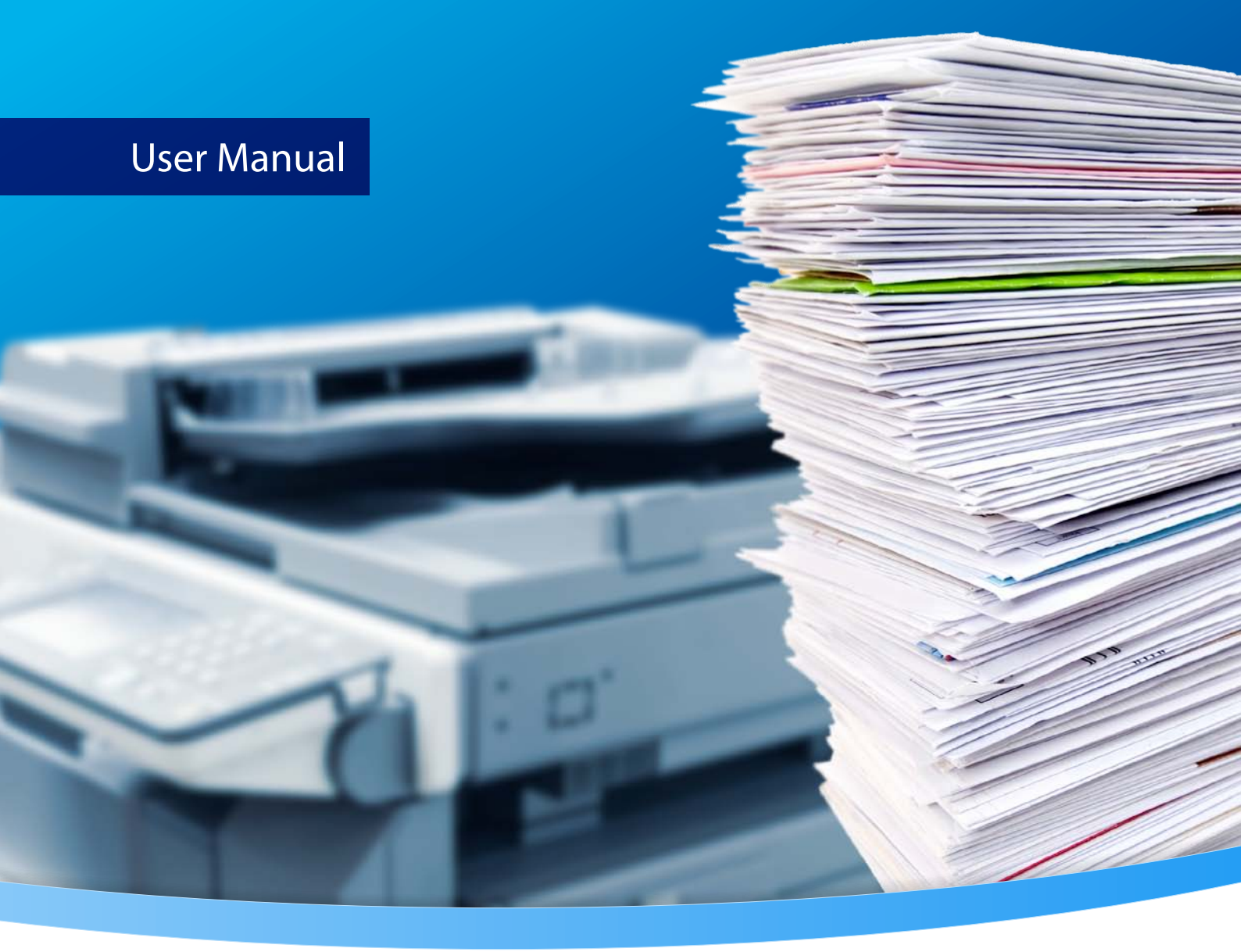
User Manual



3-Heights®
Image to PDF Converter API

PDF/A conform

Version 6.27.6

pdftools

Contents

1 Introduction 6
1.1 Description 6
1.2 Functions 6
1.2.1 Features 6
1.2.2 Formats 7
1.2.3 Conformance 8
1.3 Interfaces 8
1.4 Operating systems 8
1.5 How to best read this manual 9

2 Installation and deployment 10
2.1 Windows 10
2.2 Linux and macOS 11
2.2.1 Linux 11
2.2.2 macOS 12
2.3 ZIP archive 12
2.3.1 Development 12
2.3.2 Deployment 14
2.4 NuGet package 15
2.5 Interface-specific installation steps 15
2.5.1 COM interface 15
2.5.2 Java interface 16
2.5.3 .NET interface 16
2.5.4 C interface 17
2.6 Uninstall, Install a new version 17
2.7 Color profiles 17
2.7.1 Default color profiles 17
2.7.2 Set other color profiles 17
2.7.3 Get other color profiles 18
2.8 Fonts 18
2.8.1 Font cache 18
2.9 Note about the evaluation license 18
2.10 Special directories 18
2.10.1 Directory for temporary files 18
2.10.2 Cache directory 19
2.10.3 Font directories 19

3 License management 21

4 Programming interfaces 22
4.1 Visual Basic 6 22
4.2 .NET 23
4.2.1 Visual Basic 23
4.2.2 C# 24
4.2.3 Deployment 24
4.2.4 Troubleshooting: TypeInitializationException 25

5 User guide 26
5.1 Overview of the API 26
5.1.1 About the 3-Heights® Image to PDF Converter API 26
5.1.2 How does the API work? 26
5.1.3 Using with the PDF Prep Tool Suite 26
5.1.4 OCR recognition of images 27
5.2 Error handling 27

6 Interface reference 28
6.1 `Img2Pdf` Interface 28
6.1.1 `AdjustOrientation` 28
6.1.2 `AdjustPage` 28
6.1.3 `Alt` 29
6.1.4 `BitonalCompression` 29
6.1.5 `BitsPerPixel` 29
6.1.6 `BorderSize` 29
6.1.7 `CenterImage` 29
6.1.8 `Close` 30
6.1.9 `Compliance` 30
6.1.10 `ContinuousCompression` 31
6.1.11 `Create` 31
6.1.12 `CreateInMemory` 32
6.1.13 `CreatePageFromCodec` 32
6.1.14 `CreatePageFromImageFile` 32
6.1.15 `CreatePageFromFile` 33
6.1.16 `DefaultDPI` 33
6.1.17 `Dithering` 33
6.1.18 `ErrorCode` 34
6.1.19 `ErrorMessage` 34
6.1.20 `ExportText` 34
6.1.21 `FitImage` 35
6.1.22 `GetOCREngine` 35
6.1.23 `GetOCREngineCount` 35
6.1.24 `GetOCRPluginCount` 35
6.1.25 `GetOCRPluginName` 36
6.1.26 `GetPdf` 36
6.1.27 `ImageQuality` 36
6.1.28 `IndexedCompression` 37
6.1.29 `InfoEntry` 37
6.1.30 `Lang` 37
6.1.31 `LicenseIsValid` 38
6.1.32 `Linearize` 38
6.1.33 `OCREmbedOCRImage` 38
6.1.34 `OCRBitonalRecognition` 39
6.1.35 `OCRDeskewImage` 39
6.1.36 `OCREmbedBarcodes` 39
6.1.37 `OCRResolutionDPI` 39
6.1.38 `OCRThresholdDPI` 40
6.1.39 `Orientation` 40
6.1.40 `Quality` 40
6.1.41 `ProductVersion` 40
6.1.42 `Recompress` 40

6.1.43 ResolutionDPI ... 41
6.1.44 SetColorSpaceProfile ... 41
6.1.45 SetLicenseKey ... 41
6.1.46 SetMetadata ... 41
6.1.47 SetMetadataStream ... 42
6.1.48 SetOCREngine ... 42
6.1.49 SetOCRLanguages ... 43
6.1.50 SetOCRParams ... 43
6.1.51 SetOutputIntent ... 44
6.1.52 SetPageSize ... 44
6.1.53 ThresholdDPI ... 45
6.2 PDFCodec Interface ... 45
6.2.1 BitsPerComponent ... 45
6.2.2 Close ... 45
6.2.3 ColorSpace ... 46
6.2.4 ComponentsPerPixel ... 46
6.2.5 Compression ... 46
6.2.6 Create ... 46
6.2.7 CreateInMemory ... 47
6.2.8 Decode ... 47
6.2.9 DefaultDPI ... 47
6.2.10 ErrorCode ... 47
6.2.11 ErrorMessage ... 48
6.2.12 fXDPI, fYDPI ... 48
6.2.13 GetImage ... 48
6.2.14 Height ... 48
6.2.15 ImageQuality ... 48
6.2.16 IsPremultipliedAlpha ... 49
6.2.17 Mask ... 49
6.2.18 Name ... 49
6.2.19 Open ... 49
6.2.20 OpenMem ... 50
6.2.21 Page ... 50
6.2.22 PageCount ... 50
6.2.23 PageNo ... 50
6.2.24 Palette ... 50
6.2.25 Quality ... 51
6.2.26 Recompress ... 51
6.2.27 Samples ... 51
6.2.28 SMask ... 51
6.2.29 Width ... 51
6.2.30 XDPI, YDPI ... 52
6.3 Img2Img Interface ... 52
6.3.1 AllowResampling ... 52
6.3.2 BitonalCompression ... 52
6.3.3 ContinuousCompression ... 52
6.3.4 ContinousCompression ... 52
6.3.5 ConvertFile ... 53
6.3.6 CopyPage ... 53
6.3.7 DPI ... 54
6.3.8 ErrorCode ... 54
6.3.9 ErrorMessage ... 54

6.3.10 Height 55
6.3.11 ImageQuality 55
6.3.12 IndexedCompression 55
6.3.13 TrueHeight 55
6.3.14 TrueWidth 55
6.3.15 Quality 56
6.3.16 Width 56
6.4 ImgOcr Interface 56
6.4.1 GetFirstOcrText 56
6.4.2 GetNextOcrText 56
6.4.3 GetOCRPluginCount 56
6.4.4 GetOCRPluginName 57
6.4.5 Recognize 57
6.4.6 SetOCREngineName 57
6.4.7 SetImage 57
6.4.8 SetOCRLanguages 57
6.4.9 SetOCRParams 57
6.5 OcrText Interface 57
6.5.1 BaseLine 58
6.5.2 FontName 58
6.5.3 FontSize 58
6.5.4 Rect 58
6.5.5 StringLength 58
6.5.6 Text 58
6.6 PdfOcr Interface 59
6.6.1 Close 59
6.6.2 ErrorCode 59
6.6.3 ErrorMessage 59
6.6.4 GetOCRPluginCount 59
6.6.5 GetOCRPluginName 60
6.6.6 GetPdf 60
6.6.7 LicenseIsValid 61
6.6.8 OCRBitonalRecognition 61
6.6.9 OCRDeskewImage 61
6.6.10 OCREmbedBarcodes 61
6.6.11 OCREmbedOCRImage 62
6.6.12 OCRMode 62
6.6.13 OCRResolutionDPI 62
6.6.14 OCRRotatePage 63
6.6.15 OCRThresholdDPI 63
6.6.16 ProductVersion 63
6.6.17 Open 63
6.6.18 OpenMem 64
6.6.19 SaveAs 64
6.6.20 SaveInMemory 65
6.6.21 SetLicenseKey 65
6.6.22 SetOCREngine 65
6.6.23 SetOCRLanguages 66
6.6.24 SetOCRParams 66
6.7 Enumerations 67
6.7.1 TPDFColorSpace Enumeration 67
6.7.2 TPDFCompliance Enumeration 67

6.7.3 TPDFCompression Enumeration ... 68
6.7.4 TPDFDithering Enumeration ... 69
6.7.5 TPDFErrorCode Enumeration ... 69
6.7.6 TPDFOrientation Enumeration ... 70
6.7.7 TPDFPermission Enumeration ... 70
6.8 Supported image extensions ... 71
6.9 Supported image compression types ... 72
6.9.1 No compression (raw) ... 72
6.9.2 DCT (JPEG) ... 72
6.9.3 Flate (ZIP) ... 73
6.9.4 LZW ... 73
6.9.5 CCITT Fax Group 3 and 4 ... 73
6.9.6 JBIG2 ... 74
6.9.7 JPEG2000 ... 74
6.10 Specification of resolution and image dimensions ... 74

