above 300 DPI, Floyd-Steinberg almost always yields the best result (exception: for pure black text on white background, use no dithering)

Keep in mind that dithering should only be applied for images with a low color depth, such as black and white (1 bit). Dithering for images with a color depth of 8 bit or higher (256 colors or grayscale) has little to no visual impact.

5.11 Error handling

Most methods of the 3-Heights® PDF to Image Converter API can either succeed or fail depending on user input, the state of the PDF to Image Converter 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 (!conv.Open(file, password))
    {
        if (conv.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}", conv.ErrorCode, conv.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.

6.1 Pdf2Img Interface

This interface is included in the Pdf2ImgAPI.dll.

This interface takes a PDF document as input and creates a raster image (e.g. a TIFF) as output.

6.1.1 BilevelThreshold

Property (get, set): Long BilevelThreshold
Default: 181

Get or set the threshold for converting from gray to bitonal when dithering is eDitherNone. Value must be in the range of 0 to 255.

6.1.2 BitmapHeight

Property (get): Long BitmapHeight

Return the height of the bitmap in pixel.

6.1.3 BitmapWidth

Property (get): Long BitmapWidth

Return the width of the bitmap in pixel.

6.1.4 BitsPerPixel

Property (get, set): Integer BitsPerPixel
Default: 24

Get or set the color depth. Bitonal: 1, grayscale: 8, RGB true color: 24, CMYK: 32.

When using 1 bit per pixel, it is suggested to disable anti-aliasing (enable [eOptionNoAntialiasing](#)) and set a suitable dithering algorithm (property [Dithering](#)).

6.1.5 Center

Property (get, set): Boolean [Center](#)
Default: [False](#)

Set or get the center mode. When set to [True](#), the document is horizontally and vertically centered on the page. When set to [False](#), the document is printed to the upper left corner of the page.

6.1.6 ClearRenderingProperties

Method: [ClearRenderingProperties\(\)](#)

Reset all rendering properties to their default value. Rendering properties can be set using [SetRenderingProperty](#).

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 CloseImage

Method: Boolean [CloseImage\(\)](#)

Close an open image document. If the document is already closed, the method does nothing.

Returns:

[True](#) The image file could successfully be closed.

[False](#) Otherwise.

6.1.9 ColorSpace

Property (get, set): `TPDFColorSpace ColorSpace`
Default: `eColorRGB`

Get or set color space of the output image. See enumeration [TPDFColorSpace](#).

For black-white bitonal images, a gray color space must be selected.

6.1.10 Compression

Property (get, set): `TPDFCompression Compression`
Default: `eComprLZW`

Get or set the compression type of TIFF images. For any other image format, the compression is automatically defined by the file extension (the file name).

The supported values for [TPDFCompression](#) are listed in the corresponding enumeration.

6.1.11 ConvertFile

Method: `Boolean ConvertFile(String PDFFileName, String ImageFileName, String Password)`

Convert a complete PDF file to a (multi-page) TIFF image file.

Parameters:

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

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

Password [`String`] 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 was converted successfully.

False The PDF file does not exist, it is corrupt, the password is invalid, or the image file is locked.

6.1.12 CreateImage

Method: Boolean `CreateImage(String FileName)`

Create a new image file.

Parameter:

FileName [`String`] The file name and optionally, the file path, drive or server string according to the operating systems file name specification rules. The file name defines the image format. Supported extensions are:

- `.bmp` (Windows Bitmap Format)
- `.gif` (Graphics Interchange Format)
- `.jb2` (JBIG2, Bi-level Images)
- `.jpg`, `.jpeg` (Joint Photographic Experts Group)
- `.jp2` (JPEG2000)
- `.jpf`, `.jpx` (JPEG2000, Part 2 - Coding Extensions)
- `.png` (Portable Network Graphics)
- `.tif`, `.tiff` (Tagged Image File Format)

Returns:

True The file could successfully be created.

False Otherwise.

6.1.13 CreateImageInMemory

Method: Boolean `CreateImageInMemory(String Extension)`

Save an image in memory as a byte array. See also the [GetImage](#) method.

Parameter:

Extension [`String`] The name of the extension. For a list of supported extensions, see the [CreateImage](#) method. The leading "." needs to be included.

Returns:

True The image could successfully be created.

