

User Manual

OPTATIS

Optatis | Nullam enim vel nisi ipsum | Switzerland

15. September 2000

Urhetspakkut
Estut Magnatur
libencrepenovendis 23
in innoilunestaturusint.
GPEP

rovidendis into illut

Estut Magnatur;

Et magnatur? Quis qui disqui heritem li ut cubica oceanum quae nam carum fugiae audi-
re sunt acilique desseque necusam qui blatusdae quam dolo vit arum quae voluptatur?
Et maximum ut maximet. sequam, tecae lanemol orporibus dolapratem ducipunt.
hene volut apocpa apoclis quam exora cupictus rem et reproc proem videst que-

offic: te recum acitis simponusam stentim incedem fugiat.

Uiam ritum, qui pend ut aut maionequi officis ut lauta am ius rem doltionis

te esti autali bleecat.

Uolor sro: laud quae vellandit: et verum quo ea quia qui asperatur, excoliqui-

met octa quoms eni regularne velam, veliqui scemti turli ut ex conquetet

ur at ce eture viliti fustate sem ut aplique conpra conepit qui veram

puccia ut aut endio dolendunt, ornis onest, illo a som am fugitio

im ne anibus auditas nobit optoquem extantemta igneo modica

ita maxim quam volent: onsequi dolaprobunt rem utata quo cum

te pono vero omni illa sus pliquo incti quo quidcum, ipsanita

te impentibus evolisio sharem at —the catet venduciat.

aptur, vidempes —m dlopipi of

ur, intest



3-Heights™ PDF to Image Converter API

Version 4.12.26.6



Contents

1	Introduction	6
1.1	Description	6
1.2	Functions	6
1.2.1	Features	6
1.2.2	Formats	7
1.3	Interfaces	8
1.4	Operating Systems	8
1.5	How to Best Read this Manual	8
2	Installation and Deployment	9
2.1	Windows	9
2.2	Interfaces	10
2.2.1	Development	11
2.2.2	Deployment	12
2.3	Interface Specific Installation Steps	13
2.3.1	COM Interface	13
2.3.2	Java Interface	13
2.3.3	.NET Interface	14
2.3.4	C Interface	14
2.4	Uninstall, Install a New Version	14
2.5	Note about the Evaluation License	14
2.6	Special Directories	15
2.6.1	Directory for temporary files	15
2.6.2	Cache Directory	15
2.6.3	Font Directories	15
3	License Management	16
3.1	License Installation and Management	16
3.1.1	Graphical License Manager Tool	16
	List all installed license keys	16
	Add and delete license keys	16
	Display the properties of a license	16
3.1.2	Command Line License Manager Tool	17
	List all installed license keys	17
	Add and delete license keys	17
	Display the properties of a license	17
3.2	License Selection and Precedence	18
3.2.1	Selection	18
3.2.2	Precedence	18
3.3	Key Update	19
3.4	License activation	19
3.4.1	Activation	19
3.4.2	Reactivation	20
3.4.3	Deactivation	20
3.5	Proxy Setting	20
3.6	Offline Usage	21
3.6.1	First Step: Create a Request File	21
3.6.2	Second Step: Use Form on Website	22
3.6.3	Third Step: Apply the Response File	22

3.7	License Key Versions	22
3.8	License Key Storage	22
3.8.1	Windows	22
3.9	Troubleshooting	23
3.9.1	License key cannot be installed	23
3.9.2	License is not visible in license manager	23
3.9.3	License is not found at runtime	23
3.9.4	Eval watermark is displayed where it should not	23
3.9.5	Activation is not recognized	24
3.9.6	Activation is invalidated too often	24
3.9.7	Connection to the licensing service fails	25
3.9.8	Offline usage fails due to a request/response mismatch	25
4	Programming Interfaces	26
4.1	Visual Basic 6	26
4.2	ASP	27
4.3	.NET	27
4.3.1	Visual Basic	28
4.3.2	C#	29
4.3.3	Deployment	30
4.3.4	Troubleshooting: TypeInitializationException	30
5	User's Guide	32
5.1	Supported Codecs	32
5.1.1	File and Compression Type	32
5.2	How to Create Multi and Single-Page Images	33
5.2.1	Multi-Page Images	33
5.2.2	Single-Page Images	33
5.3	How to Set Pixels Equal Points	33
5.4	How to Reduce the File Size	33
5.4.1	Dimensions	34
5.4.2	Resolution	34
5.4.3	Bits per Pixel	34
5.4.4	Format/Compression Type	35
5.4.5	Image Content, Dithering	36
5.5	How to Use the In-Memory Methods	36
5.5.1	Creating a Document in Memory	36
5.5.2	Reading a Document from Memory	36
5.6	Color Profiles	37
5.6.1	Default Color Profiles	37
5.6.2	Set other Color Profiles	37
5.6.3	Get Other Color Profiles	38
5.7	Fonts	38
5.7.1	Font Cache	38
5.7.2	Font Configuration File fonts.ini	38
5.8	How to Change the Colors—Obtain a Darker Black	39
5.9	How to Apply Isomorphic Stretching	40
5.10	Dithering	40
5.10.1	Remarks	40
5.10.2	Color Images	41
5.10.3	Bi-tonal Images	42
5.10.4	Guidelines	44
5.11	Error Handling	45

6	Interface Reference	46
6.1	Pdf2Img Interface	46
6.1.1	BilevelThreshold	46
6.1.2	BitmapHeight	46
6.1.3	BitmapWidth	46
6.1.4	BitsPerPixel	46
6.1.5	Center	47
6.1.6	Close	47
6.1.7	CloseImage	47
6.1.8	ColorSpace	47
6.1.9	Compression	48
6.1.10	ConvertFile	48
6.1.11	CreateImage	48
6.1.12	CreateImageInMemory	49
6.1.13	Dithering	49
6.1.14	DPI	49
6.1.15	ErrorCode	50
6.1.16	ErrorMessage	50
6.1.17	FaxHSetting, FaxSSetting	50
6.1.18	FillOrder	51
6.1.19	FilterRatio	51
6.1.20	FitPage	51
6.1.21	GetImage	51
6.1.22	GetOcg	51
6.1.23	HasAnnotations	52
6.1.24	HasColor	52
6.1.25	HasPopups	52
6.1.26	ImageQuality	52
6.1.27	LicenseIsValid	53
6.1.28	OcgCount	53
6.1.29	Open	53
6.1.30	OpenMem	53
6.1.31	Options	54
6.1.32	PageCount	54
6.1.33	PageHeight	54
6.1.34	PageWidth	54
6.1.35	PageNo	55
6.1.36	PreserveAspectRatio	55
6.1.37	ProductVersion	55
6.1.38	Quality	55
6.1.39	RenderingMode	55
6.1.40	RenderPage	56
6.1.41	RepeatWatermark	56
6.1.42	RotateMode	56
6.1.43	SetBitmapDimensions	56
6.1.44	SetCMSEngine	57
6.1.45	SetCMYKProfile	57
6.1.46	SetLicenseKey	58
6.1.47	SetPageSize	58
6.1.48	SetsRGBProfile	58
6.1.49	SetWatermarkImage	58
6.1.50	XDPI, YDPI	59

6.2	Pdf2PdfImg Interface	59
6.2.1	BitsPerPixel	59
6.2.2	Center	59
6.2.3	Close	59
6.2.4	CloseImage	60
6.2.5	ColorSpace	60
6.2.6	Compression	60
6.2.7	ConvertFile	60
6.2.8	CopyLinks	61
6.2.9	CopyOutlines	61
6.2.10	CopyViewerPreferences	61
6.2.11	CreateImage	61
6.2.12	CreateImageInMemory	62
6.2.13	Dithering	62
6.2.14	DPI	63
6.2.15	ErrorCode	63
6.2.16	FitPage	63
6.2.17	GetPdf	63
6.2.18	GrayScale	64
6.2.19	LicenseIsValid	64
6.2.20	Open	64
6.2.21	OpenMem	64
6.2.22	Options	65
6.2.23	PageCount	65
6.2.24	PreserveAspectRatio	65
6.2.25	ProductVersion	65
6.2.26	RenderingMode	66
6.2.27	RenderPage	66
6.2.28	RepeatWatermark	66
6.2.29	RetainText	66
6.2.30	RotateMode	67
6.2.31	SetBitmapDimensions	67
6.2.32	SetLicenseKey	67
6.2.33	SetPageSize	67
6.2.34	SetWatermarkImage	68
6.2.35	XDPI, YDPI	68
6.3	Ocg Interface	68
6.3.1	Label	68
6.3.2	Level	68
6.3.3	Name	69
6.3.4	Visible	69
6.4	Enumerations	70
6.4.1	TPDFColorSpace Enumeration	70
6.4.2	TPDFCompression Enumeration	70
6.4.3	TPDFDithering Enumeration	71
6.4.4	TPDFErrorCode Enumeration	71
6.4.5	TPDFPermission Enumeration	72
6.4.6	TPDFRendererOption Enumeration	73
6.4.7	TPDFRotateMode Enumeration	74