7 Version history ... 76
7.1 Changes in versions 6.19–6.27 ... 76
7.2 Changes in versions 6.13–6.18 ... 76
7.3 Changes in versions 6.1–6.12 ... 76
7.4 Changes in version 5 ... 76
7.5 Changes in version 4.12 ... 76
7.6 Changes in version 4.11 ... 77
7.7 Changes in version 4.10 ... 77
7.8 Changes in version 4.9 ... 77
7.9 Changes in version 4.8 ... 77

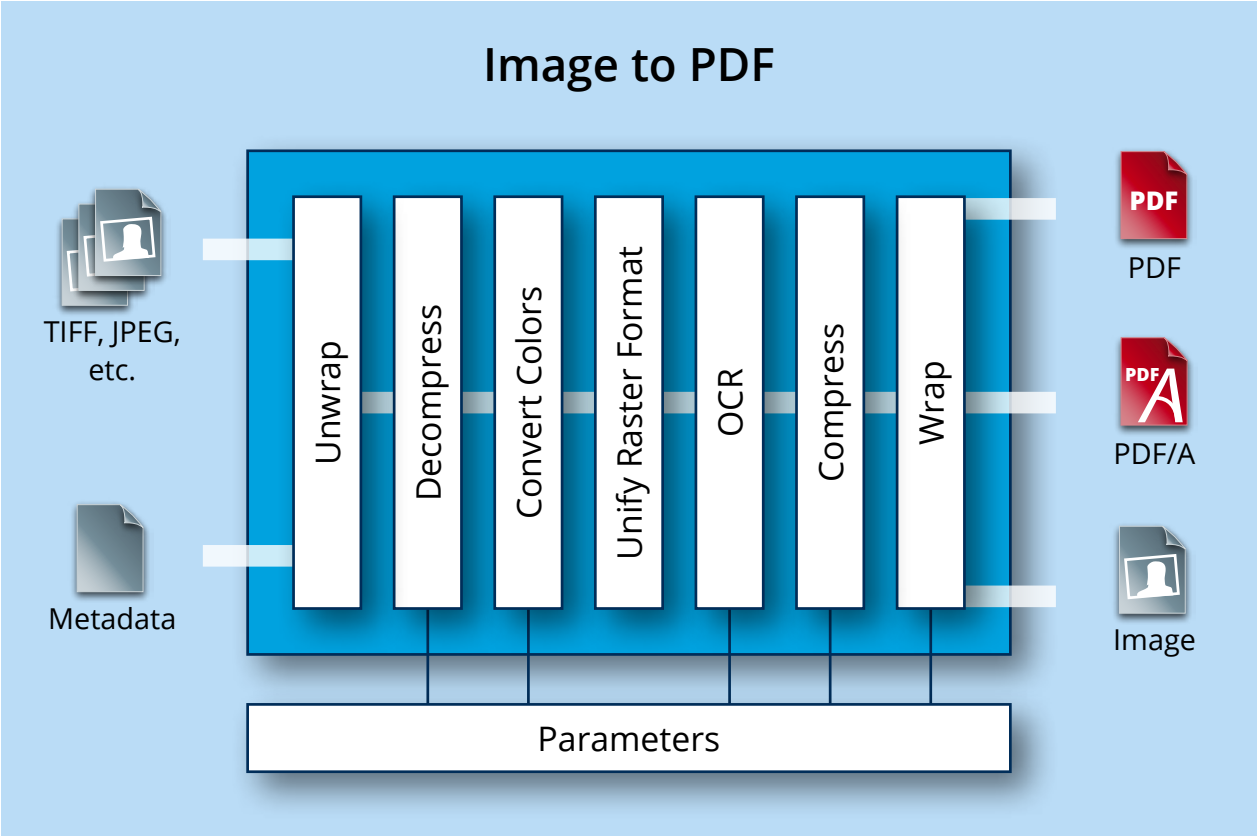
8 Licensing, copyright, and contact ... 79

1 Introduction

1.1 Description

The 3-Heights® Image to PDF Converter API converts raster image formats to PDF and PDF/A. PDF/A has been acknowledged world-wide as the ISO standard for long-term archiving since 2005. The Image to PDF Converter is used to convert images into a standardized format, for instance for electronic archiving or electronic data exchange.

It is also possible to include metadata from external sources. The Converter is characterized by a robust design, high throughput and accurate image reproduction. The optional OCR add-in makes output files searchable in full text mode.



1.2 Functions

The 3-Heights® Image to PDF Converter API converts raster image formats such as JPEG, TIFF or PNG to PDF or PDF/A. It can merge pages from various image files to form a single PDF and can also split multi-page image files into single-page PDF files. Further options include defining page size and resolution, image scaling and the inclusion of (external) metadata. Optical character recognition (OCR) is also available as an option.

1.2.1 Features

Image to PDF

- Conversion of single-page or multi-page raster images to PDF
- Set PDF conformance

- Automatic or selectable image compression, depending on the image type
- Automatic or selectable PDF page size
- Selectable page area
- Selectable image quality for lossy compression
- Set image position
- Set scaling
- Set standard resolution (DPI / X and Y coordinates)
- Set encryption and user access permissions
- Selectable and embeddable ICC color profile
- Define alternative text (tagging) and image language
- Set document attributes
- Optional JPEG image recompression
- Set image orientation (portrait or landscape)
- Optical character recognition (OCR)
- Embedding XMP metadata
- Support for image masks
- Support for mixed raster content (MRC)

Image to Image

- Split single-page or multi-page raster images into individual, single page images
- Merge multiple images to form one multi-page image
- Convert to an image format of the same color depth
- Modify TIFF image compression
- Set quality index for lossy image compression
- Create lossless JBIG2 images and lossy/lossless JPEG2000
- Set resolution and image dimensions

PdfOcr

- Recognition of machine generated text
- Recognition of typewriter scripts and barcodes (1D)
- Image manipulation
- Image pre-processing

1.2.2 Formats

Input formats

- BMP (1, 2, 4, 8, 24 bit)
- GIF (2 to 8 bit)
- HEIC/HEIF
- JBIG2 (lossless compression)
- JPEG, JPEG2000 and JPEG-LS (grayscale, RGB)
- PBM and PNG (1 to 8, 24 bit)
- TIFF
  - Bitonal : uncompressed, CCITT G3, CCITT G3-2D, CCITT G4, LZW, ZIP, Packbits
  - Grayscale, RGB and CMYK: uncompressed, LZW, JPEG, JPEG (old), ZIP, Packbits

Output formats - Image to PDF Converter

- PDF 1.x (PDF 1.0, …, PDF 1.7)
- PDF 2.0
- PDF/A-1a, PDF/A-1b
- PDF/A-2a, PDF/A-2b, PDF/A-2u
- PDF/A-3a, PDF/A-3b, PDF/A-3u

Output formats - Image to Image Converter

- All input formats plus EPS

1.2.3 Conformance

- Standards:
  - ISO 32000-1 (PDF 1.7)
  - ISO 32000-2 (PDF 2.0)
  - ISO 19005-1 (PDF/A-1)
  - ISO 19005-2 (PDF/A-2)
  - ISO 19005-3 (PDF/A-3)
  - TIFF V6
- Quality assurance: Isartor test suite

1.3 Interfaces

The following interfaces are available:

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

1.4 Operating systems

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

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

'+' indicates the minimum supported version.

[1] Limited supported OS versions. Operating systems

1.5 How to best read this manual

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

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

2 Installation and deployment

2.1 Windows

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

To install the software, proceed as follows:

1. You need administrator rights to install this software.
2. Log in to your download account at https://www.pdf-tools.com. Select the product "Image to PDF Converter API". If you have no active downloads available or cannot log in, please contact pdfsales@pdf-tools.com for assistance.
   You can find different versions of the product available. Download the version that is selected by default. You can select a different version.
   The product comes as a ZIP archive containing all files, or as a NuGet package containing all files for development in .NET.
   There is a 32 and a 64-bit version of the product available. While the 32-bit version runs on both 32 and 64-bit platforms, the 64-bit version runs on 64-bit platforms only. The ZIP archive as well as the NuGet package contain both the 32-bit and the 64-bit version of the product.
3. If you are using the ZIP archive, uU nzip the archive to a local folder, e.g. `C:\Program Files\PDF Tools AG\`.
   This creates the following subdirectories (see also ZIP archive):

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

4. The usage of the NuGet package is described in section NuGet package.
5. (Optional) Register your license key using the License management.
6. Identify the interface you are using. Perform the specific installation steps for that interface described in Interface-specific installation steps.
7. Ensure the cache directory exists as described in Special directories.
8. Make sure your platform meets the requirements regarding fonts described in Fonts.
9. (Optional) Download and install the 3-Heights® OCR Enterprise Add-on, and the OCR Engine as described in the respective manuals:
   - 3-Heights® OCR Add-on for ABBYY FineReader Engine v10: `OcrAbbyy10.pdf`
   - 3-Heights® OCR Add-on for ABBYY FineReader Engine v11: `OcrAbbyy11.pdf`
   - 3-Heights® OCR Add-on for ABBYY FineReader Engine v12: `OcrAbbyy12.pdf`
   - 3-Heights® OCR Service: `OcrService.pdf` from the separate product kit.

2.2 Linux and macOS

This section describes installation steps required on Linux or macOS.

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

- Java interface
- Native C interface

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

File description

Name	Description
`bin/x64/libImg2PdfAPI.so`	Shared library that contains the main functionality. The file's extension differs on macOS, (`.dylib` instead of `.so`).
`bin/x64/*.ocr`	OCR plugin modules
`doc/*.*`	Documentation
`include/*.h`	Header files to include in your C/C++ project
`jar/I2PA.jar`	Java API archive
`samples`	Example code

2.2.1 Linux

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

   ```
   ldd libImg2PdfAPI.so
   ```

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

   ```
   ln -s /opt/pdf-tools.com/bin/x64/libImg2PdfAPI.so /usr/lib
   ```

4. Optionally, register your license key using the license manager.
5. Identify the interface you are using. Perform the specific installation steps for that interface described in Interface-specific installation steps.
6. Ensure the cache directory exists as described in Special directories.
7. Make sure your platform meets the requirements regarding fonts described in Fonts.
8. (Optional) Download and install the 3-Heights® OCR Enterprise Add-on, and the OCR Engine as described in the respective manuals:
   - 3-Heights® OCR Add-on for ABBYY FineReader Engine v10: `OcrAbbyy10.pdf`
   - 3-Heights® OCR Add-on for ABBYY FineReader Engine v11: `OcrAbbyy11.pdf`

- 3-Heights® OCR Add-on for ABBYY FineReader Engine v12: OcrAbbyy12.pdf
- 3-Heights® OCR Service: OcrService.pdf from the separate product kit.

2.2.2 macOS

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

```
ln libImg2PdfAPI.dylib libImg2PdfAPI.jnilib
```

2.3 ZIP archive

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

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

2.3.1 Development

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

Req. The file is required for this interface.

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

Doc. The file is for documentation only.

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

Files for development

Name	.NET	Java	COM	C
bin\‹platform›\Img2PdfAPI.dll	Req.	Req.	Req.	Req.
bin*NET.dll	Req.			
bin*NET.xml	Doc.			
bin\‹platform›*.ocr	Opt.	Opt.	Opt.	Opt.
doc*.pdf	Doc.	Doc.	Doc.	Doc.
doc\Img2PdfAPI.idl			Doc.	
doc\javadoc*.*		Doc.		
include\img2pdfapi_c.h				Req.
include*.*				Opt.
jar\I2PA.jar		Req.		
lib\‹platform›\Img2PdfAPI.lib				Req.[2]
samples*.*	Doc.	Doc.	Doc.	Doc.