False Otherwise.

6.1.14 Dithering

Property (get, set):	TPDFDithering Dithering
Default:	eDitherFloydSteinberg

Get or set the dithering algorithm. Dithering refers to the procedure of simulating colors or grayscales. This is mainly useful for low color depth (e.g. black and white or indexed) images.

The supported values for [TPDFDithering](#) are listed in the corresponding enumeration. For more information see [Dithering](#).

6.1.15 DPI

Property (get, set):	Single DPI
Default:	150

Get or set the resolution of the image in DPI (dots per inch).

Set Both the resolutions for the X- and Y-axis are set to the same value.

Get Return the square root of the product of X and Y.

Setting DPI is redundant to setting the specialized properties [XDPI, YDPI](#) and [XDPI, YDPI](#).

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 to Image Converter API has returned a value, which signals 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 to Image Converter 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 to Image Converter 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 FaxHSetting, FaxSSetting

Method: Boolean FaxHSetting()

Method: Boolean FaxSSetting()

These two methods set the TIFF Class F settings, which is equal to:

```
conv.RotateMode = eRotatePortrait
conv.SetBitmapDimensions(1728, 0)
conv.XDPI = 204
conv.YDPI = 196 ' for Fax H
conv.YDPI = 98  ' for Fax S
conv.Compression = eComprGroup3
```

6.1.19 FillOrder

Property (get, set): Integer FillOrder

Default: 1 (MSB)

Get or set the bit fill order. 1 is MSB (most significant bit) first, 2 is LSB (least significant bit) first.

6.1.20 FilterRatio

Property (get, set): Integer FilterRatio

Default: 1

This property is used to enable and parameterize super-sampling, a technique to initially render the image at a higher resolution and then sample it down to the target resolution. As a result of that process, the final image appears smoother, i.e. anti-aliased.

Applying super-sampling improves the image quality when rendering at low target resolutions (72 DPI or less); the higher the target resolution, the less the visual impact.

This property requires memory and CPU time quadratically to the ratio, therefore only small values, such as 2 or 3 should be used.

If a too high value (in combination with the original image size) is set, it is ignored.

6.1.21 FitPage

Property (get, set): Boolean FitPage

Default: True

Get or set the fit page mode. If set to **True**, the page is scaled to fit the image (in either width or height). If set to **False**, the page is rendered with its true size.

6.1.22 GetImage

Method: Variant `GetImage()`

Return the byte array, which was previously saved using [CreateImageInMemory](#).

6.1.23 GetOcg

Method: `Ocg GetOcg(Integer Count)`

Return an interface to an optional content group item.

Parameter:

Count [`Integer`] The number of the optional content group. Optional content groups are numbered from `0` to `OcgCount-1`.

Returns:

An interface to an optional content group item.

See also [Ocg](#) interface.

6.1.24 HasColor

Method: Boolean `HasColor(Long IPageNo)`

Return `True` if the selected page contains colors. `False` otherwise.

6.1.25 ImageQuality

Property (get, set): Single `ImageQuality`
Default: `80`

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.

6.1.26 LicenseIsValid

Property (get): Boolean `LicenseIsValid`
Static

Check if the license is valid.

6.1.27 OcgCount

Property (get): Long `OcgCount`

Get the number of optional content groups (also known as “layers”) of the document.

See also [GetOcg](#).

6.1.28 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.

6.1.29 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.

6.1.30 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.

6.1.31 Options

[Deprecated] Property (get, set): TPDFRendererOption `Options`

Use [Options2](#).

6.1.32 Options2

Property (get, set): TPDFRendererOption2 `Options2`

Set or get a specific rendering option.

Use bitwise "OR" to add an option.

Use bitwise "AND NOT" to remove an option.

For more information on the options available in the 3-Heights® PDF to Image Converter API and how to use the this property, see [TPDFRendererOption2](#).

6.1.33 PageBoxType

Property (get): TPDFPageBox PageBoxType
Default: ePageBoxCrop

Get or set the box type used for rendering the page.

The supported values for [TPDFPageBox](#) are listed in the corresponding enumeration.

6.1.34 PageCount

Property (get): Long PageCount

Get the number of pages of an open document. If the document is closed or if the document is a collection (also known as PDF portfolio), then this property is 0.