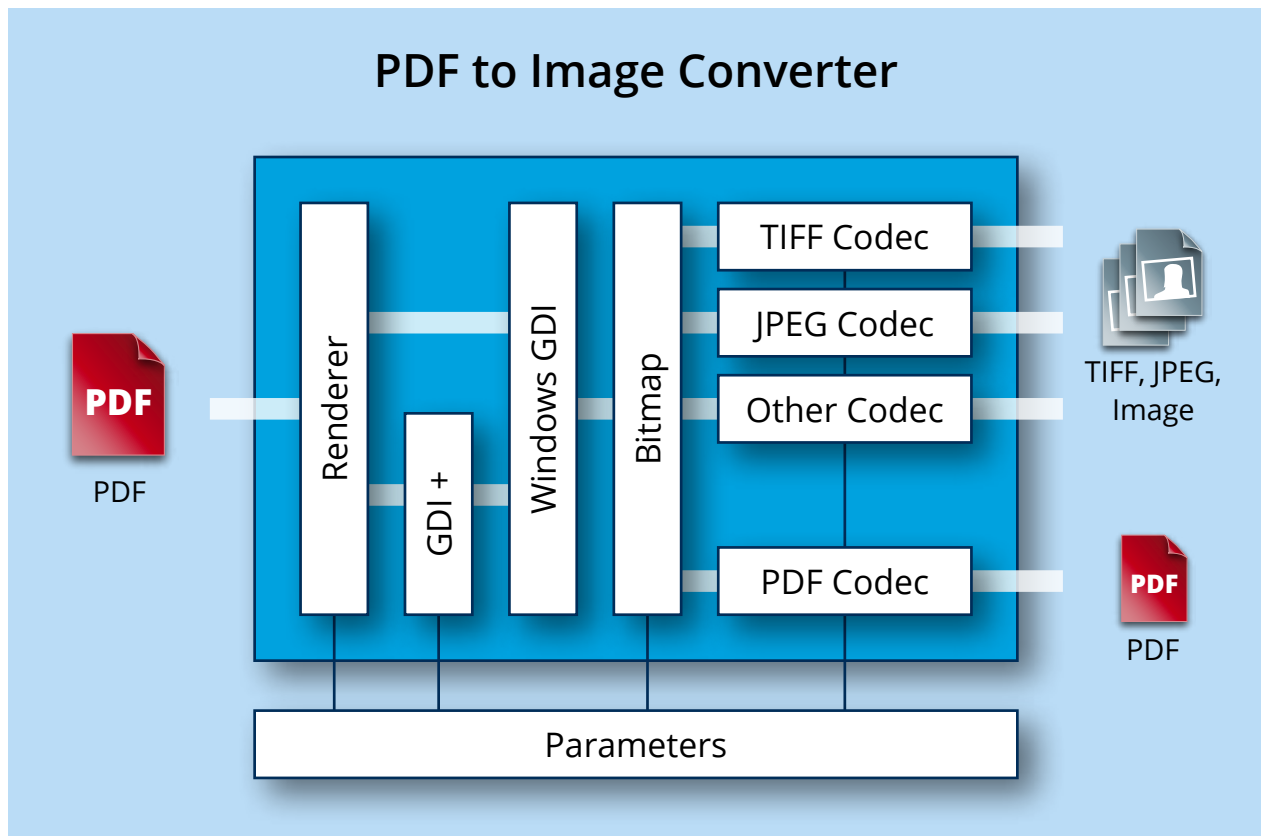
7	Tips, Tricks and Troubleshooting	76
7.1	Font and Text Issues	76
7.1.1	Handle Non-Embedded Fonts	76
	Font Replacement Strategy	76
7.1.2	Handle Embedded Fonts	77
7.2	Transparency	77
8	Version History	78
8.1	Patches in Version 4.12	78
8.2	Changes in Version 4.12	78
8.3	Changes in Version 4.11	78
8.4	Changes in Version 4.10	78
8.5	Changes in Version 4.9	78
8.6	Changes in Version 4.8	78
9	Licensing, Copyright, and Contact	79

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, archive 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)
- 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
- Define minimum line width

PDF to PDF Image

- Raster PDF content (image)
- Keep or remove links, outlines or viewer preferences in PDF output document

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)

Compliance

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
- COM

1.4 Operating Systems

The 3-Heights™ PDF to Image Converter API is available for the following operating systems:

- Windows 7, 8, 8.1, 10 – 32 and 64 bit
- Windows Server 2008, 2008 R2, 2012, 2012 R2, 2016 – 32 and 64 bit

1.5 How to Best Read this Manual

If you are reading this manual for the first time, i.e. would like to evaluate the software, the following steps are suggested.

1. Read the chapter [Introduction](#) to verify this product meets your requirements.
2. Identify what interface your programming language uses.
3. Read and follow the instructions in the chapter [Installation and Deployment](#)
4. In the chapter [Programming Interfaces](#) find your programming language. Please note that not every language is covered in this manual.
For many programming languages there is sample code available. For a start it is generally best to refer to these samples rather than writing code from scratch.
5. (Optional) Read the chapter [User's 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.

The installation of the software requires the following steps.

1. You need administrator rights to install this software.
2. Log in to your download account at <http://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 will find different versions of the product available. We suggest to download the version, which is selected by default. If another is required, it can be selected using the combo box.

The product comes as a ZIP archive containing all files.

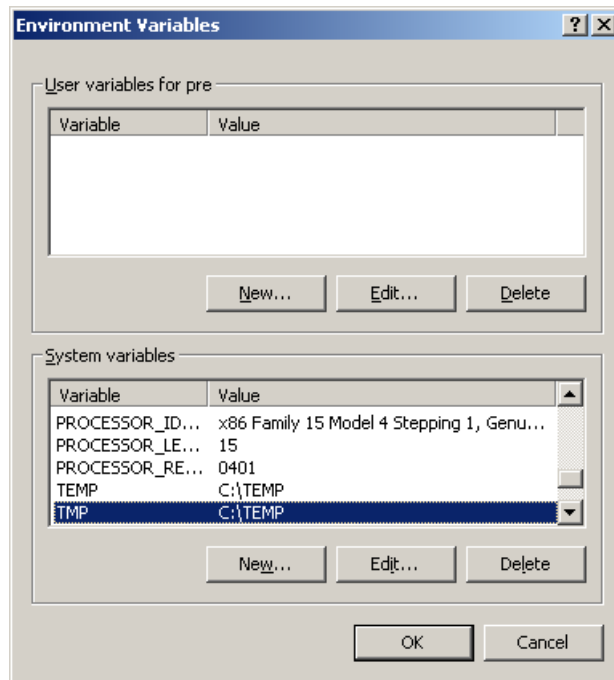
There are 32 and 64-bit versions of the product available. While the 32-bit version runs on both, 32 and 64-bit platforms, the 64-bit version runs on 64-bit platforms only. The ZIP file contains both the 32-bit and the 64-bit version of the product.

3. Unzip the archive to a local folder, e.g. C:\Program Files\PDF Tools AG\.

This creates the following subdirectories:

Subdirectory	Description
bin	Contains the runtime executable binaries.
doc	Contains documentation.
include	Contains header files to include in your C/C++ project.
jar	Contains Java archive files for Java components.
lib	Contains the object file library to include in your C/C++ project.
samples	Contains sample programs in various programming languages

4. (Optional) Register your license key using the [License Management](#).
5. Identify which interface you are using. Perform the specific installation steps for that interface described in chapter [Interface Specific Installation Steps](#)
6. Ensure the system environment variable TMP exists and points to an existing directory. This directory is required to temporarily install fonts that are embedded in PDF documents.
Control Panel → System → Advanced → Environment Variables



7. Ensure the cache directory exists as described in chapter [Special Directories](#).
8. Make sure your platform meets the requirements regarding color spaces and fonts described in chapters [Color Profiles](#) and [Fonts](#) respectively.

2.2 Interfaces

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 examples with which programming languages they can be used.

Interface	Programming Languages
.NET	<p>The MS software platform .NET can be used with any .NET capable programming language such as:</p> <ul style="list-style-type: none"> ■ C# ■ VB .NET ■ J# ■ others <p>This interface is available in the Windows version only.</p>
Java	The Java interface.

COM	<p>The component object model (COM) interface can be used with any COM-capable programming language, such as:</p> <ul style="list-style-type: none"> ■ MS Visual Basic ■ MS Office Products such as Access or Excel (VBA) ■ C++ ■ VBScript ■ others <p>This interface is available in the Windows version only.</p>
C	<p>The native C interface is for use with C and C++.</p>

2.2.1 Development

The software developer kit (SDK) contains all files that are used for developing the software. The role of each file with respect to the four different interfaces is shown in table [Files for Development](#). The files are split in four categories:

Req. This file is required for this interface.

Opt. This file is optional. See also table [File Description](#) to identify which files are required for your application.

Doc. This file is for documentation only.

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

Files for Development

Name	.NET	Java	COM	C
bin\ <platform>\pdf2imgocx.dll< td=""> <td>Req.</td> <td>Req.</td> <td>Req.</td> <td>Req.</td> </platform>\pdf2imgocx.dll<>	Req.	Req.	Req.	Req.
bin*NET.dll	Req.			
bin*NET.xml	Doc.			
doc*.pdf	Doc.	Doc.	Doc.	Doc.
doc\Pdf2ImgOCX.idl			Doc.	
doc\javadoc*.*		Doc.		
include\pdf2imgapi_c.h				Req.
include*.*				Opt.
jar\P2IA.jar		Req.		
lib\ <platform>\pdfimgocx.lib< td=""> <td></td> <td></td> <td></td> <td>Req.</td> </platform>\pdfimgocx.lib<>				Req.
samples*.*	Doc.	Doc.	Doc.	Doc.

The purpose of the most important distributed files of is described in table [File Description](#).

File Description

Name	Description
bin\<platform>\Pdf2ImgOCX.dll	This is the DLL that contains the main functionality (required).
bin\<platform>\PDF2PdfImgOCX.dll	This DLL is for converting PDF documents to rasterized PDF documents (required for converting to PDF).
bin*NET.dll	The .NET assemblies are required when using the .NET interface. The files bin*NET.xml contain the corresponding XML documentation for MS Visual Studio.
doc*.*	Various documentations.
include*.*	Contains files to include in your C / C++ project.
lib\<platform>\PdfImgOCX.lib	The object file library needs to be linked to the C/C++ project.
jar\P2IA.jar	The Java API archive.
samples*.*	Contains sample programs in different programming languages.

2.2.2 Deployment

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

Files for Deployment

Name	.NET	Java	COM	C
bin\<platform>\Pdf2ImgOCX.dll	Req.	Req.	Req.	Req.
bin*NET.dll	Req.			
jar\P2IA.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 into an installation routine such as an MSI file or 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 `Pdf2ImgOCX.dll` must be distributed.

¹ These files must reside in the same directory as `Pdf2ImgOCX.dll`.