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

File description

Name	Description
bin\‹platform›\Img2PdfAPI.dll	DLL that contains the main functionality (required), where ‹platform› is either Win32 or x64 for the 23-bit or the 64-bit library, respectively.
bin*NET.dll	.NET assemblies are required when using the .NET interface. The files bin*NET.xml contain the corresponding XML documentation for MS Visual Studio.
bin\‹platform›*.ocr	OCR plugin DLLs that are used in combination with the 3-Heights® OCR Enterprise Add-on, which can be purchased as a separate product[3]
doc*.*	Documentation
include*.*	Files to include in your C / C++ project

[2] Not required for Linux or macOS.

File description

`lib\‹platform›\Img2PdfAPI.lib`	On Windows operating systems, the object file library needs to be linked to the C/C++ project.
`jar\I2PA.jar`	Java API archive
`samples\*.*`	Sample programs in different programming languages

2.3.2 Deployment

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

Files for deployment

Name	.NET	Java	COM	C
`bin\‹platform›\Img2PdfAPI.dll`	Req.	Req.	Req.	Req.
`bin\*NET.dll`	Req.			
`bin\‹platform›\*.ocr`	Opt.	Opt.	Opt.	Opt.
`jar\I2PA.jar`		Req.		

The deployment of an application works as described below:

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

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

1. The developed and compiled application consists of the file `application.exe`. Color profiles are not used.
2. The application uses the COM interface and is distributed on Windows only.
   - The main DLL `Img2PdfAPI.dll` must be distributed.
3. All files are copied to the target location using a batch script. This script contains the following commands:

```
copy application.exe %targetlocation%\.
copy Img2PdfAPI.dll %targetlocation%\.
```

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

```
regsvr32 /s %targetlocation%\Img2PdfAPI.dll.
```

[3] These files must reside in the same directory as Img2PdfAPI.dll.

2.4 NuGet package

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

Installation

The package `PdfTools.Img2Pdf 6.27.6` is available on nuget.org. Right-click on your .NET project in Visual Studio and select "Manage NuGet Packages...". Finally, select the package source "nuget.org" and navigate to the package `PdfTools.Img2Pdf 6.27.6`.

Development

The package `PdfTools.Img2Pdf 6.27.6` contains .NET libraries with versions .NET Standard 1.1, .NET Standard 2.0, and .NET Framework 2.0, and native libraries for Windows, macOS, and Linux.

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

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

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

2.5 Interface-specific installation steps

2.5.1 COM interface

Registration

Before you can use the 3-Heights® Image to PDF Converter API component in your COM application program, you have to register the component using the `regsvr32.exe` program that is provided with the Windows operating system. The following command shows how to register the Img2PdfAPI.dll. In Windows Vista and later, the command needs to be executed from an administrator shell.

```
regsvr32 "C:\Program Files\PDF Tools AG\bin\‹platform›\Img2PdfAPI.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® Image to PDF Converter API, you need to use the `regsvr32` from the directory `%SystemRoot%\SysWOW64` instead of `%SystemRoot%\System32`.[4]

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

Other files

[4] Otherwise, you get the following message: `LoadLibrary("Img2PdfAPI.dll") failed - The specified module could not be found.`

The other DLLs do not need to be registered, but for simplicity, it is suggested that they reside in the same directory as the `Img2PdfAPI.dll`.

2.5.2 Java interface

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

For compilation and execution

When using the Java interface, the Java wrapper `jar\I2PA.jar` needs to be on the CLASSPATH. You can do this by either adding it to the environment variable CLASSPATH, or by specifying it using the switch `-classpath`:

```
javac -classpath ".;C:\Program Files\PDF Tools AG\jar\I2PA.jar" ^
 sampleApplication.java
```

For execution

Additionally, the library `Img2PdfAPI.dll` needs be in one of the system's library directories[5] or added to the Java system property `java.library.path`. You can add the library by either adding it dynamically at program startup before using the API, or by specifying it using the switch `-Djava.library.path` when starting the Java VM. Choose the correct subdirectory (`x64` or `Win32` on Windows) depending on the platform of the Java VM[6].

```
java -classpath ".;C:\Program Files\PDF Tools AG\I2PA.jar" ^
 "-Djava.library.path=C:\Program Files\PDF Tools AG\bin\x64" sampleApplication
```

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

```
... -classpath ".:/path/to/I2PA.jar" ...
```

2.5.3 .NET interface

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

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

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

The .NET assemblies (`*NET.dll`) are added as references to the project; they are needed at compile time. `Img2PdfAPI.dll` is not a .NET assembly, but a native library. It is not added as a reference to the project. Instead, it is loaded during execution of the application.

For the operating system to find and successfully load the native library `Img2PdfAPI.dll`, it must match the executing application's bitness (32-bit versus 64-bit) and it must reside in either of the following directories:

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

[5] On Windows defined by the environment variable PATH, and on Linux defined by LD_LIBRARY_PATH.
[6] If the wrong data model is used, there is an error message similar to this: "`Can't load IA 32-bit .dll on a AMD 64-bit platform`"

In Visual Studio, when using the platforms "x86" or "x64", you can do this by adding the 32-bit or 64-bit `Img2PdfAPI.dll`, respectively, as an "existing item" to the project, and setting its property "Copy to output directory" to true. When using the "AnyCPU" platform, make sure, by some other means, that both the 32-bit and the 64-bit `Img2PdfAPI.dll` are copied to subdirectories `win-x86` and `win-x46` of the output directory, respectively.

2.5.4 C interface

- The header file `img2pdfapi_c.h` needs to be included in the C/C++ program.
- On Windows operating systems, the library `Img2PdfAPI.lib` needs to be linked to the project.
- The dynamic link library `Img2PdfAPI.dll` needs to be in a path of executables (e.g. on the environment variable %PATH%).

2.6 Uninstall, Install a new version

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

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

2.7 Color profiles

In PDF/A, using uncalibrated color spaces (DeviceGray, DeviceRGB, and DeviceCMYK) is prohibited because colors that are specified in this way cannot be reproduced reliably on multiple output devices. Therefore, when converting to PDF/A, a color profile has to be embedded.

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

2.7.1 Default color profiles

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

Windows

1. `%SystemRoot%\System32\spool\drivers\color`
2. directory `Icc`, which must be a direct subdirectory of where the `Img2PdfAPI.dll` resides.

Linux and macOS

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

2.7.2 Set other color profiles

Another color profile may be set using the `SetOutputIntent` or `SetColorSpaceProfile` methods.

2.7.3 Get other color profiles

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

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

2.8 Fonts

Fonts are required, if OCR is preformed and OCR text is added to a PDF document. Therefore, it is crucial, that the fonts available in the Font directories contain all characters required for the OCR text. For example, when recognizing Japanese OCR text, it is recommended to add the fonts “MS Mincho” or “MS Gothic” to the Font directories.

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

2.8.1 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® Image to PDF 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.

2.9 Note about the evaluation license

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

2.10 Special directories

2.10.1 Directory for temporary files

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

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

Windows

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

Linux and macOS

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

2.10.2 Cache directory

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

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

Windows

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

Linux and macOS

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

where `<TempDirectory>` refers to the Directory for temporary files.

2.10.3 Font directories

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

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

Windows

1. `%SystemRoot%\Fonts`
2. User fonts listed in the registry key `\HKEY_CURRENT_USER\Software\Microsoft\Windows NT\CurrentVersion\Fonts`. This includes user specific fonts from `C:\Users\<user>\AppData\Local\Microsoft\Windows\Fonts` and app specific fonts from `C:\Program Files\WindowsApps`
3. `Fonts` directory, which must be a direct subdirectory of where `Img2PdfAPI.dll` resides.

macOS

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

Linux

1. `/usr/share/fonts`
2. `/usr/local/share/fonts`

3. `~/.fonts`
4. `$PDFFONTDIR` or `/usr/lib/X11/fonts/Type1`

3 License management

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

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

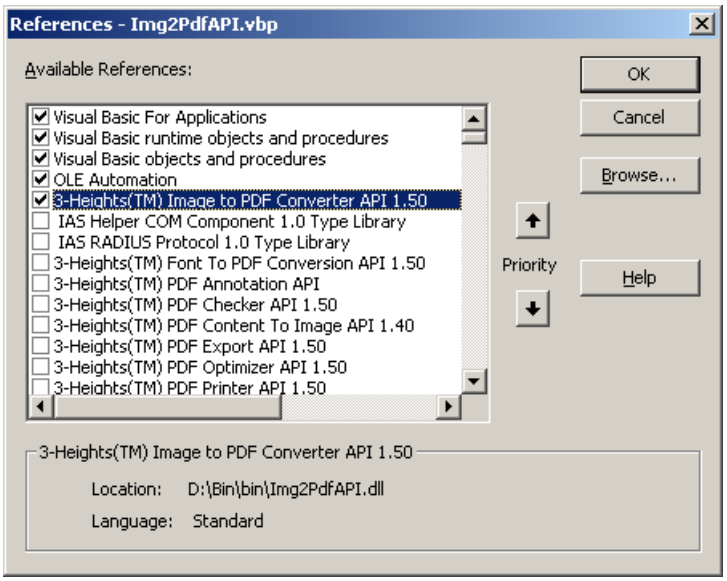
4 Programming interfaces

4.1 Visual Basic 6

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

If you start from scratch, perform these steps:

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



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

```
Private Sub Command1_Click()
  Dim conv As New IMG2PDFAPILib.Img2Pdf
  conv.Create App.Path & "\output.pdf"
  conv.CreatePageFromImage App.Path & "\input.jpg"
  conv.Close
End Sub
```

The four steps of the above code are very simple:

1. Create a Img2Pdf object
2. create a PDF file for output
3. Open an image file for input and copy its page(s)
4. Close PDF- and image file

And that's all - a few lines of code. To modify your program and set options, consult the Reference Manual section.

4.2 .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® Image to PDF Converter API. The easiest to quickly start is to refer to this sample.

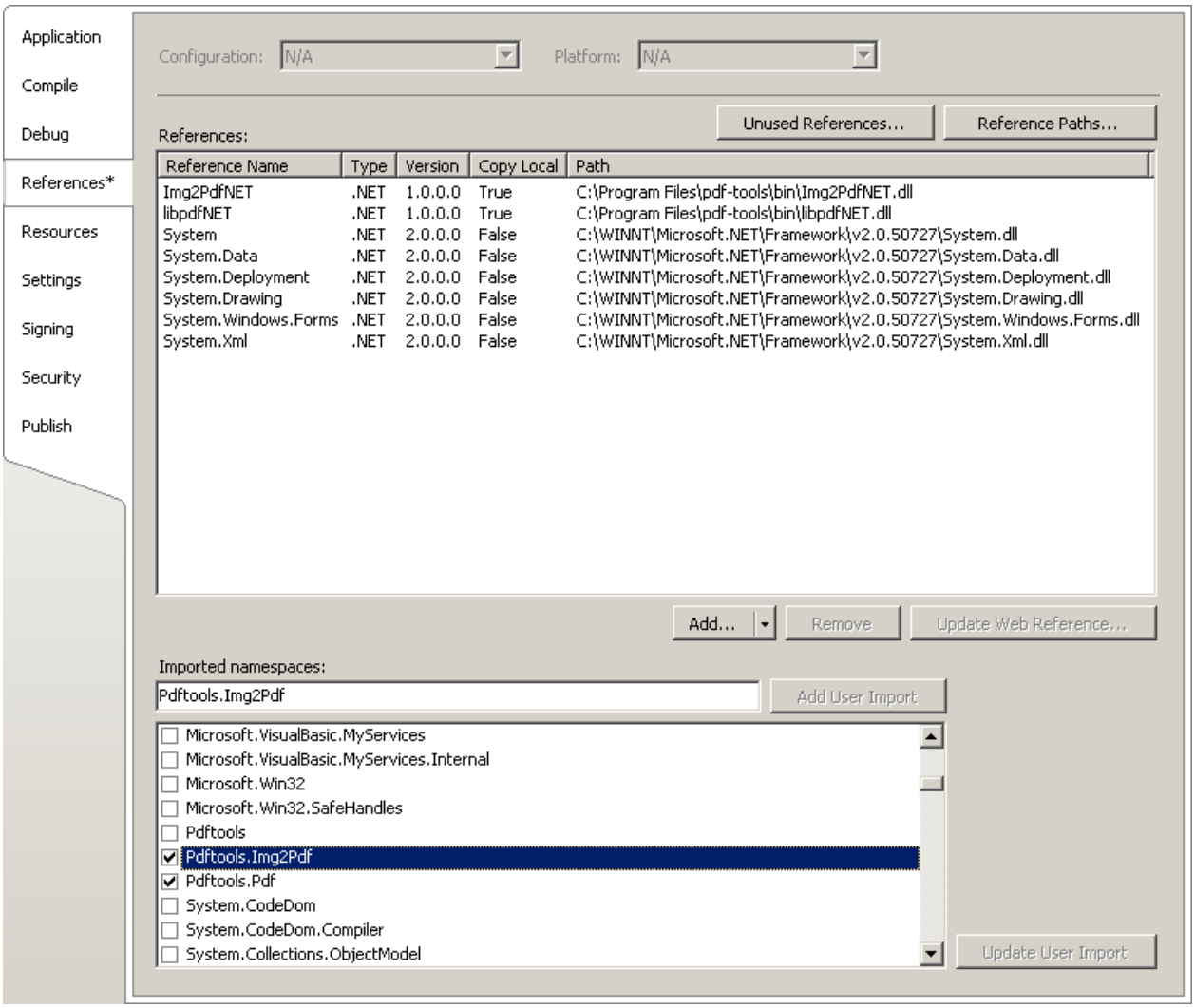
To create a new project from scratch, perform the following steps:

1. Start Visual Studio and create a new C# or VB project.
2. Add references to the NuGet package `PdfTools.Img2Pdf 6.27.6`, as described in NuGet package.
3. Import namespaces (Note: This step is optional, but useful.)
4. Write your code.

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

4.2.1 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`, and `Pdftools.Img2Pdf`.
   You should now have settings similar as in the screenshot below:



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

Example:

```
Dim converter As New Pdftools.Img2Pdf.Img2Pdf()
converter.Create(...)
converter.CreatePagesFromFile(...)
...
converter.Close()
```

4.2.2 C#

3. Add the following namespaces:

Example:

```
using Pdftools.Pdf;
using Pdftools.Img2Pdf;
```

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

Example:

```
using (Img2Pdf converter = new Img2Pdf())
{
    converter.Create(...);
    converter.CreatePagesFromFile(...);
    ...
    converter.Close();
}
```

4.2.3 Deployment

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

1. The project must be compiled using Microsoft Visual Studio. See also .NET interface.
2. For deployment, all items in the project's output directory (e.g. `bin\Release`) must be copied to the target computer. This includes the 3-Heights® Image to PDF Converter API's .NET assemblies (`*NET.dll`), as well as the native library (`Img2PdfAPI.dll`) in its 32 bit or 64 bit version or both. The native library can alternatively be copied to a directory listed in the PATH environment variable, e.g. `%SystemRoot%\System32`.
3. It is crucial that the native library `Img2PdfAPI.dll` is found at execution time, and that the native library's format (32 bit versus 64 bit) matches the operating system.
4. The output directory may contain multiple versions of the native library, e.g. for Windows 32 bit, Windows 64 bit, MacOS 64 bit, and Linux 64 bit. Only the versions that match the target computer's operating system need be deployed.
5. If required by the application, optional DLLs must be copied to the same folder. See Deployment for a list and description of optional DLLs.

4.2.4 Troubleshooting: TypeInitializationException

The most common issue when using the .NET interface is that the correct native DLL `Img2PdfAPI.dll` is not found at execution time. This normally manifests when the constructor is called for the first time and an exception of type `System.TypeInitializationException` is thrown.

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

`System.DllNotFoundException` Unable to load DLL `Img2PdfAPI.dll`: The specified module could not be found.

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

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

Troubleshooting: DllNotFoundException

This means that the native DLL `Img2PdfAPI.dll` could not be found at execution time.

Resolve this by performing one of these actions:

- Use the NuGet package.
- Add `Img2PdfAPI.dll` as an existing item to your project and set its property "Copy to output directory" to "Copy if newer", or
- Add the directory where `Img2PdfAPI.dll` resides to the environment variable `%Path%`, or
- Manually copy `Img2PdfAPI.dll` to the output directory of your project.

Troubleshooting: BadImageFormatException

The exception means that the native DLL `Img2PdfAPI.dll` has the incorrect "bitness" (i.e. platform 32 vs. 64 bit). There are two versions of `Img2PdfAPI.dll` available in the ZIP archive: one is 32-bit (directory `bin\Win32`) and the other 64-bit (directory `bin\x64`). It is crucial that the platform of the native DLL matches the platform of the application's process.

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

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

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

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

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

5 User guide

5.1 Overview of the API

5.1.1 About the 3-Heights® Image to PDF Converter API

The API can be used in any application that requires a process to convert images to PDF documents or split or merge images. Here is a typical use case:

An application takes raster images as input. These can come from any source, such as a scanner or are uploaded from the internet. The application processes these images, e.g. resizes them, applies down-sampling, compresses them, merges them with other images. Finally, it creates an output document. The output document can be an image again or a PDF or a PDF/A document. The output is used for any purpose, such as sending it back to the submitter of the original image, archiving it, or forwarding it for post-processing.

5.1.2 How does the API work?

The way to use the 3-Heights® Image to PDF Converter API is output-oriented. An Img2Pdf object is bound to a PDF output document, which can be a PDF file or a PDF in memory. One or multiple images can be opened and their pages, or a selection of pages, are converted to PDF pages and added to the PDF output document.

This allows for single document conversion as well as merging multiple image documents into one PDF document or split one multi-page image (e.g. a TIFF) to single-page PDF documents.

The basic call sequence is:

- Create object
- Set PDF output document conformance (such as PDF 1.5 or PDF/A 2b)
- Create PDF output document
- Apply settings (page size, quality, color profiles, etc.)
- Create page(s) from image input document(s)
- Close PDF

In Visual Basic 6, these calls could look as below::

```
Dim conv As New IMG2PDFAPILib.Img2Pdf
conv.Compliance = IMG2PDFAPILib.ePDFA1b
If Not conv.Create(outputPDF.txt) Then ...
conv.AdjustPage = 1
If Not conv.CreatePagesFromFile( inputImage.txt, 1, -1) Then ...
conv.Close
```

5.1.3 Using with the PDF Prep Tool Suite

The 3-Heights® Image to PDF Converter API is also bundled to the PDF Prep Tool Suite (PTS) to convert raster images to PDF images, which then can be added to PDF documents.

The PTS does not support .NET. Therefore, any comments in this manual about .NET can be disregarded if working in combination with the PTS. The .NET assemblies are not bundled with the PTS.

5.1.4 OCR recognition of images

The 3-Heights® Image to PDF Converter API can also be used to perform OCR on an image and extract the detected text. During this process, no PDF output document is created. This feature can, for example, be used to read a barcode from an image.

The basic call sequence in Visual Basic 6 is as follows:

- Create a PDFCodec object.
- Open the image file and set the page number.
- Create an ImgOcr object and configure it (OCR engine, parameters, language).
- Set the image using the SetImage method of the ImgOcr object and call the Recognize method to perform OCR recognition.
- Read the OCR text using the GetFirstOcrText and GetNextOcrText methods.

5.2 Error handling

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

Methods communicate their level of success or failure using their return value. The return values to be interpreted as failures are documented in the Interface reference. To identify the error on a programmatic level, check the ErrorCode property. The ErrorMessage property provides a human readable error message, which describes the error.

Example:

```
using (Img2Pdf converter = new Img2Pdf())
{
  if (!converter.Create(txtOutput.Text, String.Empty, String.Empty,
      PDFPermission.ePermNoEncryption))
  {
    MessageBox.Show(String.Format(
      "Error {0}: {1}", converter.ErrorCode, converter.ErrorMessage));
    return;
  }
  [...]
}
```

6 Interface reference

The reference manual is based on the COM interface. However there is an equivalent function to each COM function in the C, .NET and Java interface. (See `img2pdfapi_c.h` and `i2pa.jar`)

The main DLL contains five classes:

`Img2Img` This class can be used to convert images, or a page range of them, from one type to another. This class can also be used to change image dimensions like width, height and resolution.

`Img2Pdf` This class can be used to convert images to PDF documents.

`PDFCodec` This class can be used to retrieve various information from images, such as image compression, color depth, resolution, size, image mask, etc. This class can also be used to interface with other libraries, such as the PDF Prep Tool to import images into a PDF document.

`ImgOcr` This class can be used to perform OCR recognition on an image and extract the detected text.

`OcrText` This class represents a text fragment detected by the ImgOcr class.

`PdfOcr` This class can be used for optical character recognition of images embedded in PDF files.

6.1 `Img2Pdf` Interface

The interface Img2Pdf provides the functionality to create a PDF document from various image formats.

Image-related properties, such as compression or quality are related to the target output file. For example, if `BitonalCompression` is set to `eComprGroup4`, any bi-tonal image that is converted to a PDF document is saved with compression CCITT G4. In order to read the property (e.g. the compression) of an existing image file, use the interface `PDFCodec`.

6.1.1 `AdjustOrientation`

Property (get, set): `Boolean AdjustOrientation`
Default: `False`

When set to `True`, every page of the PDF is oriented in such a way that the longer side length of the input image conforms with the longer side length of the corresponding page.

6.1.2 `AdjustPage`

Property (get, set): `Boolean AdjustPage`
Default: `True`

When set to `True`, the page dimensions of the PDF will be chosen, so that the image fits exactly on the page. If set to `True`, the properties `CenterImage` and `FitImage` are automatically set to `False`.

6.1.3 Alt

Property (get, set): `String Alt`
Default: `Imported image`

In order to create a document that conforms to PDF/A level A (PDF/A-1a, PDF/A-2a, PDF/A-3a), each image must have an alternate text with a description of the image in support of accessibility to users with disabilities. This property sets this alternate text used for images added subsequently. The property should be set before adding images. It is only relevant in combination with PDF/A level A. See also properties Lang and Compliance.

6.1.4 BitonalCompression

Property (get, set): `TPDFCompression BitonalCompression`
Default: `eComprGroup4`

Get or set the compression type for bitonal images. Normally, either CCITT G4 or JBIG2 is used for bitonal compression. Due to the simpler algorithm, CCITT G4 has the advantage of being faster. JBIG2 can achieve compression ratios that are up to twice as high as CCITT G4 at the cost of longer computation time. See also TPDFCompression.

6.1.5 BitsPerPixel

Property (get, set): `Integer BitsPerPixel`
Default: `-1 (no effect)`

Get or set the color depth. Available: Bi-tonal: 1. When using 1 bit per pixel, it is suggested to set a suitable dithering algorithm (see property Dithering).

6.1.6 BorderSize

Property (get, set): `Single BorderSize`
Default: `0`

This property sets or gets the border between the image and the page border. The units are points (1 point = 1/72 inch). The border does not change the dimension of the page set by the method SetPageSize.

6.1.7 CenterImage

Property (get, set): `Boolean CenterImage`
Default: `False`

Center the image on the page horizontally and vertically. If set to `True`, the property AdjustPage is automatically set to `False`.

6.1.8 Close

Method: `Boolean Close()`

This method closes the PDF file. It is called after a PDF document has been created and the desired pages from images are added. Avoiding the call to this function may still result in a valid output, but it can also cause memory leaks.

Returns:

`True` The PDF file was closed successfully.

`False` Otherwise.

6.1.9 Compliance

Property (get, set): `TPDFCompliance Compliance`
Default: `ePDF17`

This property allows setting a PDF conformance level. It must be set before calling Create. Supported conformance modes are:

`ePDF1x` Regular PDF Versions such as 1.4, 1.5, 1.6, 1.7

`ePDF20` Regular PDF Version 2.0

`ePDFA1b` PDF/A-1b format

`ePDFA1a` PDF/A 1a format (accessibility)

`ePDFA2b` PDF/A 2b format

`ePDFA2u` PDF/A 2u format (Unicode)

`ePDFA2a` PDF/A 2a format (accessibility)

`ePDFA3b` PDF/A 3b format

`ePDFA3u` PDF/A 3u format (Unicode)

`ePDFA3a` PDF/A 3a format (accessibility)

In order to create PDF/A compatible documents, there are additional requirements besides setting the conformance level:

Metadata Selecting a PDF/A conformance level will automatically generate the XML metadata and other requirements to meet the PDF/A specification.

Tagging For PDF/A level A (accessibility) it is also requested to have an alternate descriptive text for images. This text can be set using the properties `Alt` and `Lang`.

Color profiles For non-calibrated colors, a color profile must be embedded. See methods `SetOutputIntent` and `SetColorSpaceProfile`. If no color profile is set, then for RGB and grayscale colors, calibrated color spaces are generated while for CMYK colors, a default CMYK output intent is set.

If JPEG2000 images are to be converted to PDF/A and the JPEG2000 compression shall be retained, a conformance level of PDF/A-2 or later must be selected.

6.1.10 ContinuousCompression

Property (get, set): `TPDFCompression ContinuousCompression`
Default: `eComprJPEG`

Get or set the compression type of color and grayscaled images in the PDF document. See also enumeration `TPDFCompression`.

6.1.11 Create

Method: `Boolean Create(String PDFFileName, String UserPwd, String OwnerPwd, TPDFPermission PermissionFlags)`

> **Note:** In order to meet PDF/A conformance, the document mustn't be encrypted.

Parameters:

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

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

`PermissionFlags` [`TPDFPermission`] (optional) Set the permission flags of the PDF document. This option requires an owner password to be set. By default no permissions are granted. To not encrypt the output document, set PermissionFlags to `-1`, user and owner password to empty string. In order to allow high quality printing, both flags `ePermPrint` and `ePermDigitalPrint` need to be set. See also enumeration `TPDFPermission`. To combine multiple flags, use a bitwise or operator. (For example in Visual Basic: `PermissionFlags = ePermPrint OR ePermDigitalPrint`).

Returns:

`True` The file was created successfully.

`False` The file could not be created, because e.g. the file already exists and is locked/read-only.

6.1.12 CreateInMemory

Method: `Boolean CreateInMemory()`

This method creates a PDF in memory. Once the document is completed and after the `Close` call, it can be accessed using the method `GetPdf`.

6.1.13 CreatePageFromCodec

Method: `Boolean CreatePageFromCodec(PDFCodec pCodec)`

This method creates a page from an image object. It must be called after `Create` or `CreateInMemory`.

Parameter:

`pCodec` [`PDFCodec`] A PDFCodec object holding an image.

Returns:

`True` The page in the PDF document was created successfully.

`False` Otherwise.

6.1.14 CreatePageFromImageFile

Method: `Boolean CreatePageFromImageFile(String FileName)`

This method adds the page (or pages for multi-page TIFF images) of an image file to the current PDF output. It must be called after `Create` or `CreateInMemory`.

Parameter:

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

Returns:

`True` The page(s) in the PDF document were created successfully.

`False` Otherwise.

6.1.15 CreatePageFromFile

Method: `Boolean CreatePageFromFile(String FileName, Long FromPageNo, Long ToPageNo)`

This method adds the page (or a page range for multi-page TIFF images) of an image file to the current PDF output. It must be called after `Create` or `CreateInMemory`.

Parameters:

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

`FromPageNo` [`Long`] (optional) The starting page number. The default is `1`.

`ToPageNo` [`Long`] (optional) The last page number. The default is `-1` (last page).

Returns:

`True` The page(s) in the PDF document were created successfully.

`False` Otherwise.

6.1.16 DefaultDPI

Property (get, set): `Single DefaultDPI`
Default: `96`

Set the default resolution in DPI (dots per inch), if it is not provided by the image. Default is `96` DPI. If the resolution is given by the image then this option does not have any effect. Basically, it changes the amount of dots per inch by changing the size of the image in the PDF document. The size of the raster image in pixel is not changed.

6.1.17 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.

6.1.18 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 Image to PDF Converter API has returned a value, which signales a failure of the function (see Error handling). See also enumeration `TPDFErrorCode`. Pdftools error codes are listed in the header file `bseerror.h`. Please note that only few of them are relevant for the 3-Heights® Image to PDF Converter API.

6.1.19 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 Image to PDF Converter API has returned a value, which signals a failure of the function (see Error handling)

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

.

6.1.20 ExportText

Method: `Boolean ExportText(String FileName)`

This function is used in combination with OCR only. It allows to write the text, which is detected by the OCR engine during conversion, not only as invisible text in the PDF, but additionally to a text file. The text file is closed when output PDF document is closed using the method `Close`.

Parameter:

`FileName` `[String]` Defines the text file and optionally its path. If the parameter is an empty string, no text file is created.

Returns:

`True` If the function call was successful.

`False` Otherwise.

6.1.21 FitImage

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

Scale the image to fit the size of the page of the PDF. The image is scaled so that either width or height match the dimensions of the page, the other dimension is scaled proportionally. If set to True, the property AdjustPage is automatically set to False.

6.1.22 GetOCREngine

[Deprecated] Method: GetOCREngine()

Use GetOCRPluginName instead.

6.1.23 GetOCREngineCount

Method: GetOCREngineCount()

Use GetOCRPluginCount instead.

6.1.24 GetOCRPluginCount

Method: Integer GetOCRPluginCount()

OCR engines are accessed through the corresponding OCR interface DLLs. At present, the following OCR engines are supported:

Abbyy FineReader 11 OCR engine This engine is accessed by the OCR interface DLL pdfocrpluginAbbyy11.ocr.

Abbyy FineReader 10 OCR engine This engine is accessed by the OCR interface DLL pdfocrpluginAbbyy10.ocr.

3-Heights® OCR service This service is accessed by the OCR interface DLL pdfocrpluginService.ocr. The service accesses the Abbyy FineReader 10 or 11 OCR Engine.

The OCR interface DLL is provided by the 3-Heights® Image to PDF Converter API.

The OCR engine is provided as a separate product: 3-Heights® OCR Enterprise Add-on.

To make use of the OCR engine, the OCR interface DLL and the OCR engine must be installed. The GetOCRPluginCount property returns the number of available OCR interface DLLs. It does not verify the corresponding OCR engines are installed and can be initialized. The OCR engine is loaded with the SetOCREngine method.

Returns:

The number of available OCR engines (i.e. their corresponding OCR interface DLLs).

6.1.25 GetOCRPluginName

Method: `String GetOCRPluginName(Integer iOCREngine)`

An OCR engine is accessed through an OCR plugin. Each plugin corresponds to one OCR engine. The number of OCR plugins is retrieved using `GetOCRPluginCount`. The method call `GetOCRPluginName(n)` returns the name of the nth OCR Engine which corresponds to that OCR plugin. At present, there are three OCR engines available: `"abbyy11"`, `"abbyy10"`, and `"service"`.

Parameter:

`iOCREngine` `[Integer]` The number of the OCR engine. The total number of engines is retrieved using `GetOCRPluginCount`.

Returns:

The name of the nth OCR engine. `Nothing` if it does not exist.

6.1.26 GetPdf

Method: `Variant GetPdf()`

Get the output file from memory. See also method `CreateInMemory`.

Returns:

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

6.1.27 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.

6.1.28 IndexedCompression

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

Get or set the compression type of indexed images in the PDF document. Supported compressions are Flate and LZW, see also enumeration TPDFCompression.

6.1.29 InfoEntry

Method: String InfoEntry(String Key)

Retrieve or add a key-value pair to the document info dictionary. Values of predefined keys are also stored in the XMP metadata package.

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

Parameter:

Key [String] A key as string.

Returns:

The value as string.

Examples in Visual Basic 6:

Get the document title.

```
t = doc.InfoEntry("Title")
```

Set the document title.

```
doc.InfoEntry("Title") = "My Title"
```

Set the creation date to 13:55:33, April 5, 2010, UTC+2.

```
doc.InfoEntry("CreationDate") = "D:20100405135533 + 02'00'"
```

6.1.30 Lang

Property (get, set): String Lang
Default: "US-EN"

Set the language for the alternate text that is set using the property Alt. The default language is "US-EN". Other languages can be set using the corresponding abbreviations, e.g. "DE" (German), "FR" (French), etc.

6.1.31 LicenseIsValid

Property (get): Boolean LicenseIsValid

Check if the license is valid.

6.1.32 Linearize

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

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

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

The 3-Heights® Image to PDF Converter API does not support linearization of PDF 2.0 documents. For such documents, processing fails.

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

- When a document is opened in a PDF viewer of a web browser, the first page can be viewed without downloading the entire PDF file. In contrast, a non-linearized PDF file must be downloaded completely before the first page can be displayed.
- When another page is requested by the user, that page is displayed as quickly as possible and incrementally as data arrives, without downloading the entire PDF file.

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

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

6.1.33 OCREmbedOCRImage

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

This option set to True currently requires the OCRDeskewImage to be also set to True.

The OCR engine deskews and de-noises the input image before recognizing the characters. This option controls whether the 3-Heights® Image to PDF Converter API should use the pre-processed image or keep the original image.

Setting this option to True has only an effect if the pre-processed image is provided by the OCR engine, which depends on the type and settings of the engine.

If this option is set to True, the resulting image may have a different color space, compression and size.

Since this option currently requires OCRDeskewImage, it is recommended only for simple scanned documents.

6.1.34 OCRBitonalRecognition

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

Specify whether the images should be converted to bitonal (black and white) before OCR recognition.

Enabling this feature can improve the memory consumption of the OCR process.

Enabling this feature automatically re-embeds the original images in the output document. The setting of the property OCREmbedOCRImage is therefore ignored.

6.1.35 OCRDeskewImage

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

Correct the skew angle of images.

This option set to True has only an effect if the required information is provided by the OCR engine, which depends on the type and settings of the engine.

This option set to True may change the appearance of the page and is only recommended for simple scanned documents that consist of a single image.

Using the option for digital-born documents may destroy the page layout.

6.1.36 OCREmbedBarcodes

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

This property specifies whether the recognized barcodes are embedded in the XMP metadata.

6.1.37 OCRResolutionDPI

Property (get, set): Single OCRResolutionDPI
Default: 300

Re-sample images to target resolution before they are sent to the OCR engine. The default is 300 DPI, which is the preferred resolution for most OCR engines.

6.1.38 OCRThresholdDPI

Property (get, set): `Single OCRThresholdDPI`
Default: `400`

Only images with a higher resolution than the threshold are re-sampled before OCR. The default is `400` DPI. If set to `-1`, no re-sampling is applied.

6.1.39 Orientation

Property (get): `TPDFOrientation Orientation`

Return the orientation rounded to the next 90°. The orientation is an enumeration with eight different values (rotation times flipping). See `TPDFOrientation`.

6.1.40 Quality

[Deprecated] Property (get, set): `Single Quality`

Deprecated, use property `ImageQuality` instead.

6.1.41 ProductVersion

Property (get): `String ProductVersion`

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

6.1.42 Recompress

Property (get, set): `Boolean Recompress`
Default: `False`

If set to `True`, JPEG, JPEG2000 and CCITT Fax Group4 streams are re-compressed.

Advantages:

- Invalid streams are repaired (as far as possible)
- Standard JPEG streams are created (which should be readable by any application)

Disadvantages:

- Recompressing a lossy stream usually increases the file size and lowers the Quality

6.1.43 ResolutionDPI

Property (get, set): `Single ResolutionDPI`
Default: `150`

Get or set the resolution in DPI (dots per inch) after re-sampling images.

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

6.1.44 SetColorSpaceProfile

Method: `Boolean SetColorSpaceProfile(String Profile)`

Set a color space profile for embedding in the output PDF. See also SetOutputIntent for color profiles. The color profile provided here is used directly for the image's color space.

Parameter:

`Profile` [String] The file name of the color profile.

Returns:

`True` The color profile was set successfully.

False The file name points to an invalid color profile. (Only PDF/A conforming profiles are accepted.)

At maximum three profiles (one RGB profile, one CMYK profile, and one Gray profile) can be set by using at most one call to `SetOutputIntent` and/or at most three calls to `SetColorSpaceProfile`.

6.1.45 SetLicenseKey

Method: `Boolean SetLicenseKey(String LicenseKey)`

Sets the license key.

6.1.46 SetMetadata

Method: `Boolean SetMetadata(String FileName)`

Sets the document's XMP metadata. The XMP metadata is inserted as is, which means it is not parsed and validated. If no XMP metadata is provided, the 3-Heights® Image to PDF Converter API generates it automatically.

Parameter:

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

Returns:

`True` The XMP metadata file was set successfully.

`False` Otherwise.

6.1.47 SetMetadataStream

Method: `Boolean SetMetadataStream(Stream Stream)`

Set the document's XMP metadata. The XMP metadata is inserted as is, which means it is not parsed and validated. If no XMP metadata is provided, the 3-Heights® Image to PDF Converter API generates it automatically.

Parameter:

`Stream` [`Stream`] Stream containing XMP metadata.

Returns:

`True` The XMP metadata stream was set successfully.

`False` Otherwise.

6.1.48 SetOCREngine

Method: `Boolean SetOCREngine(String Engine)`

This method requires the 3-Heights® OCR Add-on, which is a separate product, to be installed. See also documentation for the 3-Heights® OCR Add-on.

Set the OCR engine that is used when OCR information shall be added during the conversion. If the engine's name is set to an empty string, OCR is not applied.

Parameter:

`Engine` [`String`] The name of the OCR engine (e.g. `"abbyy11"`). For every available OCR engine, there is a corresponding OCR interface DLL. The OCR interface DLLs (e.g. `pdfocrAbbyy11.ocr`) are distributed with the 3-Heights® Image to PDF Converter API and are required to communicate with the OCR engine. The names of all available OCR engines can be retrieved using the properties `GetOCRPluginCount` and `GetOCRPluginName`.

Returns:

`True` The OCR interface DLL was found, the OCR engine was found and the OCR engine was successfully initialized.

`False` Otherwise.

6.1.49 SetOCRLanguages

Method: `Boolean SetOCRLanguages(String Languages)`

This method requires the 3-Heights® OCR Add-on, which is a separate product, to be installed. See also documentation for the 3-Heights® OCR Add-on.

Setting languages helps the OCR engine to minimize errors by means of using dictionaries of the defined languages.

This method must be called after `SetOCREngine`.

If `SetOCRParams` is used, `SetOCRLanguages` must be called after `SetOCRParams`.

Parameter:

`Languages` `[String]` A string of one or multiple, comma-separated languages. The supported names depend on the OCR engine. The OCR engine will only use dictionaries of the set languages.

Returns:

`True` The language(s) were successfully set.

`False` Otherwise.

Example:

```
SetOCREngine("abbyy11")
SetOCRLanguages("English, German")
```

6.1.50 SetOCRParams

Method: `Boolean SetOCRParams(String Params)`

This method requires the 3-Heights® OCR Add-on, which is a separate product, to be installed. See also documentation for the 3-Heights® OCR Add-on.

By means of this method, OCR engine specific settings can be applied in the form of key-value pairs. These pairs depend on the OCR engine and are described in the corresponding manual.

Parameter:

`Params` `[String]` A list of comma-separated key value pairs. See example.

Returns:

`True` The OCR parameters were successfully set.

`False` Otherwise.

Example: Set a predefined profile for ABBYY 11.

```
SetOCREngine("abbyy11")
SetOCRParams("PredefinedProfile = DocumentArchiving_Accuracy")
```

6.1.51 SetOutputIntent

Method: `Boolean SetOutputIntent(String Profile)`

The output intent represents the output color profile. Setting the output intent is generally only recommended for images intended for a particular CMYK output device. In other cases, the `SetColorSpaceProfile` method should be used.

Color profiles are usually provided with the operating system. On Windows, for example, they can be found at `C:\Windows\System32\spool\drivers\color`. Alternatively, profiles can be found here:

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

Most color profiles are copyrighted. Therefore, you should read the license agreements on the above links before using the color profiles.

Parameter:

`Profile` [`String`] The name of the color profile. An example could be:

`C:\Windows\System32\spool\drivers\color\USWebCoatedSWOP.icc`

If PDF/A conformance is selected and no output intent is defined, then CMYK USWebCoatedSWOP.icc is embedded as default output intent.

This method must be called after `Create` has been called.

6.1.52 SetPageSize

Method: `Boolean SetPageSize(Single Width, Single Height)`

Set the page size of the current and following pages in the PDF document in points. (1 point = 1/72 inch.) Sets property `AdjustPage` to `False`.

Parameters:

`Width` [`Single`] The width of the page in points.

`Height` `[Single]` The height of the page in points.

Returns:

`True` The page size was set successfully.

`False` Otherwise.

The default values, if the property `AdjustPage` is set `False`, are Width=`595` and Height=`842` (A4).

6.1.53 ThresholdDPI

Property (get, set): `Single ThresholdDPI`
Default: `255`

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

The value `-1` deactivates re-sampling.

A typical threshold value when optimizing for the web is `225` DPI.

6.2 PDFCodec Interface

The codec interface provides information about the image. Such as bits per component, components per pixel, color space, the image data itself, etc. This data can be used by other applications such as the PDF Prep Tool Suite.

Keep in mind that most properties are not read before a page number is defined using the `PageNo` property. This is also true for images with just one page.

6.2.1 BitsPerComponent

Property (get): `Integer BitsPerComponent`

Return the number of bits that are used to represent a single color component of an image sample. The number of color components per image data sample can be retrieved through the image's color space interface.

6.2.2 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.

6.2.3 ColorSpace

Property (get): `TPDFColorSpace ColorSpace`

This property returns the color space. See also enumeration TPDFColorSpace.

6.2.4 ComponentsPerPixel

Property (get): `Integer ComponentsPerPixel`

Return the number of components per pixel.

6.2.5 Compression

Property (get): `TPDFCompression Compression`

This property returns the compression type. See also enumeration TPDFCompression and the property Recompress. This property is initially set to eComprRaw.

6.2.6 Create

Method: `Boolean Create(String FileName)`

Create an empty 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 of the image file. Supported extensions are listed in the chapter Supported image extensions.

Returns:

True The file was created successfully.

False The file was not created, e.g. the file already exists and is read-only.

6.2.7 CreateInMemory

Method: `Boolean CreateInMemory(String Extension)`

Create an image in memory.

Parameter:

`Extension` `[String]` The type of the image to be created. Supported extensions are listed in the chapter Supported image extensions.

Returns:

`True` The image was created successfully in memory.

`False` Otherwise.

6.2.8 Decode

Property (get): `Boolean Decode`

Indicates whether the samples of the compressed stream need to be decoded (inverted).

6.2.9 DefaultDPI

Property (get, set): `Single DefaultDPI`
Default: `96.0`

Set the default resolution in dots per inch (DPI). It is only effective in case where the input image has no resolution stated.

6.2.10 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 Image to PDF Converter API has returned a value, which signales a failure of the function (see Error handling). See also enumeration `TPDFErrorCode`. Pdftools error codes are listed in the header file `bseerror.h`. Please note that only few of them are relevant for the 3-Heights® Image to PDF Converter API.

6.2.11 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 Image to PDF Converter API has returned a value, which signals a failure of the function (see Error handling)

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

.

6.2.12 fXDPI, fYDPI

[Deprecated] Property (get, set): `Single fXDPI`

[Deprecated] Property (get, set): `Single fYDPI`

Use the properties XDPI, YDPI instead.

6.2.13 GetImage

Method: `Variant GetImage()`

This method returns an image which was previously created in memory using the methods `CreateInMemory` and `Close`.

6.2.14 Height

Property (get): `Long Height`

Get the height of the image in pixels (also called samples). The unit of pixels can be converted to a distance unit such as inch, millimeter etc. using a resolution value, i.e. 72 DPI (dots per inch).

6.2.15 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.

6.2.16 IsPremultipliedAlpha

Property (get): `Boolean IsPremultipliedAlpha`

This property returns `True` if the image pixels are stored as the original pixel times the alpha value. (I.e. pixel = backdrop * (alpha - 1) + image * alpha)

6.2.17 Mask

Property (get): `Variant Mask`

Return the image's explicit mask as byte array if available. The mask's sample data is organized the same way as the image data except that the data contains one bit per pixel. A one bit indicates an opaque pixel and a zero bit indicates a transparent pixel.

6.2.18 Name

Property (get): `String Name`

The name of the codec, e.g. `"GIF"`, `"JPEG"`, or `"PNG"`.

6.2.19 Open

Method: `Boolean Open(String FileName)`

This method opens an 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 of the image file.

Returns:

`True` The file was opened successfully.

`False` Otherwise.

6.2.20 OpenMem

Method: `Boolean OpenMem(Variant varMem)`

This method opens an image from memory.

Parameter:

`varMem` [`Variant`] A byte array containing the image.

Returns:

`True` The file was opened successfully.

`False` Otherwise.

6.2.21 Page

[Deprecated] Property (get, set): `Long Page`

Use PageNo instead.

6.2.22 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`.

6.2.23 PageNo

Property (get, set): `Long PageNo`
Default: `1`

Set or get the current page number in the image. The page number must always be set, also for single-page images.

The numbers are counted starting from `1` for the first page to the value of PageCount for the last page.

6.2.24 Palette

Property (get): `Variant Palette`

This property returns the palette of the image (if existing).

6.2.25 Quality

[Deprecated] Property (get, set): `Single Quality`

Use `ImageQuality` instead.

6.2.26 Recompress

Property (get, set): `Boolean Recompress`
Default: `True`

If set to `False`, JPEG, JPEG2000 and CCITT Fax Group4 streams are not de-compressed. As a result, the Samples property will return the compressed stream as indicated by the `Compression` property. If possible, the Recompress property should be set before calling the `Open` method, because for some image formats changing the Recompress property might result in reloading some image data.

6.2.27 Samples

Property (get): `Variant Samples`

Return the image's data samples in a byte array. The sample data is ordered by line from top to bottom and within a line from left to right. The lines are byte- aligned. If the number of bits per component is less than one byte, then the samples are ordered beginning with the most significant bit first.

If Recompress is set to `False`, Samples returns a stream compressed with the algorithm indicated by the `Compression` property.

6.2.28 SMask

Property (get, set): `Variant SMask`

With this property the soft mask of an image can be extracted. If the image has a soft mask then this property provides a byte array of the the soft mask data. An image soft mask is a monochrome image of the same resolution as the associated image. Sample values in the soft mask designate the alpha value of the image. A value of 0 ("black") designates a "fully transparent" place and a value of 255 ("white") "fully opaque".

6.2.29 Width

Property (get): `Long Width`

Return the width of the image in pixels (also called samples). The unit of pixels can be converted to a distance unit such as inch, millimeter etc. using a resolution value, i.e. 72 DPI (dots per inch).

6.2.30 XDPI, YDPI

Property (get): `Single XDPI`

Property (get): `Single YDPI`

These properties return the resolution in dots per inch in X and Y direction.

6.3 Img2Img Interface

The image to image interface is a separate interface that provides functionality to convert images from one format to another. It allows changing the compression type and allows setting the width, the height and the resolution. Re-sampling is supported.

6.3.1 AllowResampling

Property (get, set): `Boolean AllowResampling`
Default: `false`

Forces re-sampling in situations where only the resolution (see DPI) or only the true_width/true_height (see TrueWidth, TrueHeight) is set. For more information about the behavior see Specification of resolution and image dimensions.

6.3.2 BitonalCompression

Property (get, set): `TPDFCompression BitonalCompression`
Default: `eComprGroup4`

Get or set the compression type for bitonal images. Normally, either CCITT G4 or JBIG2 is used for bitonal compression. Due to the simpler algorithm, CCITT G4 has the advantage of being faster. JBIG2 can achieve compression ratios that are up to twice as high as CCITT G4 at the cost of longer computation time. See also TPDFCompression.

6.3.3 ContinuousCompression

Property (set): `TPDFCompression ContinuousCompression`

Set the compression type for color and grayscale images in the output image. See also enumeration TPDFCompression.

6.3.4 ContinousCompression

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

Use ContinuousCompression instead.

6.3.5 ConvertFile

Method: Boolean ConvertFile(String InputFileName, String OutputFileName, Long FromPageNo, Long ToPageNo)

Convert an image from one type to another and save it to a file. The image type is defined by the extension of the parameter OutputFileName. See also chapter Supported image extensions.

Parameters:

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

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

FromPageNo [Long] (optional) The first page of the page range to be copied from a multi-page input file. The default is 1.

ToPageNo [Long] (optional) The last page of the page range to be copied from a multi-page input file. The default is -1 (last page).

Returns:

True The file of the image was created successfully.

False Otherwise.

6.3.6 CopyPage

Method: Boolean CopyPage(PDFCodec InputCodec, PDFCodec OutputCodec)

This method copies the current page (PDFCodec.Page) from a PDFCodec object to another PDFCodec object. Target codec has to be an opened file using Open or Create.

Parameters:

InputCodec [PDFCodec] A PDFCodec object containing a valid image at the currently set page number of the input codec.

OutputCodec [PDFCodec] A PDFCodec object, to which the page is appended. The currently set page number in the output codec is not relevant.

Returns:

`True` The page was copied successfully.

`False` Otherwise.

6.3.7 DPI

Property (get, set): `Single DPI`
Default: `0 (not applied)`

Set the resolution of the output image in dots per inch (DPI). The width and the height remain constant. If furthermore option `AllowResampling` is set, true_width and true_height are constant, whereas the width and the height are changed. This results in a re-sampling of the image. For more information about the behavior see Specification of resolution and image dimensions.

6.3.8 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 Image to PDF Converter API has returned a value, which signales a failure of the function (see Error handling). See also enumeration `TPDFErrorCode`. Pdftools error codes are listed in the header file `bseerror.h`. Please note that only few of them are relevant for the 3-Heights® Image to PDF Converter API.

6.3.9 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 Image to PDF Converter API has returned a value, which signals a failure of the function (see Error handling)