6.1.35 PageHeight

Property (get): Float PageHeight

Return the height of the page in points.

6.1.36 PageWidth

Property (get): Float PageWidth

Return the width of the page in points.

6.1.37 PageNo

Property (get, set): Long PageNo

Get or set the current page number. Allowed values are in the range from 1 to [PageCount](#). This property is not needed for rendering, but the extraction of page and output bitmap dimensions only.

6.1.38 PageXOffs, PageYOffs

Property (get): Float PageXOffs
Default: CropBox

Property (get): Float PageYOffs
Default: CropBox

Method: SetPageOffs(Float x, Float y)

Get the offset of the page in points. The default offset is the CropBox's offset. Set the offset before [RenderPage](#) and get it afterwards.

6.1.39 PreserveAspectRatio

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

If **True**, a uniform up- or down-scaling is applied, i.e. the output image has the same ratio of width to height as the input file and its size fits into the defined dimensions, given by [SetBitmapDimensions](#).

6.1.40 ProductVersion

Property (get): String ProductVersion

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

6.1.41 Quality

[Deprecated] Property (get, set): Integer Quality

Use [ImageQuality](#) instead.

6.1.42 RenderingMode

[Deprecated] Property (get, set): RenderingMode RenderingMode

In version 2.0 and higher there is only one rendering mode.

6.1.43 RenderPage

Method: Boolean RenderPage(Long PageNumber)

Render (convert) the selected page in the PDF document to the raster image.

Parameter:

PageNumber [Long] The page number in the PDF document, non-zero based.

Returns:

True The page was rendered successfully.

False The page could not be rendered. Possible reasons are: out of range, no PDF opened, no image created.

6.1.44 RotateMode

Property (get, set): `TPDFRotateMode RotateMode`
Default: `eRotateAttribute`

Get or set the rotation mode of the page. There are four valid values, which are described in the enumeration [TPDFRotateMode](#).

6.1.45 SetBitmapDimensions

Method: `Void SetBitmapDimensions(Long X, Long Y)`

Set the dimensions of the image in pixels.

Parameters:

X [Long] The X dimension of the image in pixels.

Y [Long] The Y dimension of the image in pixels.

6.1.46 SetCMSEngine

Method: `Boolean SetCMSEngine(String CMSEngine)`

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.

6.1.47 SetCMYKProfile

Method: Boolean `SetCMYKProfile(String FileName)`

Set the path to the CMYK profile. If no path is set, a default profile is used (see [Color profiles](#)).

Parameter:

FileName [String] The path and file name of the ICC CMYK color profile.

Returns:

True The color profile could successfully be selected.

False Otherwise.

6.1.48 SetLicenseKey

Method: Boolean `SetLicenseKey(String LicenseKey)`

Sets the license key.

6.1.49 SetPageSize

Method: `Void SetPageSize(Single X, Single Y)`

Set the dimensions of the image in points.

Parameters:

X [`Single`] The X dimension of the image in points.

Y [`Single`] The Y dimension of the image in points.

6.1.50 SetRenderingProperty

Method: `Boolean SetRenderingProperty(TPDFRenderingProperty Property, String Value)`

Set a rendering property. A list of rendering properties and their allowed values can be found in [TPDFRenderingProperty](#). All rendering properties can be reset to their initial value using [ClearRenderingProperties](#).

Parameters:

Property [`TPDFRenderingProperty`] The property to set.

Value [`String`] The new value.

Returns:

True The property could successfully be set.

False Otherwise.

6.1.51 SetsRGBProfile

Method: `Boolean SetsRGBProfile(String FileName)`

Set the path to the RGB profile. If no path is set, a default profile is used (see [Color profiles](#)).

Parameter:

FileName [`String`] The path and file name of the ICC RGB color profile.

Returns:

True The color profile could successfully be selected.

False Otherwise.

6.1.52 UnsupportedFeatures

Property (get, set): long [UnsupportedFeatures](#)

Get the unsupported features used on the last page rendered (see enumeration [TPDFUnsupportedFeature](#)).

6.1.53 XDPI, YDPI

Property (get, set): Single [XDPI](#)

Default: 150

Property (get, set): Single [YDPI](#)

Default: 150

Get or set the resolution in the X and Y-axis of the image in dots per inch.