3. All files are copied to the target location using a batch script. This script contains the following commands:

```
copy application.exe %targetlocation%\.  
copy Pdf2ImgOCX.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%\Pdf2ImgOCX.dll.
```

2.3 Interface Specific Installation Steps

2.3.1 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 the registration of `Pdf2ImgOCX.dll`. Note that in Windows Vista and later, the command needs to be executed from an administrator shell.

```
regsvr32 "C:\Program Files\PDF Tools AG\bin\<platform>\Pdf2ImgOCX.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`.²

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 `Pdf2ImgOCX.dll`.

2.3.2 Java Interface

The 3-Heights™ PDF to Image Converter API requires Java version 6 or higher.

For compilation and execution When using the Java interface, the Java wrapper `jar\P2IA.jar` needs to be on the CLASSPATH. This can be done 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\P2IA.jar" sample.java
```

For execution Additionally the library `Pdf2ImgOCX.dll` needs to be in one of the system's library directories³ or added to the Java system property `java.library.path`. This can be achieved by either adding it dynamically at

² Otherwise you get the following message: `LoadLibrary("Pdf2ImgOCX.dll") failed - The specified module could not be found.`

³ On Windows defined by the environment variable `PATH` and e.g. on Linux defined by `LD_LIBRARY_PATH`.

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` depending on the platform of the Java VM⁴.

```
java -classpath ".;C:\Program Files\PDF Tools AG\P2IA.jar" ^
-Djava.library.path=C:\Program Files\PDF Tools AG\bin\x64 sample
```

Note that when creating rasterized PDF documents instead of images, the above statements apply to `PCIA.jar` and `Pdf2PdfImgOCX.dll` instead.

2.3.3 .NET Interface

The 3-Heights™ PDF to Image Converter API does not provide a pure .NET solution. Instead, it consists of .NET assemblies, which are added to the project and a native DLL, which is called by the .NET assemblies. This has to be accounted for when installing and deploying the tool.

The .NET assemblies (`*.NET.dll`) are to be added as references to the project. They are required at compilation time.

`Pdf2ImgOCX.dll` is not a .NET assembly, but a native DLL. It is not to be added as a reference in the project.

The native DLL `Pdf2ImgOCX.dll` is called by the .NET assembly `Pdf2ImgNET.dll`.

`Pdf2ImgOCX.dll` must be found at execution time by the Windows operating system. The common way to do this is adding `Pdf2ImgOCX.dll` as an existing item to the project and set its property “Copy to output directory” to “Copy if newer”.

Alternatively the directory where `Pdf2ImgOCX.dll` resides can be added to the environment variable `%Path%` or it can simply be copied manually to the output directory.

When creating rasterized PDF documents instead of images, the above statement applies to `Pdf2PdfImgOCX.dll` instead.

When using the new rendering engine (R2), the statements apply for `Pdf2ImgAPI.dll` instead.

2.3.4 C Interface

- The header file `pdf2imgapi_c.h` needs to be included in the C/C++ program.
- The library `PdfImgOCX.lib` needs to be linked to the project.
- The dynamic link library `Pdf2ImgOCX.dll` needs to be in a path of executables (e.g. on the environment variable `%PATH%`).

2.4 Uninstall, Install a New Version

If you have used the ZIP file for the installation: In order to uninstall the product, undo all the steps done during installation, e.g. un-register using `regsvr32.exe /u`, delete all files, etc.

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

2.5 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.

⁴ 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`

2.6 Special Directories

2.6.1 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 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.

2.6.2 Cache Directory

The cache directory is used for data that is persisted 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 will degrade 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

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

2.6.3 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 will always take precedence over system fonts.

Windows

1. %SystemRoot%\Fonts
2. directory Fonts, which must be a direct sub-directory of where Pdf2ImgOCX.dll resides.

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](#) will fail with an error code and error message indicating the reason.

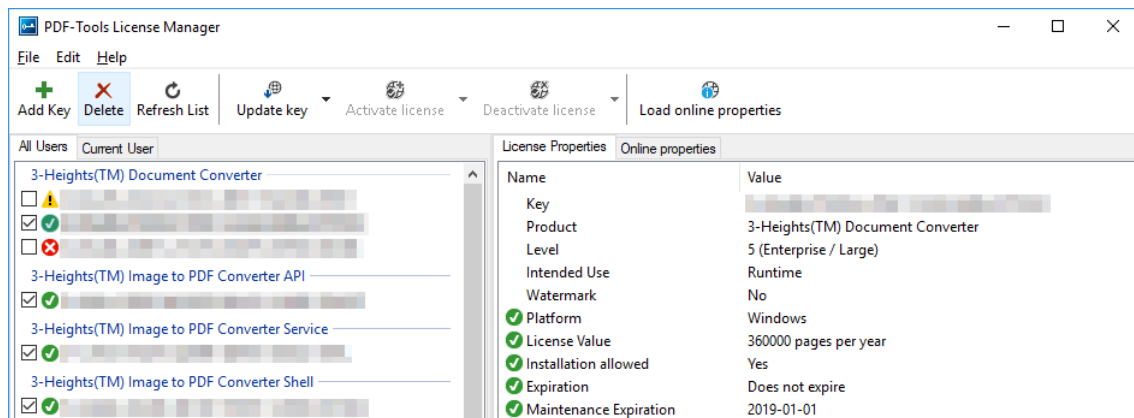
3.1 License Installation and Management

There are three possibilities to pass the license key to the application:

1. The license key is installed using the GUI tool (graphical user interface). This is the easiest way if the licenses are managed manually. It is only available on Windows.
2. The license key is installed using the shell tool. This is the preferred solution for all non-Windows systems and for automated license management.
3. The license key is passed to the application at run-time via the [SetLicenseKey](#) method. This is the preferred solution for OEM scenarios.

3.1.1 Graphical License Manager Tool

The GUI tool `LicenseManager.exe` is located in the `bin` directory of the product kit (Windows only).



List all installed license keys

The license manager always shows a list of all installed license keys in the left pane of the window. This includes licenses of other PDF Tools products. The user can choose between:

- Licenses available for all users. Administrator rights are needed for modifications.
- Licenses available for the current user only.

Add and delete license keys

License keys can be added or deleted with the “Add Key” and “Delete” buttons in the toolbar.

- The “Add key” button installs the license key into the currently selected list.
- The “Delete” button deletes the currently selected license keys.

Display the properties of a license

If a license is selected in the license list, its properties are displayed in the right pane of the window.

3.1.2 Command Line License Manager Tool

The command line license manager tool `licmgr` is available in the `bin\x86` and `bin\x64` directory.

Note: The command line tool `licmgr` is not included in Windows platform kits, as the GUI tool is the recommended tool for managing Licenses. A Windows `licmgr` shelltool is available on request.

A complete description of all commands and options can be obtained by running the program without parameters:

```
licmgr
```

List all installed license keys

```
licmgr list
```

The currently active license for a specific product is marked with a `*` on the left side.

Example:

```
>licmgr list
Local machine:
  Product Name:
    1-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
    1-YYYYY-YYYYY-YYYYY-YYYYY-YYYYY-YYYYY-YYYYY
    * 1-ZZZZZ-ZZZZZ-ZZZZZ-ZZZZZ-ZZZZZ-ZZZZZ-ZZZZZ
Current user:
```

Add and delete license keys

Install new license key:

```
licmgr store 1-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

Delete old license key:

```
licmgr delete 1-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

Both commands have the optional argument `-s` that defines the scope of the action:

g For all users

u Current user

Display the properties of a license

```
licmgr info 1-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

Properties that invalidate the license are marked with an X, properties that require attention are marked with an !. In that case an additional line with a comment is displayed.

Example:

```
>licmgr info 1-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
- Key:          1-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
- Product:      Product Name
- Features:     Feature1,Feature2
- Intended use: Development
- Watermark:    No
- Platform:     Windows
- Installation: Yes
! Activation:   2018-05-07
                (The license has not yet been activated.)
- Expiration:   Does not expire
- Maintenance: 2019-04-27
```

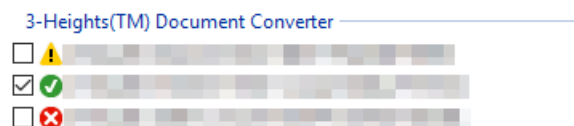
3.2 License Selection and Precedence

3.2.1 Selection

If multiple keys for the same product are installed in the same scope, only one of them can be active at the same time.

Installed keys that are not selected are not considered by the software!

In the Graphical User Interface use the check box on the left side of the license key to mark a license as selected.



With the Command Line Interface use the `select` subcommand:

```
licmgr select 1-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

3.2.2 Precedence

License keys are considered in the following order:

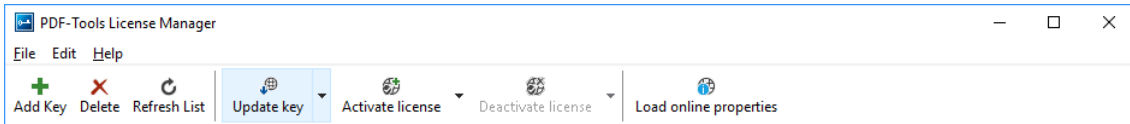
1. License key passed at runtime.
2. License selected for the current user
3. License selected for the current user ([legacy key format](#))
4. License selected for all users
5. License selected for all users ([legacy key format](#))

The first matching license is used, regardless whether it is valid or not.

3.3 Key Update

If a license property like the maintenance expiration date changes, the key can be update directly in the license manager.