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

.

6.3.10 Height

Property (get, set): `Long Height`
Default: `0 (not applied)`

Set the height in pixel of the output image. The width is calculated respecting proportions. If the width is set too (see `Width`), the height is omitted. For more information about the behavior see Specification of resolution and image dimensions.

6.3.11 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.

6.3.12 IndexedCompression

Property (set): `TPFDCompression IndexedCompression`

Set the compression type for indexed images in the output image. See also enumeration `TPDFCompression`.

6.3.13 TrueHeight

Property (get, set): `Long TrueHeight`
Default: `0 (not applied)`

Set the true_height in mm of output image. The true_width is calculated respecting proportions. If the true_width is set too (see `TrueWidth`), the true_height is omitted. For more information about the behavior see Specification of resolution and image dimensions.

6.3.14 TrueWidth

Property (get, set): `Long TrueWidth`
Default: `0 (not applied)`

Set the true_width in mm of output image. The true_height is calculated respecting proportions. If the true_height is set too (see `TrueHeight`), the true_height is omitted. For more information about the behavior see Specification of resolution and image dimensions.

6.3.15 Quality

[Deprecated] Property (get, set): `Single Quality`

Deprecated, use `ImageQuality` instead.

6.3.16 Width

Property (get, set): `Long Width`
Default: `0 (not applied)`

Set the width in pixel of the output image. The height is calculated respecting proportions. If the height is set too (see `Height`), the height is omitted. For more information about the behavior see Specification of resolution and image dimensions.

6.4 ImgOcr Interface

The image OCR interface allows you to extract OCR text from an image opened using the `PDFCodec` interface. During that process, no output file is created. The `ImgOcr` interface is not needed to create a searchable PDF, use the `Img2Pdf` interface for that task.

6.4.1 GetFirstOcrText

Method: `OcrText GetFirstOcrText()`

Get the first text fragment recognized, or `Nothing` if none available.

6.4.2 GetNextOcrText

Method: `OcrText GetNextOcrText()`

Get the next text fragment recognized, or `Nothing` if none available.

6.4.3 GetOCRPluginCount

Method: `Integer GetOCRPluginCount()`

Get the number of available OCR plugins (see GetOCRPluginCount of the `Img2Pdf` interface).

6.4.4 GetOCRPluginName

Method: `String GetOCRPluginName(Integer iOCREngine)`

Get the name of the i-th OCR plugin engine.

6.4.5 Recognize

Method: `Boolean Recognize()`

Perform OCR recognition. The return value indicates whether or not the recognition has been successful.

6.4.6 SetOCREngineName

Method: `Boolean SetOCREngineName(String Engine)`

Set the OCR engine.

6.4.7 SetImage

Method: `Boolean SetImage(PDFCodec Image)`

Set the image to OCR. Before calling this method the image must be opened and the correct page set. The return value indicates whether or not the image could be set.

6.4.8 SetOCRLanguages

Method: `Boolean SetOCRLanguages(String Languages)`

Set the OCR languages (see SetOCRLanguages of the `Img2Pdf` interface).

6.4.9 SetOCRParams

Method: `Boolean SetOCRParams(String Parameters)`

Set the OCR parameters (see SetOCRParams of the `Img2Pdf` interface).

6.5 OcrText Interface

The OCR text interface represents a text fragment detected by the image OCR interface.

6.5.1 BaseLine

Property (get): `Single BaseLine`

Get the Y coordinate of the text's base line.

6.5.2 FontName

Property (get): `String FontName`

Get the name of the font. For barcodes the font name is `"Barcode"`.

6.5.3 FontSize

Property (get): `Single FontSize`

Get the size of the font in points.

6.5.4 Rect

Property (get): `Variant Rect`

Get the bounding box rectangle of the text.

6.5.5 StringLength

Property (get): `Integer StringLength`

Get the number of characters of the recognized string.

6.5.6 Text

Property (get): `String Text`

Get the recognized text.

6.6 PdfOcr Interface

The interface PdfOcr provides the functionality for optical character recognition of PDF files and embedding of the recognized text in the PDF.

6.6.1 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.

6.6.2 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 Image to PDF Converter API has returned a value, which signales a failure of the function (see Error handling). See also enumeration TPDFErrorCode. Pdftools error codes are listed in the header file bseerror.h. Please note that only few of them are relevant for the 3-Heights® Image to PDF Converter API.

6.6.3 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 Image to PDF Converter API has returned a value, which signals a failure of the function (see Error handling)

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

.

6.6.4 GetOCRPluginCount

Method: Integer GetOCRPluginCount()

OCR engines are accessed through the corresponding OCR interface DLLs. At present, the following OCR engines are supported:

Abbyy FineReader 11 OCR engine This engine is accessed by the OCR interface DLL `pdfocrpluginAbbyy11.ocr`.

Abbyy FineReader 10 OCR engine This engine is accessed by the OCR interface DLL `pdfocrpluginAbbyy10.ocr`.

3-Heights® OCR service This service is accessed by the OCR interface DLL `pdfocrpluginService.ocr`. The service accesses the Abbyy FineReader 10 or 11 OCR Engine.

The OCR interface DLL is provided by the 3-Heights® Image to PDF Converter API.

The OCR engine is provided as a separate product: 3-Heights® OCR Enterprise Add-on.

To make use of the OCR engine, the OCR interface DLL and the OCR engine must be installed. The `GetOCRPluginCount` property returns the number of available OCR interface DLLs. It does not verify the corresponding OCR engines are installed and can be initialized. The OCR engine is loaded with the `SetOCREngine` method.

Returns:

The number of available OCR engines (i.e. their corresponding OCR interface DLLs).

6.6.5 GetOCRPluginName

Method: `String GetOCRPluginName(Integer iOCREngine)`

An OCR engine is accessed through an OCR plugin. Each plugin corresponds to one OCR engine. The number of OCR plugins is retrieved using `GetOCRPluginCount`. The method call `GetOCRPluginName(n)` returns the name of the nth OCR Engine which corresponds to that OCR plugin. At present, there are three OCR engines available: `"abbyy11"`, `"abbyy10"`, and `"service"`.

Parameter:

`iOCREngine` `[Integer]` The number of the OCR engine. The total number of engines is retrieved using `GetOCRPluginCount`.

Returns:

The name of the nth OCR engine. `Nothing` if it does not exist.

6.6.6 GetPdf

Method: `Variant GetPdf()`

Get the output file from memory. See also method `CreateInMemory`.

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.

6.6.7 LicenseIsValid

Property (get): `Boolean LicenseIsValid`
Static

Check if the license is valid.

6.6.8 OCRBitonalRecognition

Property (get, set): `Boolean OCRBitonalRecognition`
Default: `False`

Specify whether the images should be converted to bitonal (black and white) before OCR recognition.

Enabling this feature can improve the memory consumption of the OCR process.

Enabling this feature automatically re-embeds the original images in the output document. The setting of the property `OCREmbedOCRImage` is therefore ignored.

6.6.9 OCRDeskewImage

Property (get, set): `Boolean OCRDeskewImage`
Default: `False`

Correct the skew angle of images.

This option set to `True` has only an effect if the required information is provided by the OCR engine, which depends on the type and settings of the engine.

This option set to `True` may change the appearance of the page and is only recommended for simple scanned documents that consist of a single image.

Using the option for digital-born documents may destroy the page layout.

6.6.10 OCREmbedBarcodes

Property (get, set): `Boolean OCREmbedBarcodes`
Default: `False`

This property specifies whether the recognized barcodes are embedded in the XMP metadata.

6.6.11 OCREmbedOCRImage

Property (get, set): `Boolean OCREmbedOCRImage`
Default: `False`

This option set to `True` currently requires the `OCRDeskewImage` to be also set to `True`.

The OCR engine deskews and de-noises the input image before recognizing the characters. This option controls whether the 3-Heights® Image to PDF Converter API should use the pre-processed image or keep the original image.

Setting this option to `True` has only an effect if the pre-processed image is provided by the OCR engine, which depends on the type and settings of the engine.

If this option is set to `True`, the resulting image may have a different color space, compression and size.

Since this option currently requires `OCRDeskewImage`, it is recommended only for simple scanned documents.

6.6.12 OCRMode

Property (get, set): `Integer OCRMode`
Default: `1`

Specify behavior of the converter for files with existing OCR text. Available OCR modes are the following:

OCR mode	Description
1	Only perform OCR for images without existing OCR text (default).
2	If OCR engine is active, remove old OCR text and perform OCR for all images. Hence, existing OCR text is not removed if OCR engine is not active.
4	Only perform OCR if the input file contains no text.

6.6.13 OCRResolutionDPI

Property (get, set): `Single OCRResolutionDPI`
Default: `300`

Re-sample images to target resolution before they are sent to the OCR engine. The default is `300` DPI, which is the preferred resolution for most OCR engines.

6.6.14 OCRRotatePage

Property (get, set): `Boolean OCRRotatePage`
Default: `False`

This property specifies whether the page is rotated according to the recognized image rotation.

6.6.15 OCRThresholdDPI

Property (get, set): `Single OCRThresholdDPI`
Default: `400`

Only images with a higher resolution than the threshold are re-sampled before OCR. The default is `400` DPI. If set to `-1`, no re-sampling is applied.

6.6.16 ProductVersion

Property (get): `String ProductVersion`

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

6.6.17 Open

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

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

Parameters:

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

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

Returns:

`True` The file could be successfully opened.

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

6.6.18 OpenMem

Method: `Boolean OpenMem(Variant MemBlock, String Password)`

Open a PDF file or raster image 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.

6.6.19 SaveAs

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

Save the currently opened document.

Parameters:

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

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

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

`PermissionFlags` [`TPDFPermission`] (optional) The permission flags.

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

Returns:

`True` The opened document could successfully be saved to file.

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

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

6.6.20 SaveInMemory

Method: `Boolean SaveInMemory()`

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

Returns:

True The document could be saved in memory successfully.

False Otherwise.

6.6.21 SetLicenseKey

Method: `Boolean SetLicenseKey(String LicenseKey)`

Sets the license key.

6.6.22 SetOCREngine

Method: `Boolean SetOCREngine(String Engine)`

This method requires the 3-Heights® OCR Add-on, which is a separate product, to be installed. See also documentation for the 3-Heights® OCR Add-on.

Set the OCR engine that is used when OCR information shall be added during the conversion. If the engine's name is set to an empty string, OCR is not applied.

Parameter:

Engine [String] The name of the OCR engine (e.g. `"abbyy11"`). For every available OCR engine, there is a corresponding OCR interface DLL. The OCR interface DLLs (e.g. `pdfocrAbbyy11.ocr`) are distributed with the 3-Heights® Image to PDF Converter API and are required to communicate with the OCR engine. The names of all available OCR engines can be retrieved using the properties `GetOCRPluginCount` and `GetOCRPluginName`.

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

Returns:

`True` The OCR interface DLL was found, the OCR engine was found and the OCR engine was successfully initialized.

`False` Otherwise.

6.6.23 SetOCRLanguages

Method: `Boolean SetOCRLanguages(String Languages)`

This method requires the 3-Heights® OCR Add-on, which is a separate product, to be installed. See also documentation for the 3-Heights® OCR Add-on.

Setting languages helps the OCR engine to minimize errors by means of using dictionaries of the defined languages.

This method must be called after `SetOCREngine`.

If `SetOCRParams` is used, `SetOCRLanguages` must be called after `SetOCRParams`.

Parameter:

`Languages` `[String]` A string of one or multiple, comma-separated languages. The supported names depend on the OCR engine. The OCR engine will only use dictionaries of the set languages.

Returns:

`True` The language(s) were successfully set.

`False` Otherwise.

Example:

```
SetOCREngine("abbyy11")
SetOCRLanguages("English, German")
```

6.6.24 SetOCRParams

Method: `Boolean SetOCRParams(String Params)`

This method requires the 3-Heights® OCR Add-on, which is a separate product, to be installed. See also documentation for the 3-Heights® OCR Add-on.

By means of this method, OCR engine specific settings can be applied in the form of key-value pairs. These pairs depend on the OCR engine and are described in the corresponding manual.

Parameter:

`Params` `[String]` A list of comma-separated key value pairs. See example.

Returns:

`True` The OCR parameters were successfully set.

`False` Otherwise.

Example: Set a predefined profile for ABBYY 11.

```
SetOCREngine("abbyy11")
SetOCRParams("PredefinedProfile = DocumentArchiving_Accuracy")
```

6.7 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).

6.7.1 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	

6.7.2 TPDFCompliance Enumeration

TPDFCompliance table

TPDFCompliance	
ePDF13	PDF version 1.3
ePDF14	PDF version 1.4 (corresponds to Acrobat 5)
ePDF15	PDF version 1.5
ePDF16	PDF version 1.6 (corresponds to Acrobat 7)
ePDF17	PDF version 1.7, ISO 32000-1
ePDF20	PDF version 2.0, ISO 32000-2
ePDFA1a	PDF/A 1a, ISO 19005-1, conformance level A
ePDFA1b	PDF/A 1b, ISO 19005-1, conformance level B
ePDFA2a	PDF/A 2a, ISO 19005-2, conformance level A
ePDFA2b	PDF/A 2b, ISO 19005-2, conformance level B
ePDFA2u	PDF/A 2u, ISO 19005-2, conformance level U
ePDFA3a	PDF/A 3a, ISO 19005-3, conformance level A
ePDFA3b	PDF/A 3b, ISO 19005-3, conformance level B
ePDFA3u	PDF/A 3u, ISO 19005-3, conformance level U

Note that only the values listed above are supported.

6.7.3 TPDFCompression Enumeration

TPDFCompression table

TPDFCompression	Description
eComprRaw	No compression
eComprJPEG	Joint Photographic Expert Group
eComprFlate	Flate compression
eComprLZW	Lempel-Ziv-Welch
eComprGroup3	CCITT Fax Group 3
eComprGroup3_2D	CCITT Fax Group 3 2D
eComprGroup4	CCITT Fax Group 4
eComprJBIG2	Joint Bi-level Image Experts Group

TPDFCompression table

eComprJPEG2000	JPEG2000
eComprUnknown	Unknown compression

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

6.7.4 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.

6.7.5 TPDFErrorCode Enumeration

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

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

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

TPDFErrorCode table

PDF_E_FILEOPEN	Failed to open the file.
PDF_E_FILECREATE	Failed to create the file.

The following warnings can occur when creating PDF/A

TPDFErrorCode	Description
PDF_I2P_W_OUTPUTINTENT	An output intent was required. An sRGB profile was created.
PDF_I2P_W_SMASK	The soft mask of the image was removed during the conversion (PDF/A-1 only).
PDF_I2P_W_JPXDECODE	JPEG2000 compression was replaced by JPEG compression (PDF/A-1 only).

6.7.6 TPDFOrientation Enumeration

TPDFOrientation table

TPDFOrientation	
eOrientationUndef	Undefined
eOrientationTopLeft	Image is untransformed.
eOrientationTopRight	Before viewing, image is flipped horizontally.
eOrientationBottomRight	Before viewing, image is rotated by 180°.
eOrientationBottomLeft	Before viewing, image is flipped vertically.
eOrientationLeftTop	Before viewing, image is rotated by 90° clockwise and then flipped horizontally.
eOrientationRightTop	Before viewing, image is rotated by 90° clockwise.
eOrientationRightBottom	Before viewing, image is rotated by 90° clockwise and flipped vertically
eOrientationLeftBottom	Before viewing, image is rotated by 90° counter-clockwise.

6.7.7 TPDFPermission Enumeration

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

TPDFPermission table

TPDFPermissionFlag	Description

TPDFPermission table

ePermNoEncryption	Do not apply encryption. This enumeration value cannot be combined with other values. When using this enumeration, set both passwords to an empty string or Nothing.
ePermNone	Grant no permissions
ePermPrint	Low resolution printing
ePermModify	Changing the document
ePermCopy	Content copying or extraction
ePermAnnotate	Annotations
ePermFillForms	Filling of form fields
ePermSupportDisabilities	Support for disabilities
ePermAssemble	Document assembly
ePermDigitalPrint	High resolution printing
ePermAll	Grant all permissions

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

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

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

Allow Printing

```
Permission = Permission Or ePermPrint
```

Prohibit Printing

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

6.8 Supported image extensions

The following extensions are supported:

Supported image extensions table

Supported Image Extensions	
`.tif, .tiff`	Tagged Image File Format
`.jpg, .jpe, .jpeg`	Joint Photographic Expert Group
`.png`	Portable Network Graphics
`.gif`	Graphics Interchange Format
`.bmp`	Window Bitmap
`.jb2`	Joint Bi-level Image Experts Group
`.jp2`	JPEG2000
`.jpx`	Extended JPEG2000
`.pbm,.pgm,.pnm,.ppm`	Portable Bitmap File Format
`.eps`	Encapsulated PostScript (Output only)

6.9 Supported image compression types

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

6.9.1 No compression (raw)

Raw means no compression is applied.

6.9.2 DCT (JPEG)

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

Application area	Sampled continuous-tone pictures (photographs)

6.9.3 Flate (ZIP)

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

6.9.4 LZW

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

6.9.5 CCITT Fax Group 3 and 4

Developer	International Telecommunications Union (ITU), formerly known as the Comité Consultatif International Téléphonique et Télégraphique
Version	PDF 1.0 and later, PDF/A-1
Color depth	1 bit per pixel
Compression type	Lossless

Compression algorithm	**Group3** 1-dimensional version of the CCITT Group 3 Huffman encoding algorithm. **Group 3 2D** 2-dimensional version of the CCITT Group 3 Huffman encoding algorithm. **Group 4** An advanced version of a bitonal algorithm based on the CCITT Fax Group 3 2D compression.
Application area	Line-art image, bitonal, faxes

6.9.6 JBIG2

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

6.9.7 JPEG2000

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

6.10 Specification of resolution and image dimensions

The three image dimensions (resolution, true_width and width) depend on each other. They have to satisfy the following relation (the same is true for the height and the true_height):

$$\text{resolution} = \frac{\text{width}}{\text{true_width}}.$$

If the width (see `Width`) and the height (see `Height`) are set at the same time, the height is omitted due to priority of width. Equivalently, true_width has priority to true_height. All transformations are done respecting image proportions. The property `AllowResampling` can be used to force in certain situations to perform a re-sampling. The table below enlists the possible parameter combinations and shows the behavior of the Image to Image Converter API.

Properties				Properties of Output Image
DPI	Width/Height	TrueWidth/TrueHeight	AllowResampling	
not set	not set	not set	true/false	no changes
not set	set	not set	true/false	true_width/true_height constant, resolution modified
not set	set	set	true/false	resolution modified
not set	not set	set	true	true_width/true_height constant, width/height modified
not set	not set	set	false	width/height constant, true_width/true_height modified
set	not set	not set	true	true_width/true_height constant, width/height modified
set	not set	not set	false	width/height constant, true_width/true_height modified
set	set	not set	true/false	true_width/true_height modfied
set	not set	set	true/false	width/height modified
set	set	set	true/false	width/height and true_width/true_height have priority over resolution

7 Version history

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

7.1 Changes in versions 6.19–6.27

- **Update** license agreement to version 2.9

7.2 Changes in versions 6.13–6.18

- **New** supported image format HEIC/HEIF.

7.3 Changes in versions 6.1–6.12

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

Interface PDFCodec

- **New** Property `Name`.

7.4 Changes in version 5

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

7.5 Changes in version 4.12

- **New** OCR plugin `"abbyy12"` for the ABBYY FineReader 12 engine.
- **Improved** reading and recovery of corrupt TIFF images.
- **New** HTTP proxy setting in the GUI license manager.

Interface `Img2Pdf`

- **Changed** behavior, method `SetPageSize` disables `AdjustPage`.

7.6 Changes in version 4.11

- **Merged** manual `PdfOcrAPI.pdf` into `Image2PdfAPI.pdf`.
- **New** support for reading and writing PDF 2.0 documents.
- **Improved** font subsetting of CFF and OpenType fonts.
- **Improved** repair of corrupt image streams.
- [PHP] **New** Interface for Windows and Linux. Supported versions are PHP 5.6 & 7.0 (Non Thread Safe). The `Img2PdfAPI` PHP Interface is contained in the 3-Heights® PDF Tools PHP5.6 Extension and the 3-Heights® PDF Tools PHP7.0 Extension.
- [C] **Changed** 32-bit binaries on Windows that link to the API need to be recompiled due to a change of the used mangling scheme.

7.7 Changes in version 4.10

- [C] **Clarified** Error handling of `TPdfStreamDescriptor` functions.

Interface `Img2Pdf`

- [.NET, C, COM, Java] **New** property `AdjustOrientation`: Adjust page orientation.

7.8 Changes in version 4.9

- **Improved** metadata generation for standard PDF properties.
- [C] **Changed** return value `pfGetLength` of `TPDFStreamDescriptor` to `pos_t`[8].

Interface `Img2Pdf`

- [.NET, C, COM, Java] **New** property `Dithering`: Get or set the dithering algorithm.
- [.NET, C, COM, Java] **Changed** property `BitsPerPixel`: Get or set the color depth. Available: Bi-tonal: 1. When using 1 bit per pixel, it is suggested to set a suitable dithering algorithm.

7.9 Changes in version 4.8

Interface `Img2Pdf`

- [.NET, C, COM, Java] **New** property `ProductVersion` to identify the product version.
- [.NET] **Deprecated** method `GetLicenseIsValid`
- [.NET] **New** property `LicenseIsValid`

Interface `Img2Img`

- [.NET, C, COM, Java] **New** property `Height` to set the height of the image in pixel.
- [.NET, C, COM, Java] **New** property `Width` to set the width of the image in pixel.

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

- [.NET, C, COM, Java] **New** property `TrueHeight` to set the true height of the image in mm.
- [.NET, C, COM, Java] **New** property `TrueWidth` to set the true width of the image in mm.
- [.NET, C, COM, Java] **New** property `AllowResampling` to force re-sampling.

Interface PDFCodec

- [.NET, C, COM, Java] **New** property `DefaultDPI` to set the resolution of input image, if input image has none.

Interface PdfOcr

- [.NET, C, COM, Java] **New** property `ProductVersion` to identify the product version.
- [.NET] **Deprecated** method `GetLicenseIsValid`.
- [.NET] **New** property `LicenseIsValid`.

8 Licensing, copyright, and contact

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

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

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

Contact

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