6.2 Ocg Interface

The optional content group (OCG) interface allows to list optional content groups (also known as “Layers”) and their properties.

Optional content groups (OCGs) in PDF differ substantially from the simple layer paradigm found e.g. in graphics editing programs. Graphics objects in PDF do not belong to an OCG. Instead, their visibility is calculated by a Boolean function dependent on the state of any number of OCGs. For example, a path could be visible only if OCG “A” is ON and OCG “B” is OFF.

The functionality of OCG are described in depth in ISO 32000-1, chapter 8.11.4 or in the PDF Reference, chapter 4.10. OCG is supported in PDF 1.5 or later. In the 3-Heights® PDF to Image Converter API, the OCG interface can be used to list “layers” and set them to visible or not. To get the OCG object, use the [OcgCount](#) and [GetOcg](#) methods from the [Pdf2Img](#) interface.

6.2.1 Label

Property (get): Boolean [Label](#)

This is a flag that indicates whether this is an OCG or a label. Labels are used to label groups of OCGs in the hierarchy. Setting their visibility has no effect.

6.2.2 Level

Property (get): Long Level

In user interfaces, OCGs can be shown in a tree. The Level property indicates the hierarchy level of the OCG in that tree.

OCG with Level 0 is a top level OCG. Level -1 means that the OCG is not part of the hierarchy, it should not be presented to the user. Parent elements in the OCG hierarchy can be labels or OCGs. If the level of a label b is higher than its predecessor a, b is the parent element of the following objects of the same level as b. If the level of an OCG b is higher than its predecessor OCG a, a is the parent of the following objects of the same level as b. Note that the hierarchy reflects actual nesting of OCGs in the content. Setting the visibility of an OCG to True only has an effect if the visibilities of all its parents are set to True.

6.2.3 Name

Property (get): String Name

Return the name of the OCG.

6.2.4 Visible

Property (get, set): Boolean Visible

Get or set if the OCG is visible. This property controls the extraction of content objects. The default value is the one configured in the PDF document.

Although invisible paths generate no marks on the page, they still have an effect on the graphics state. For example, their effect on the current drawing position and the clipping region does not change. Therefore, all paths are “active” and extracted regardless of their visibility. Invisible paths just use the end path operator “n”, instead of a filling or stroking operator.

Example 1:

ID, OCGs, Level:	Hierarchy
0, OCG A, 0	- OCG A
1, OCG B, 0	- OCG B
2, OCG B1, 1	-- OCG B1
3, OCG B2, 1	-- OCG B2
4, OCG C, -1	hidden: OCG C

Example 2:

ID, OCGs/Labels, Level	Hierarchy
0, OCG A, 0	- OCG A
1, Label B, 1	- Label B
2, OCG B1, 1	-- OCG B1
3, OCG B2, 1	-- OCG B2
4, Label C, 1	- Label C
5, OCG C1, 1	-- OCG C1
6, OCG D, 0	- OCG D

6.3 Enumerations

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

6.3.1 TPDFColorSpace Enumeration

TPDFColorSpace table

Value		Number of Channels
eColorGray	Gray	1: gray
eColorGrayA	Gray with alpha channel	2: gray and alpha
eColorRGB	Color RGB	3: red, green, and blue
eColorRGBA	Color RGB with alpha channel	4: red, green, blue, and alpha
eColorCMYK	Color CMYK	4: cyan, magenta, yellow, and key (black)
eColorYCbCr	Color YCbCr	3: luminance (Y) and chroma (Cb, Cr)
eColorYCbCrK	Color YCbCrK	4: Luminance (Y), Chroma (Cb, Cr), and black
eColorPalette	Color space using a palette	1: palette indices (into an RGB color table).
eColorLAB	Color CIE L*a*b*	3: L, A, and B
eColorOther	Other	
eColorCMYK_Konly	Grayscale CMYK	Only the K channel is used

6.3.2 TPDFCompression Enumeration

TPDFCompression table

TPDFCompression	Description
eComprRaw	No compression
eComprJPEG	Joint Photographic Expert Group
eComprFlate	Flate compression
eComprLZW	Lempel-Ziv-Welch
eComprGroup3	CCITT Fax Group 3
eComprGroup3_2D	CCITT Fax Group 3 2D
eComprGroup4	CCITT Fax Group 4
eComprTIFFJPEG	JPEG (6). This is an older version of JPEG. Certain (older) image software may support this compression, but not the newer version of JPEG (e.g. Photoshop 8).