In the Graphical User Interface select the license and press the button "Update Key" in the toolbar:



With the Command Line Interface use the update subcommand:

```
licmgr update 1-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

3.4 License activation

New licenses keys have to be activated (except for OEM licenses).

Note: Licenses that need activation have to be installed in the license manager and must not be passed to the component at runtime.

The license activation is tied to a specific computer. If the license is installed at user scope, the activation is also tied to that specific user. The same license key can be activated multiple times, if the license quantity is larger than 1.

Every license key includes a date, after which the license has to be activated, which is typically 10 days after the issuing date of the key. Prior to this date, the key can be used without activation and without any restrictions.

3.4.1 Activation

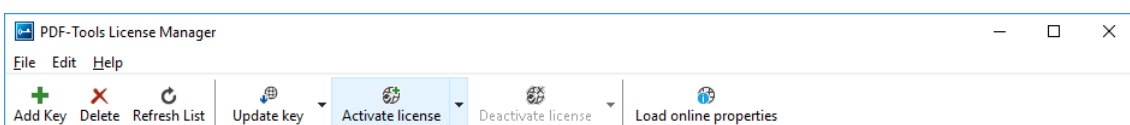
The License can be activated directly within the license manager. Every activation increases the activation count of the license by 1.

It is recommended to add a comment to the activation request which helps keeping track of all activations for a specific license key. In case of problems it also helps us providing support.

The comment is stored in the activation database as long as the license key remains activated. Upon deactivation it is deleted from the database immediately.

All activations and the corresponding comments can be examined using the **Load online properties** function of the license manager. The information is accessible to anyone with access to the license key.

In the Graphical User Interface select the license and press the button "Activate license" in the toolbar:



It is recommended to add a comment to the activation request by using the subsequent dialog box.

With the Command Line Interface use the `activate` subcommand:

```
licmgr activate 1-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

Note that the key has to be installed first.

It is recommended to add a comment to the activation request by using the `-c` or `-cd` option:

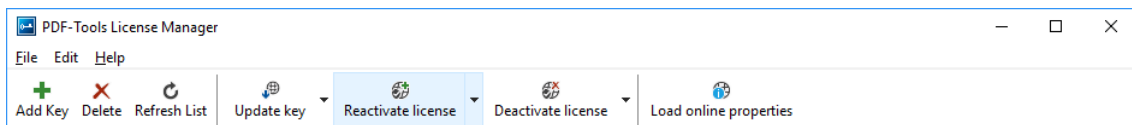
```
licmgr activate -cd 1-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX  
licmgr activate -c "custom comment" 1-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

3.4.2 Reactivation

The activation is tied to specific properties of the computer like the MAC address or host name. If one of these properties changes, the activation becomes invalid and the license has to be reactivated. A reactivation does **not** increase the activation count on the license.

The process for reactivation is the same as for the activation.

In the Graphical User Interface the button "Activate license" changes to "Reactivate license":



With the Command Line Interface the subcommand `activate` is used again:

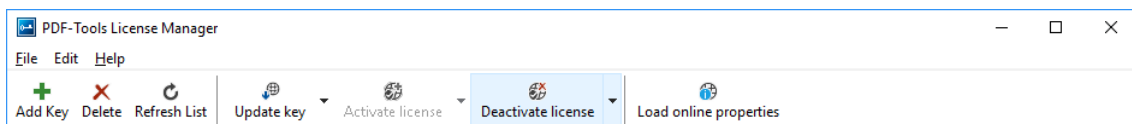
```
licmgr activate 1-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

3.4.3 Deactivation

To move a license to a different computer, it has to be deactivated first. Deactivation decreases the activation count of the license by 1.

The process for deactivation is similar to the activation process.

In the Graphical User Interface select the license and press the button "Deactivate license" in the toolbar:



With the Command Line Interface use the `deactivate` subcommand:

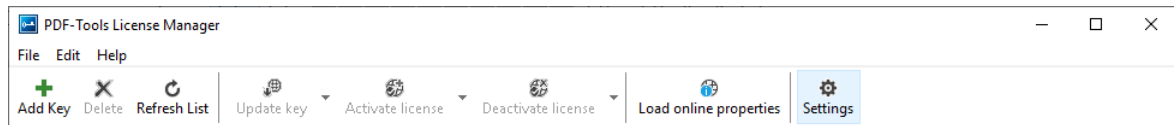
```
licmgr deactivate 1-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

3.5 Proxy Setting

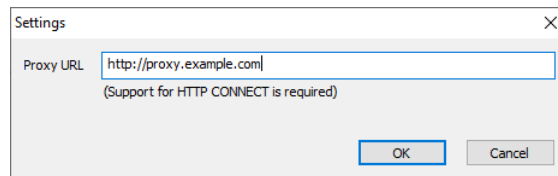
A proxy URL can be configured for computers that cannot access the internet without a web proxy.

Note: The proxy must allow connections via HTTP CONNECT to the server www.pdf-tools.com:443.

In the Graphical User Interface press the button "Settings" in the toolbar:



and enter the proxy URL in the respective field:



3.6 Offline Usage

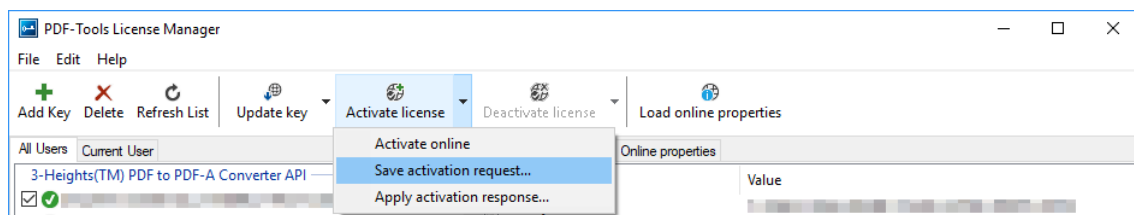
The following actions in the license manager need access to the internet:

- [License Activation](#)
- [License Reactivation](#)
- [License Deactivation](#)
- [Key Update](#)

On systems without internet access, a three step process can be used instead, using a form on the PDF Tools website.

3.6.1 First Step: Create a Request File

In the Graphical User Interface select the license and use the dropdown menu on the right side of the button in the toolbar:



With the Command Line Interface use the `-fs` option to specify the destination path of the request file:

```
licmgr activate -fs activation_request.bin 1-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

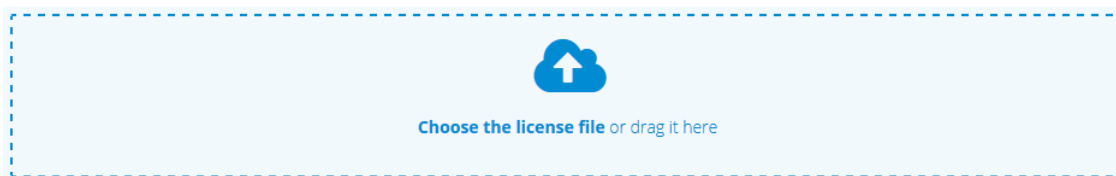
License Deactivation: When saving the deactivation request file, the license is **deactivated immediately** and cannot be used any further. It can however only be activated again after completing the deactivation on the website.

3.6.2 Second Step: Use Form on Website

Open the following website in a web browser: <http://www.pdf-tools.com/pdf20/en/mypdftools/licenses-kits/license-activation/> Upload the request by dragging it onto the marked area:

License activation (offline)

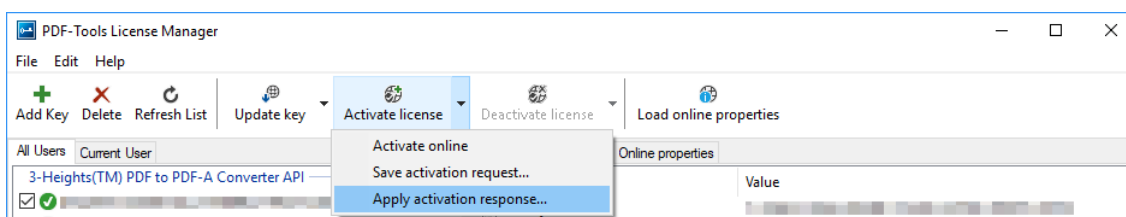
Upload your license request. For more information and instructions please check the manual of your product.



Upon success, the response will be downloaded automatically if necessary.

3.6.3 Third Step: Apply the Response File

In the Graphical User Interface select the license and use the dropdown menu on right side of the button in the toolbar:



With the Command Line Interface use the `-fl` option to specify the source path of the response file:

```
licmgr activate -fl activation_response.bin 1-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

3.7 License Key Versions

As of 2018 all new keys will have the format 1-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX. Legacy keys with the old format 0-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX are still accepted for a limited time period.

For compatibility reasons, old and new version keys can be installed side by side and one key of each version can be selected at the same time. In that case, the software always uses the new version.

3.8 License Key Storage

Depending on the platform the license management system uses different stores for the license keys.

3.8.1 Windows

The license keys are stored in the registry:

- "HKLM\Software\PDF Tools AG" (for all users)

- "HKCU\Software\PDF Tools AG" (for the current user)

3.9 Troubleshooting

3.9.1 License key cannot be installed

The license key cannot be installed in the license manager application. The error message is: "Invalid license format."

Possible causes:

- The license manager application is an older version that only supports the [legacy key format](#).

Solution

Use a current version of the license manager application or use a license key in the legacy key format if available.

3.9.2 License is not visible in license manager

The license key was successfully installed previously but is not visible in the license manager anymore. The software is still working correctly.

Possible causes:

- The license manager application is an older version that only supports the [legacy key format](#).

Solution

Use a current version of the license manager application.

3.9.3 License is not found at runtime

The license is not found at runtime by the software. The error message is: "No license key was set."

Possible causes:

- The license key is actually missing (not installed).
- The license key is installed but not selected in the license manager.
- The application is an older version that only supports the [legacy key format](#), while the license key has the new license format.

Solution

Install and select a valid license key that is compatible with the installed version of the software or use a newer version of the software. The new license key format is supported starting with version 4.10.26.1

For compatibility reasons, one license key of each format can be selected at the same time.

3.9.4 Eval watermark is displayed where it should not

The software prints an evaluation watermark onto the output document, even if the installed license is a productive one.

Possible causes:

- There is an evaluation license key selected for the **current user**, that takes precedence over the key for **all users**.

Note: The software might be run under a different user than the license manager application.

- An evaluation license key that is passed at runtime takes precedence over those selected in the license manager.
- There is an evaluation license key selected with a [newer license format](#) that takes precedence over the key in the older format.
- The software was not restarted after changing the license key from an evaluation key to a productive one.

Solution

Disable or remove all evaluation license in all scopes, check that no evaluation key is passed at runtime and restart the software.

3.9.5 Activation is not recognized

The license is installed and activated in the license manager, but the software does not recognize it as activated. The error message is: "The license has not been activated."

Possible causes:

- There is an unregistered license key selected for the **current user**, that takes precedence over the key for **all users**. This leads to an error even if the same license is registered for all users.

Note: The software might be run under a different user than the license manager application.

- A license key that is passed at runtime takes precedence over those selected in the license manager. This leads to an error even if the same license is registered in the license manager.

Note: Licenses that need activation have to be installed in the license manager and must not be passed to the component at runtime.

- The software was not restarted after activating the license.

Solution

Disable, remove or activate all unregistered licenses in all scopes, check that no key is passed at runtime and restart the software.

3.9.6 Activation is invalidated too often

The license activation is invalidated regularly, for no obvious reason.

Possible causes:

- The MAC address used for computing the machine fingerprint is not static. This may happen e.g. for virtual network adapters with dynamic MAC address (VPN, Juniper, ...).

Solution

Update to a newer version (≥ 4.12) of the PDF Tools product, deactivate the license key using the new license manager and activate it again. After that, an improved fingerprinting algorithm is used.

Deactivation and activation have to be **executed separately**, a reactivation of the license in one step does not change the fingerprinting algorithm and thus does not solve the problem.

Note: After this procedure, older products might not recognize the activation as valid anymore. Reactivating the license using an old license manager will revert the activation to the old fingerprinting algorithm.

As an alternative, remove any virtual network adapter with a dynamic MAC address.

3.9.7 Connection to the licensing service fails

The license activation/deactivation/update fails because the license manager cannot reach the licensing server.

The error message depends on the platform and the exact error condition.

Possible causes:

- The computer is not connected to the internet.
- The connection is blocked by a corporate firewall.

Solution

Make sure that the computer is connected to the internet and that the host `www.pdf-tools.com` is reachable on port 443 (HTTPS).

If this is not possible, try [Offline Usage](#) instead.

3.9.8 Offline usage fails due to a request/response mismatch

The offline license activation/deactivation/update fails because the response file does not match the request file.

The error message is: "Mismatch between request and response."

Possible causes:

- The response file is applied to a different machine than the request file was created.
- The response file is applied to a different user than the request file was created.
- The response file was applied to a specific user while the request was created for all users, or vice versa.
- The response file is applied to the wrong license key.
- Another request file has been created between creating the request file and applying the response file.
- The license key was updated between creating the request file and applying the response file.
- The license key was removed and re-added between creating the request file and applying the response file.

Solution

Delete any old request and response files to make sure they are not used by accident.

Retry the entire process as outlined in [chapter 3.6](#) and refrain from making any other license-related actions between creating the request file and applying the response file.

Make sure that the response file is applied to exactly the same license key in exactly the same location (machine, all users or specific user) where the request file was created.

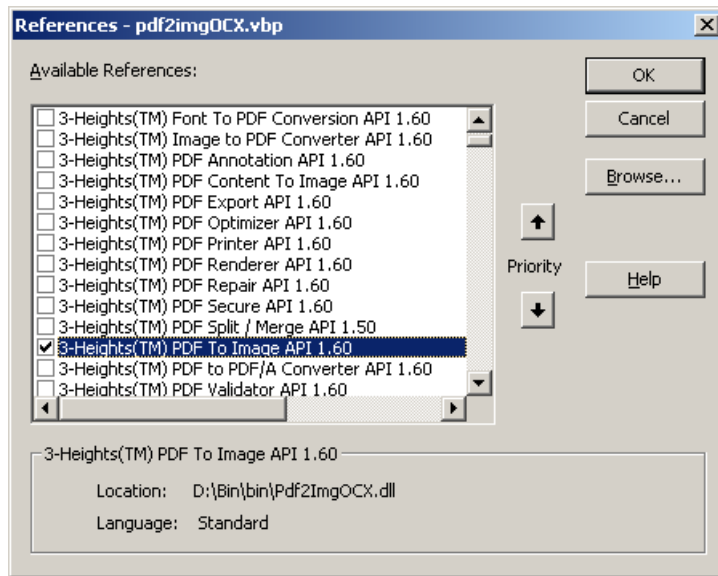
4 Programming Interfaces

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 .vbp in the directory samples/VB/. You can either use this sample as a base for an application, or you can start from scratch.

If you start from scratch, here is a quick start guide:

1. First 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 if you like.
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 PDF2IMGOCLib.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, a bit longer, but also more powerful approach, is dividing 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 which does this could look like that:

```
Private Sub Command1_Click()  
    Dim conv As New PDF2IMGOCLib.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

```

Note: Applying the digital signature is optional (you may also change the certificate name), therefore the two corresponding lines could be skipped.

4.2 ASP

The COM name of the class, for example used in ASP or PHP, of the 3-Heights™ PDF to Image Converter API is `PDF2IMGOCX.Pdf2Img`.

Here is a small ASP sample using VBScript:

```

<%@ Language=VBScript %>
<%
option explicit
dim conv set
conv = Server.CreateObject("Pdf2ImgOCX.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

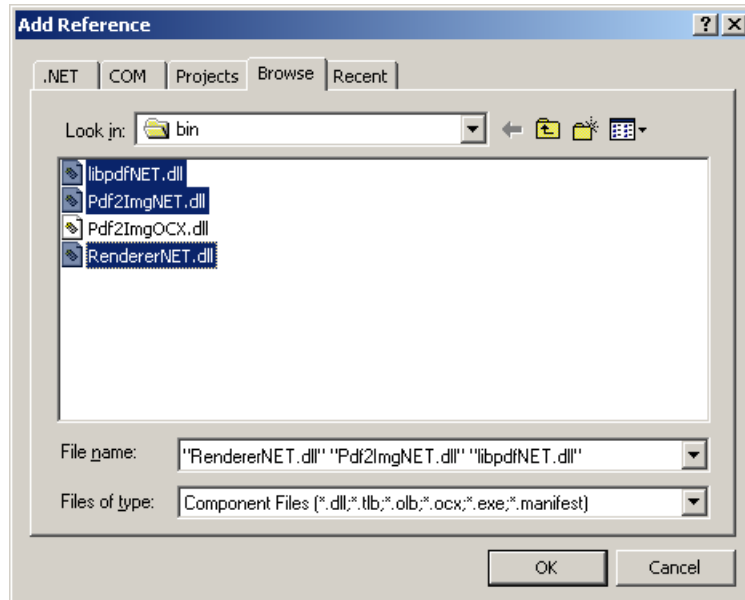
There should be at least one .NET sample for MS Visual Studio available in the ZIP archive of the Windows version of the 3-Heights™ PDF to Image Converter API. The easiest for a quick start is to refer to this sample.

In order to create a new project from scratch, do the following steps:

1. Start Visual Studio and create a new C# or VB project.
2. Add references to the .NET assemblies.

To do so, in the "Solution Explorer" right-click your project and select "Add Reference...". The "Add Reference" dialog will appear. In the tab "Browse", browse for the .NET assemblies `libpdfNET.dll`, `RendererNET.dll`, and `Pdf2ImgNET.dll`.

Add them to the project as shown below:

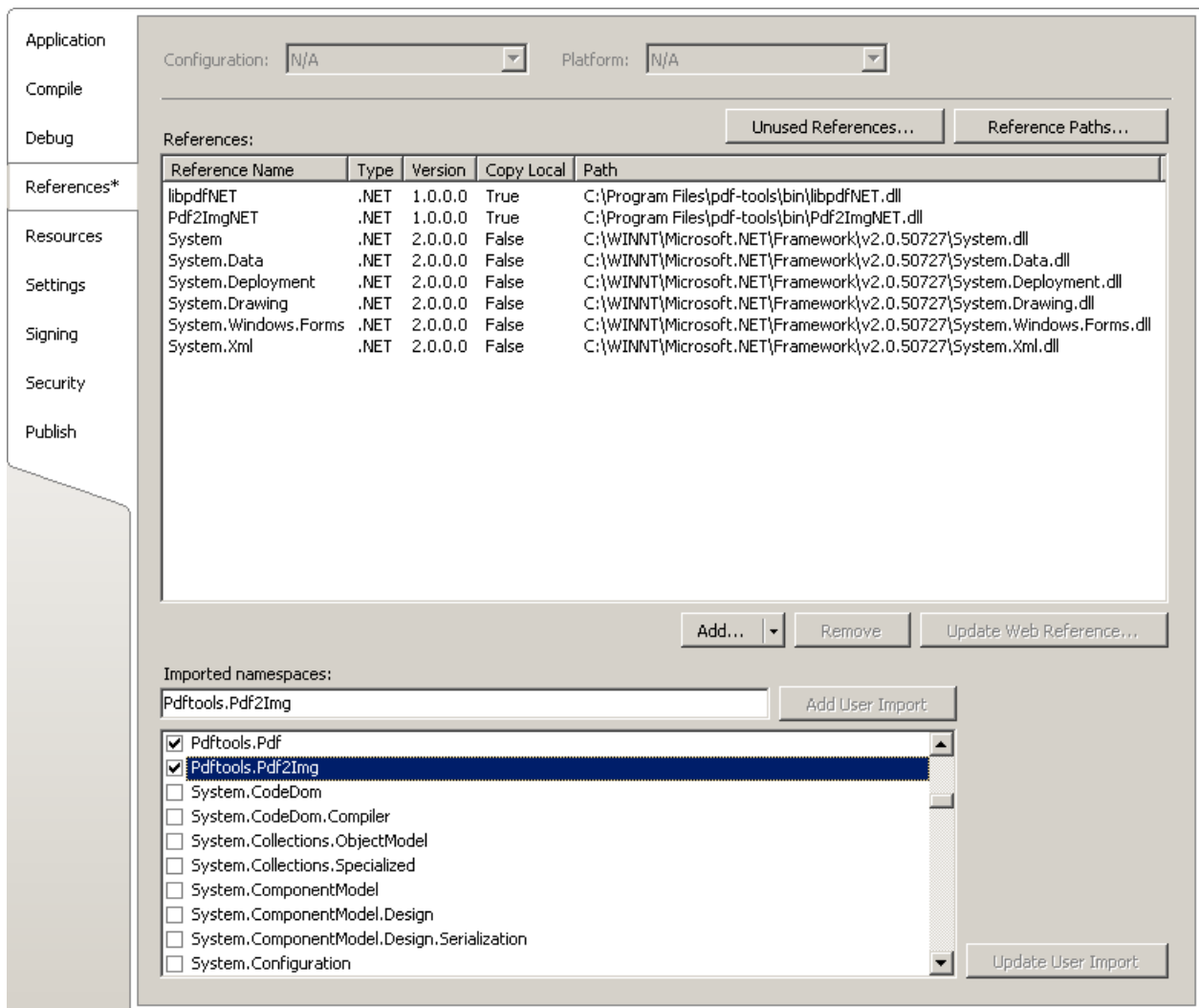


3. Import namespaces (Note: This step is optional, but useful.)
4. Write your code.

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. Hereby it is crucial that depending on the solution platform (x86 or x64) the matching native DLL Pdf2ImgOCX.dll (from the directory bin\Win32 or bin\x64) is copied to the output directory.
2. The executable is created in the directory bin\Release.
3. For deployment, the executable and all .NET assemblies must be copied into the same folder on the target computer. The .NET assemblies of the 3-Heights™ PDF to Image Converter API have the file name bin*NET.dll.
4. At runtime, the native DLL Pdf2ImgOCX.dll must be found on the target computer by the DLL search sequence. To ensure this, the DLL must either be copied to the folder containing the executable or to a directory on the environment variable Path (e.g. %SystemRoot%\system32).
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 Pdf2ImgOCX.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, distinguishable by the inner exception (property [InnerException](#)):

System.DllNotFoundException Unable to load DLL Pdf2ImgOCX.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 the respective issue.

Troubleshooting: DllNotFoundException

This means, that the native DLL Pdf2ImgOCX.dll could not be found at execution time.

Resolve this by either:

- adding Pdf2ImgOCX.dll as an existing item to your project and set its property "Copy to output directory" to "Copy if newer", or
- adding the directory where Pdf2ImgOCX.dll resides to the environment variable %Path%, or
- copying Pdf2ImgOCX.dll to the output directory of your project.

Troubleshooting: BadImageFormatException

The exception means, that the native DLL Pdf2ImgOCX.dll has the wrong "bitness" (i.e. platform 32 vs. 64 bit). There are two versions of Pdf2ImgOCX.dll available: 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.

The platform of the application's process is defined by the project's platform configuration for which there are 3 possibilities:

AnyCPU This means, that the application will run as a 32-bit process on 32-bit Windows and as 64-bit process on 64-bit Windows. When using AnyCPU one has to use a different native DLL, depending on the platform of Windows. This can be ensured 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).

x86 This means, that the application will always run as 32-bit process, regardless of the platform of the Windows installation. The 32-bit DLL runs on all systems, which makes this the simplest configuration. Hence, if an application needs to be portable and does not require any specific 64-bit features, it is recommended to use this setting.

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

5 User's Guide

5.1 Supported Codecs

The following table lists which capabilities of the different codecs are supported by the 3-Heights™ PDF to Image Converter API.

Codec Capabilities

Codec	Bits per Pixel	Gray	Indexed	Quality	Compression
TIFF	1,2,3,4,8,24 ⁵	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 ⁵	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) ⁶
PBM	1-8, 24	Yes	No	No	Raw
EPS	1, 2, 4, 8, 24 ⁵	Yes	No	No	Raw

Codec The **Compression/Decompression** Type.

Bits Per Pixel The supported values for bits per pixel. 1 = bi-tonal, 8 = 256 colors/grey scales, 24 = True Color

Gray This format supports grey scale.

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.

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

⁵ For palette creation: The number of palette entries is equal to 2 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.

⁶ To create lossless JPEG2000 images, set the quality parameter to 100. For values <100, a lossy compression algorithm is applied.

Here are some suggestions of what image type could be selected for which purpose.

Lossless

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

Lossy

- Gray scale 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 How to Create Multi and Single-Page Images

5.2.1 Multi-Page Images

The TIFF format is an image format which 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 How to Set Pixels Equal Points

How to 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. In order 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. Please note that generally this yields in a grainy image when viewed at 100% zoom, since the monitor uses a resolution of 96 DPI.

5.4 How to Reduce the File Size

There are different ways to reduce the file size of an image. One needs 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 zoomed into. A larger value generates a more detailed image, but also will increase 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 grey scale instead of 24-bit true color will reduce the file size. Keep in mind that not all formats support all color depths.

8-bit grey scale 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 gray scale 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. How much 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.

Please consult 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 grey scale 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 therefore 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 chapter [Dithering](#).

5.5 How to Use 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 Pdf2ImgOCX.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 data base.

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

```
Private Sub OpenFromMemory_Click()  
    ' (1) Write PDF document to memory  
    Dim conv As New Pdf2ImgOCX.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

The 3-Heights™ PDF to Image Converter API PDF Rendering Engine works in the RGB color space. Other color spaces such as CMYK or spot colors are first converted to RGB before rendering.

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 sub-directory of where the Pdf2ImgOCX.dll resides.

5.6.2 Set other Color Profiles

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

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:

- <http://www.pdf-tools.com/public/downloads/resources/colorprofiles.zip>
- <http://www.color.org/srgbprofiles.html>
- https://www.adobe.com/support/downloads/iccprofiles/iccprofiles_win.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 cope with font issues, please refer to section [Font and Text Issues](#).

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 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 :

Windows: In a directory named `Fonts`, which must be a direct sub-directory of where `Pdf2ImgOCX.dll` resides.

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; e.g. 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 amongst 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 How to Change the Colors—Obtain 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 CMKY color profiles can be found at the following location: %System-Root%\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 Pdf2ImgOCX.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 will 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 How to Apply 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 Pdf2ImgOCX.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 grey need to be simulated with other colors (e.g. only black/white pixels).

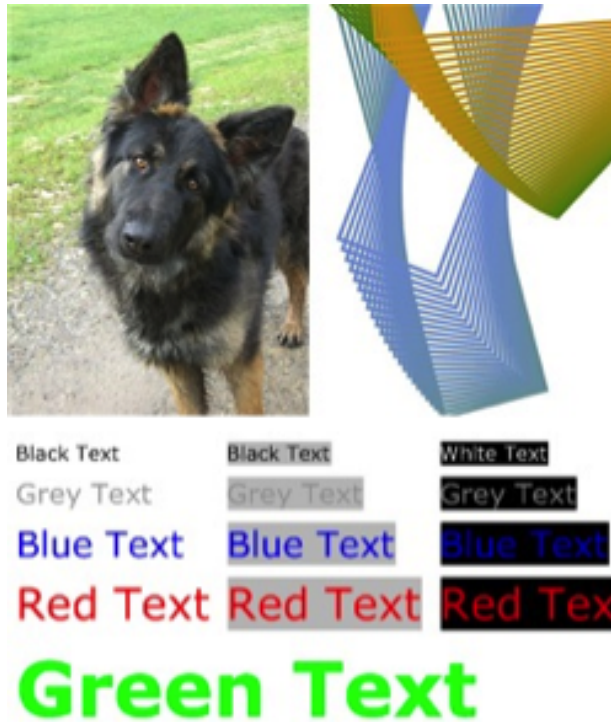
5.10.1 Remarks

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 is, the higher the quality of the image.

- The rendering filter and current zoom level of the PDF viewing application may have an additional impact on how the images below are displayed.

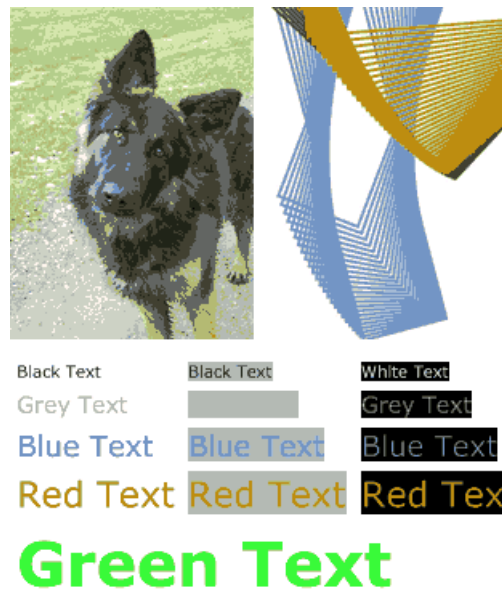
5.10.2 Color Images

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

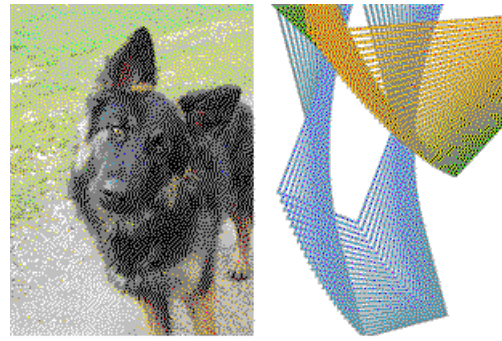


A 24 bit RGB color image can have up to 16.7 millions of different colors. Dithering does not need to be applied since all required colors exist and none need to be simulated.

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



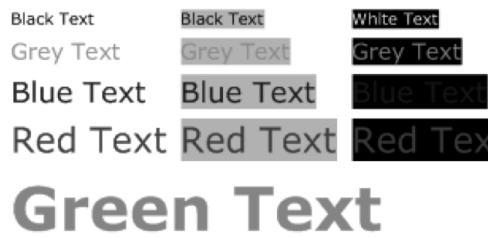
- Color Space 16 colors (4 bit)
- Dithering Floyd-Steinberg
- File Size as PNG 18 kB
- + Renders details better
- + Usually better overall quality, especially in photographic images than without dithering
- Sometimes generates unwanted artifacts (striking pixels)
- Larger file size then without dithering



5.10.3 Bi-tonal Images

(The 8 bit image just acts as reference.)

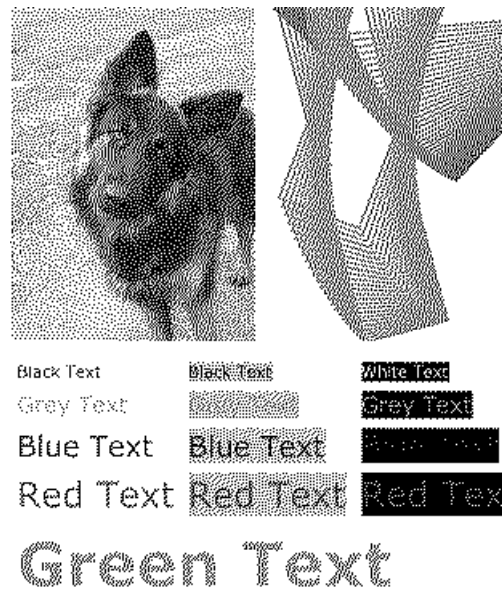
- Color Space Grayscale (8 bit)
- Dithering None
- File Size as PNG 46 kB



- 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, which dithering type is normally 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 grey scale) 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, 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. Which return values have to be interpreted as failures is documented in the chapter [Interface Reference](#). To identify the error on a programmatic level, check the property [ErrorCode](#). The property [ErrorMessage](#) provides a human readable error message, describing 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) however 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 Pdf2ImgOCX.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 bi-tonal 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. Bi-tonal: `1`, gray scale: `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 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.

6.1.7 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.

6.1.8 ColorSpace

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

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

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

6.1.9 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.

6.1.10 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.

6.1.11 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)
- `.j2` (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.

6.1.12 CreateImageInMemory

Method: Boolean `CreateImageInMemory(String Extension)`

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

Parameter:

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

Returns:

True The image could successfully be created.

False Otherwise.

6.1.13 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 chapter [Dithering](#).

6.1.14 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](#).

6.1.15 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 chapter [Error Handling](#)). See also enumeration [TPDFErrorCode](#). PDF-Tools 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.

6.1.16 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 chapter [Error Handling](#))

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

6.1.17 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
```