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

6.3.3 TPDFDithering Enumeration

TPDFDithering table

TPDFDithering	Description
eDitherNone	No dithering
eDitherFloydSteinberg	Floyd-Steinberg (Default)
eDitherHalftone	Half-toning
eDitherPattern	Pattern Dithering
eDitherG30ptimized	Dithering optimized to compress well with Group 3
eDitherG40ptimized	Dithering optimized to compress well with Group 4
eDitherAtkinson	Atkinson dithering is very fast and produces images that can be compressed really well with a reasonably good image quality.

6.3.4 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 to Image Converter 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_E_PASSWORD	The authentication failed due to a wrong password.
PDF_E_UNKSECHANDLER	The file uses a proprietary security handler, e.g. for a proprietary digital rights management (DRM) system.
PDF_E_COLLECTION	The input file is a PDF collection without an initial document
PDF_E_XFANEEDSRENDERING	The file contains unrendered XFA form fields, i.e. the file is an XFA and not a PDF file. The XFA (XML Forms Architecture) specification is referenced as an external document to ISO 32'000-1 (PDF 1.7) and has not yet been standardized by ISO. Technically spoken, an XFA form is included as a resource in a shell PDF. The PDF's page content is generated dynamically from the XFA data, which is a complex, non-standardized process. For this reason, XFA is forbidden by the ISO Standards ISO 19'005-2 (PDF/A-2) and ISO 32'000-2 (PDF 2.0) and newer.

6.3.5 TPDFPageBox Enumeration

TPDFPageBox table

TPDFPageBox	Description
ePageBoxArt	ArtBox
ePageBoxBleed	BleedBox
ePageBoxCrop	CropBox
ePageBoxMedia	MediaBox
ePageBoxTrim	TrimBox

6.3.6 TPDFRendererOption2 Enumeration

Renderer options are set using the property [Options2](#). To combine multiple options, use a bitwise OR operator. To disable an option, use the bitwise AND NOT operators.

Example: Visual Basic

Enable an option, and leave all other options untouched:

```
' Enable printing mode
.Options2 = .Options2 OR eOptionPrintingMode
```

Example: C/C++

```
int iOptions = Pdf2ImgGetOptions2(pDocument);
// Enable printing mode
Pdf2ImgSetOptions2(pDocument, iOptions | eOptionPrintingMode);
```

The following list includes renderer options that are relevant for the 3-Heights® PDF to Image Converter API.

TPDFRendererOption2 table

TPDFRendererOption2	
eOptionNoAntialiasing	Anti-aliasing for text and path objects and filtering of image objects can be turned off with this option.
eOptionNoInterpolation	Don't use interpolation filtering for images.
eOptionNoLowPassFilter	Disable image low pass filtering, which is used to downscale images Images are scaled using the nearest-neighbor algorithm, which improves performance at the cost of rendering quality.
eOptionNoHinting	Don't use hinting for glyph rendering.
eOptionPrintingMode	Draw the document as it was intended for printing. Otherwise, the document is drawn as it is shown in an interactive viewer. For example, this has an effect on which annotations are visible.
eOptionNoBPC	If this option flag is set, then the black point compensation feature is disabled when converting colors e.g. from CMYK to RGB.
eOptionFitPaths	Fit rectangle clipping paths to pixel grid.
eOptionUseBoxFilter	Use a box filter instead of a Gauss filter to downsample images.
eOptionNoAnnotation	Don't draw annotations.
eOptionDrawPopupAnnots	Draw open pop-up annotations.

6.3.7 TPDFRenderingProperty Enumeration

Rendering properties are set using the method [SetRenderingProperty](#).

TPDFRenderingProperty table

TPDFRenderingProperty	
eRenderingPropertyNone	Placeholder value for future property.

6.3.8 TPDFRotateMode Enumeration

TPDFViewerOption table

TPDFRotateMode	
eRotateAttribute	Set the rotation to the viewing rotation attribute of the PDF page, i.e. rendering the page with the same rotation as it is displayed in a PDF viewer.
eRotatePortrait	Rotate page to portrait.
eRotateLandscape	Rotate page to landscape.
eRotateNone	Process the page as it is saved in the PDF file.