6.1.18 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.

6.1.19 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.

6.1.20 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.

6.1.21 GetImage

Method: Variant `GetImage()`

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

6.1.22 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.

6.1.23 HasAnnotations

Method: Boolean HasAnnotations(Long IPageNo)

Return **True** if the selected page contains annotations, **False** otherwise.

6.1.24 HasColor

Method: Boolean HasColor(Long IPageNo)

Return **True** if the selected page contains colors, **False** otherwise.

6.1.25 HasPopups

Method: Boolean HasPopups(Long IPageNo)

Return **True** if the selected page contains popups, **False** otherwise.

6.1.26 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.

6.1.27 LicenseIsValid

Property (get): Boolean `LicenseIsValid`
Static

Check if the license is valid.

6.1.28 OcgCount

Property (get): Long `OcgCount`

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

See also [GetOcg](#).

6.1.29 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 properties [ErrorCode](#) and [ErrorMessage](#) for additional information.

6.1.30 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.

6.1.31 Options

Property (get, set): TPDFRendererOption **Options**
Default: eOptionBicubic + eOptionHighQuality + eOptionTransparency

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 please see [TPDFRendererOption](#).

6.1.32 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.

6.1.33 PageHeight

Property (get): Float **PageHeight**

Return the height of the page in points.

6.1.34 PageWidth

Property (get): Float **PageWidth**

Return the width of the page in points.

6.1.35 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.

6.1.36 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 will fit into the defined dimensions, given by [SetBitmapDimensions](#).

6.1.37 ProductVersion

Property (get): String ProductVersion

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

6.1.38 Quality

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

Use [ImageQuality](#) instead.

6.1.39 RenderingMode

Property (get, set): RenderingMode RenderingMode

There are two rendering modes: [eModeAccurate](#) (default), and [eModeFast](#).

The accurate mode uses the Windows GDI+ for rendering. This mode allows for image filtering, sub-pixel rendering and anti-aliasing.

The fast mode uses the Windows GDI for rendering. This mode is generally faster and better suitable to create bitonal output.

6.1.40 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.

6.1.41 RepeatWatermark

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

Place the watermark set by [SetWatermarkImage](#) as adjacent tiles over the whole page.

6.1.42 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](#).

6.1.43 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.

6.1.44 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, however, 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 section [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.

6.1.45 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.

6.1.46 SetLicenseKey

```
Method: Boolean SetLicenseKey(String LicenseKey)
```

Set the license key.

6.1.47 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.

6.1.48 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.

6.1.49 SetWatermarkImage

```
Method: Boolean SetWatermarkImage(String FileName, Single Left, Single Bottom)
```

Set a watermark image from a file at the specified position in points. The watermark is placed once, unless the property [RepeatWatermark](#) is set to **True**.

Only RGB color space images can be used as watermarks.

6.1.50 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 Pdf2PdfImg Interface

This Interface is included in the Pdf2PdfImgOCX.dll.

This interface takes a PDF document as input and creates a rasterized PDF document as output. Rasterized means the document no longer contains vector graphics or texts, but only one raster image per page.

6.2.1 BitsPerPixel

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

Get or set the color depth. Bi-tonal: **1**, gray scale: **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](#)).

6.2.2 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.

6.2.3 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.

6.2.4 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.

6.2.5 ColorSpace

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

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

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

6.2.6 Compression

Property (get, set): TPDFCompression Compression
Default: eComprFlate

Get or set the compression used for the images in the PDF file.

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

6.2.7 ConvertFile

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

Convert a complete PDF file to a PDF-image file.

Parameters:

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

ImageFileName [String] The PDF output 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 input 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.

6.2.8 CopyLinks

Property (get, set): Boolean CopyLinks
Default: True

Get or set the option to copy links.

6.2.9 CopyOutlines

Property (get, set): Boolean CopyOutlines
Default: True

Get or set the option to copy outlines (bookmarks).

6.2.10 CopyViewerPreferences

Property (get, set): Boolean CopyViewerPreferences
Default: True

Get or set the option to copy viewer preferences which include: PageLayout, PageMode and OpenActions.

6.2.11 CreateImage

Method: Boolean CreateImage(String FileName, String UserPwd, String OwnerPwd, TPDFPermission PermissionFlags, Long KeyLength)

Create a new PDF-image file.

Parameters:

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

UserPwd [String] (optional) Set the user password of the PDF document. If this parameter is omitted, the default password is used. Use `0` to set no password.

OwnerPwd [String] (optional) Set the owner password of the PDF document. If this parameter is omitted, the default password is used. Use `0` to set no password.

PermissionFlags [TPDFPermission] (optional) The permission flags. By default no permissions are granted. The permissions that can be granted are described in the enumeration [TPDFPermission](#).

KeyLength [Long] (optional) With this option the length of the encryption key can be set. Due to the nature of the MD5 algorithm, the maximum length is limited to **128** bits. The minimum is **40** and it is required to be a multiple of **8**. The two most commonly used values are **40** (standard encryption) and **128** (requires Acrobat 5 or higher). Keep in mind that Acrobat only supports **40** and **128** bit encryption. Other tools, such as the 3-Heights™ tools also support other encryption key lengths. The default value is calculated based on the selected permission flags.

Returns:

True The file was created successfully.

False Otherwise.

To not encrypt the output document, set `PermissionFlags` to `-1`, user and owner password to `0`.

In order to allow high quality printing, flags `ePermPrint` and `ePermDigitalPrint` need to be set.

6.2.12 CreateImageInMemory

Method: Boolean `CreateImageInMemory()`

Store the PDF in memory as a byte array. See also method [GetPdf](#).

Returns:

True The output PDF document was successfully created in-memory.

False Otherwise.

6.2.13 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 chapter [Dithering](#).

6.2.14 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](#).

6.2.15 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 chapter [Error Handling](#)). See also enumeration [TPDFErrorCode](#). PDF-Tools 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.

6.2.16 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.

6.2.17 GetPdf

Method: Variant `GetPdf()`

Get the output file from memory. See also method [CreateImageInMemory](#)

Returns:

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

6.2.18 GrayScale

[Deprecated] Property (get, set): Boolean `GrayScale`

Default: `False`

Use [ColorSpace](#) instead.

6.2.19 LicenseIsValid

Property (get): Boolean `LicenseIsValid`

Static

Check if the license is valid.

6.2.20 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 properties [ErrorCode](#) and [ErrorMessage](#) for additional information.

6.2.21 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.

6.2.22 Options

Property (get, set): TPDFRendererOption Options
Default: eOptionBicubic + eOptionHighQuality

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 please see [TPDFRendererOption](#).

6.2.23 PageCount

Property (get): Long PageCount
Default: undef.

Return the number of pages in the PDF document. If the return value is **undef**, no PDF document is open.

6.2.24 PreserveAspectRatio

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

Keep the ratio of the page dimensions. If set to **True** and only one value of the page dimensions is provided, the other value is calculated.

6.2.25 ProductVersion

Property (get): String ProductVersion

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

6.2.26 RenderingMode

Property (get, set): RenderingMode RenderingMode

There are two rendering modes: `eModeAccurate` (default), and `eModeFast`.

The accurate mode uses the Windows GDI+ for rendering. This mode allows for image filtering, sub-pixel rendering and anti-aliasing.

The fast mode uses the Windows GDI for rendering. This mode is generally faster and better suitable to create bitonal output.