7 Tips, tricks, and troubleshooting

7.1 Font and text issues

1. For issues with text using non-embedded fonts:
 1. Ensure the required fonts are available on the system (see [Fonts](#)).
 2. See [Handle non-embedded fonts](#).

7.1.1 Handle non-embedded fonts

Font replacement strategy

This section describes how the rendering engine handles fonts. It is rather technical and it is not required to be understood to use the software.

The following steps are performed sequentially when searching for a font. If a font is found, the search is stopped; otherwise, the next step is performed.

1. If the font is not embedded:
 - a. If the font name appears in the `[replace]` section in the configuration file `fonts.ini`, the name is replaced and looked up in the installed font collection.
 - b. If it is a standard font⁹ it is replaced by the equivalent TrueType font name and it is looked up in the installed font collection.
 - c. If the font name appears in the `[fonts]` section in the configuration file `fonts.ini`, the name is replaced and looked up in the installed font collection.
 - d. If the font has "Italic" or "Bold" in its name, the font without these styles is looked up in the installed font collection.
2. If a font name is looked up in the installed font collection, then the name comparison is performed as follows:
 - a. PostScript name.
 - b. TrueType name without blanks (a missing style is interpreted as "Regular" or "Normal").
 - c. TrueType name without modifications.
3. If a font from the installed font collection matches the metrics of the font, the installed font is used.
4. If the font is a CID font using a specific character collection, e.g. "Japan1", an installed font that contains the required code pages is used.
5. If the font is a non-symbolic simple font, a font program with the font metrics required is created dynamically.

⁹ e.g. Times-Roman, Helvetica, Courier

8 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.

8.1 Changes in versions 6.19–6.27

- **Update** license agreement to version 2.9

8.2 Changes in versions 6.13–6.18

No functional changes.

8.3 Changes in versions 6.1–6.12

- **Improved** search algorithm for installed fonts: User fonts under Windows are now also taken into account.
- **New** enum value [eOptionDrawPopupAnnots](#) for [TPDFRendererOption2](#).
- **New** functionality to select the box type which is used for rendering the page.
- [\[Java\]](#) **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.

8.4 Changes in version 5

- **New** additional supported operating system: Windows Server 2019.
- [\[PHP\]](#) **New** extension PHP 7.3 (non thread safe) for Linux.

8.5 Changes in version 4.12

- **Improved** the rendering quality of small text using Type3 fonts (bitmap fonts).
- **Improved** the quality of low-pass filtered images, if the default Gauss algorithm is used. Note that for creating bitonal output images, the option [eOptionUseBoxFilter](#) is recommended.
- **Improved** the rendering speed for documents that have resources with circular references.
- **New** HTTP proxy setting in the GUI license manager.

8.6 Changes in version 4.11

- **New** support for reading PDF 2.0 documents.
- **Improved** search in installed font collection to also find fonts by other names than TrueType or PostScript names.

8.7 Changes in version 4.10

- **Improved** rendering engine R2 performance when using Type 3 fonts.
- **Improved** robustness against corrupt input PDF documents.
- **Improved** annotation appearance generation for polyline, squiggly, and stamp annotations.
- **Removed** the font `ZapfDingbats.ttf` from the product kit as it is not required anymore.
- [C] **Clarified** Error handling of `TPdfStreamDescriptor` functions.
- [PHP] **New** Interface for Windows and Linux. Supported versions are PHP 5.6 & 7.0 (Non Thread Safe). The Pdf2ImgAPI 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.

8.8 Changes in version 4.9

- **Improved** support for and robustness against corrupt input PDF documents.
- **Improved** repair of embedded font programs that are corrupt.
- **New** support for OpenType font collections in installed font collection.
- **Improved** metadata generation for standard PDF properties.
- [C] **Changed** return value `pfGetLength` of `TPDFStreamDescriptor` to `pos_t`¹⁰.
- [.NET, C, Java] **New** method `OpenStream()`.
- [.NET, C, COM, Java] **New** property `ErrorMessage` to obtain a message about the occurred error in the most recent API call.

8.9 Changes in version 4.8

- **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 `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

9 Licensing, copyright, and contact

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

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

Licensing and copyright The 3-Heights® PDF to Image Converter 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