6.2.27 RenderPage

Method: Boolean RenderPage(Long PageNumber)

Render the assigned page number in the PDF input file to the PDF output file.

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.

6.2.28 RepeatWatermark

Property (get, set): Boolean RepeatWatermark
Default: **False**

Place the watermark set by [SetWatermarkImage](#) as adjacent tiles over the whole page.

6.2.29 RetainText

Property (get, set): Boolean RetainText
Default: **False**

If set to **True**, all text is copied as text to the output document and placed in front of a background image that contains all non-text content. If, in the original PDF, not all of the text content is placed on top of other content then the text might be un-covered by graphics objects such as white rectangles (redaction objects).

If set to **False**, all content is rendered to the image.

6.2.30 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](#).

6.2.31 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.

6.2.32 SetLicenseKey

Method: Boolean SetLicenseKey(String LicenseKey)

Set the license key.

6.2.33 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.

6.2.34 SetWatermarkImage

Method: Boolean `SetWatermarkImage(String FileName, Single Left, Single Bottom)`

Set a watermark image from a file at the specified position in points. The watermark is placed once, unless the property [RepeatWatermark](#) is set to **True**.

Only RGB color space images can be used as watermarks.

6.2.35 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.

6.3 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 methods [OcgCount](#) and [GetOcg](#) from the [Pdf2Img](#) interface.

6.3.1 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.

6.3.2 Level

Property (get): Long `Level`

In user interfaces OCGs can be shown in a tree. The property `Level` 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`.

6.3.3 Name

Property (get): `String Name`

Return the name of the OCG.

6.3.4 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.

Note that though 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

6.4 Enumerations

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

6.4.1 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	Gray-scale CMYK	Only the K channel is used

6.4.2 TPDFCompression Enumeration

TPDFCompression Table

TPDFCompression	Description
eComprRaw	No compression
eComprJPEG	Joint Photographic Expert Group

TPDFCompression Table

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.

6.4.3 TPDFDithering Enumeration

TPDFDithering Table

TPDFDithering	Description
eDitherNone	No dithering
eDitherFloydSteinberg	Floyd-Steinberg (Default)
eDitherHalftone	Half-toning
eDitherPattern	Pattern Dithering
eDitherG3Optimized	Dithering optimized to compress well with Group 3
eDitherG4Optimized	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.

6.4.4 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's header file `bseerror.h`, the javadoc documentation of `com.pdfutils.NativeLibrary.ERRORCODE` and the .NET documentation of `Pdfutils.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	<p>The file contains unrendered XFA form fields, i.e. the file is an XFA and not a PDF file.</p> <p>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.</p>

6.4.5 TPDFPermission Enumeration

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

TPDFPermission Table

TPDFPermissionFlag	Description
ePermPrint	Low resolution printing
ePermModify	Changing the document
ePermCopy	Content copying or extraction
ePermAnnotate	Annotations
ePermFillForms	Filling of form fields
ePermSupportDisabilities	Support for disabilities
ePermAssemble	Document assembly
ePermDigitalPrint	High resolution printing

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

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

Allow Printing

```
Permission = Permission Or ePermPrint
```

Prohibit Printing

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

6.4.6 TPDFRendererOption Enumeration

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

Example: Visual Basic

Enable or disable an option, and leave all other options untouched:

```
' Enable high quality rendering (anti-aliasing)
.Options = .Options OR eOptionHighQuality
' Disable high quality rendering (anti-aliasing)
.Options = .Options AND NOT eOptionHighQuality
```

Example: C/C++

```
int iOptions = Pdf2ImgGetOptions (pDocument);
// Enable high quality rendering (anti-aliasing)
Pdf2ImgSetOptions (pDocument, iOptions | eOptionHighQuality);
// Disable high quality rendering (anti-aliasing)
Pdf2ImgSetOptions (pDocument, iOptions & ~eOptionHighQuality);
```

The following list includes renderer options that are relevant for the 3-Heights™ PDF to Image Converter API. Note that there are more enumerations available, but they are unrelated to this API.

TPDFRendererOption Table

TPDFRendererOption	
eOptionAutoAccurateMode	Detect content that cannot be rendered using RenderingMode eModeFast (GDI) and switch to eModeAccurate (GDI+) automatically, e.g. to render transparent tiling patterns. Does not have any effect if eModeAccurate was already set.
eOptionBilinear	A bilinear image filter is applied to images to improve the image quality. This option cannot be combined with eOptionBicubic .
eOptionDisableAnnots	When setting this option flag then annotations are not drawn.
eOptionDisableBPC	If this option flag is set then the black point compensation feature is disabled when converting colors e.g. from CMYK to RGB.

TPDFRendererOption Table

<code>eOptionDisableContent</code>	When setting this option flag then only form fields and annotations are drawn without the underlying page content.
<code>eOptionDisableFilter</code>	Disable image filtering. Images are scaled using the nearest-neighbor algorithm, which improves performance at the cost of rendering quality.
<code>eOptionDisablePatterns</code>	Disable patterns.
<code>eOptionDrawPopups</code>	Draw pop up windows of annotations, such as sticky notes.
<code>eOptionHighQuality</code>	(default) Anti-aliasing for text and path objects and filtering of image objects can be turned off and on with this option.
<code>eOptionNoBackground</code>	Do not paint a white background. With this option, the background of the output will be transparent, if supported.
<code>eOptionNoEmbedded</code>	Do not use embedded fonts. Instead fonts from the operating system's font directory are used (%Systemroot%\fonts).
<code>eOptionOutlines</code>	Convert fonts into vector graphics.
<code>eOptionOverprint</code>	This option enables overprint simulation, i.e. the overprint (OP, op) and overprint mode (OPM) operators in the PDF content stream. Otherwise these operators are ignored. Use this option with caution since overprint simulation is not fully supported. In particular overprint simulation of separation and device-n colors does not produce the expected results.
<code>eOptionPreInstalled</code>	Replace embedded fonts with a pre-installed font if the same font is already installed on the OS.
<code>eOptionPrint</code>	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.
<code>eOptionPrintOnlySig</code>	Draw the digital signature appearance only (without any status appearances, e.g. valid or invalid).
<code>eOptionDoNotPrintSig</code>	Do not draw digital signature appearances.
<code>eOptionTransparency</code>	Deprecated option that has no effect.
<code>eOptionTrueType</code>	CFF and Type1 fonts are converted to True Type fonts. This option overrides option <code>eOptionType1</code> .
<code>eOptionType1</code>	CFF fonts are converted to Type1 fonts.
<code>eOptionUseFastImages</code>	Always print images in fast mode. This should help resolving performance issues with complex images and image masks of documents that are to be printed in accurate mode.

6.4.7 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 Chapter [Fonts](#)).
 2. See Section [Handle Non-Embedded Fonts](#).
2. For issues with text using embedded fonts:
 1. Ensure embedded fonts are used (i.e. `eOptionNoEmbedded` is not set).
 2. Ensure the two system environment variables TEMP and TMP exist and point to an existing directory. These variables not being set is a common error source for service applications that run under a user that has no temporary directory and thus cannot install fonts. See also Chapter [Installation and Deployment](#).
 3. See Section [Handle Embedded Fonts](#).

7.1.1 Handle Non-Embedded Fonts

Font Replacement Strategy

This section describes the exact behavior of font handling of the rendering engine. It is rather technical and it is not required to be understood in order to properly use the software.

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

1. If the font is not embedded or `eOptionPreInstalled` is set:
 - 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 font⁷ 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 the font is embedded, it is converted to a Windows compatible font and temporarily installed. If `eOption-NoEmbedded` is used then the glyphs of the fonts are converted to either bitmaps or outlines⁸. If `eOptionOut-lines` is used then the glyphs are converted to outlines only.
4. If the font is not embedded and the Unicodes are available then the nearest font from the installed font collection is tailored to the metrics of the font.

⁷ e.g. Times-Roman, Helvetica, Courier

⁸ The outline of a glyph is a vector graphic without any reference to the original font program.

7.1.2 Handle Embedded Fonts

The following list provides possible work-arounds if text is printed incorrectly. Options should be tried in ascending order.

1. Using the option `eOptionNoEmbedded` inhibits all embedded fonts from being used in the spool file and the printer hardware. Instead the glyphs are converted to either bitmaps or outlines. Using the option `eOptionOutlines` at the same time the conversion is restricted to outlines.
2. Using the option `eOptionPreInstalled` inhibits embedded fonts which have the same name as the corresponding installed font from being used. This option can also be used to reduce the number of fonts in a spool file if the printer hardware memory capacity is limited.
3. Pre-render the page in a bitmap and send the pre-rendered image to the printer (`eOptionBitmap`). This results in large spool files.

7.2 Transparency

The 3-Heights™ rendering engine supports transparency functions such as a number of blend modes as well as isolated and non-isolated transparency groups, but not transparency in general.

Certain types of tiling and shading patterns may not correctly be reproduced by the rendering engine.

8 Version History

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

8.1 Patches in Version 4.12

Note that the version number of the initial “final release” is 4.12.26.3.

Patch 4.12.26.4

- **Improved** error messages for failed HTTP connections in various situations (including license manager).
- **Added** missing documentation and release note for the proxy setting in the GUI license manager.
- **Improved** license reactivation behavior of the commandline license manager (licmgr): The server is now only contacted if necessary.
- **Improved** behavior of license manager when dealing with licenses of unreleased products.

8.2 Changes in Version 4.12

- **New** HTTP proxy setting in the GUI license manager.

8.3 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.

8.4 Changes in Version 4.10

- **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.

8.5 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`⁹.

8.6 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.

⁹ This has no effect on neither the .NET, Java, nor COM API

9 Licensing, Copyright, and Contact

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

PDF Tools AG 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's manual is also copyright protected; it may be copied and given away provided that it remains unchanged including the copyright notice.

Contact

PDF Tools AG
Kasernenstrasse 1
8184 Bachenbülach
Switzerland
<http://www.pdf-tools.com>
pdfsales@pdf-tools.com