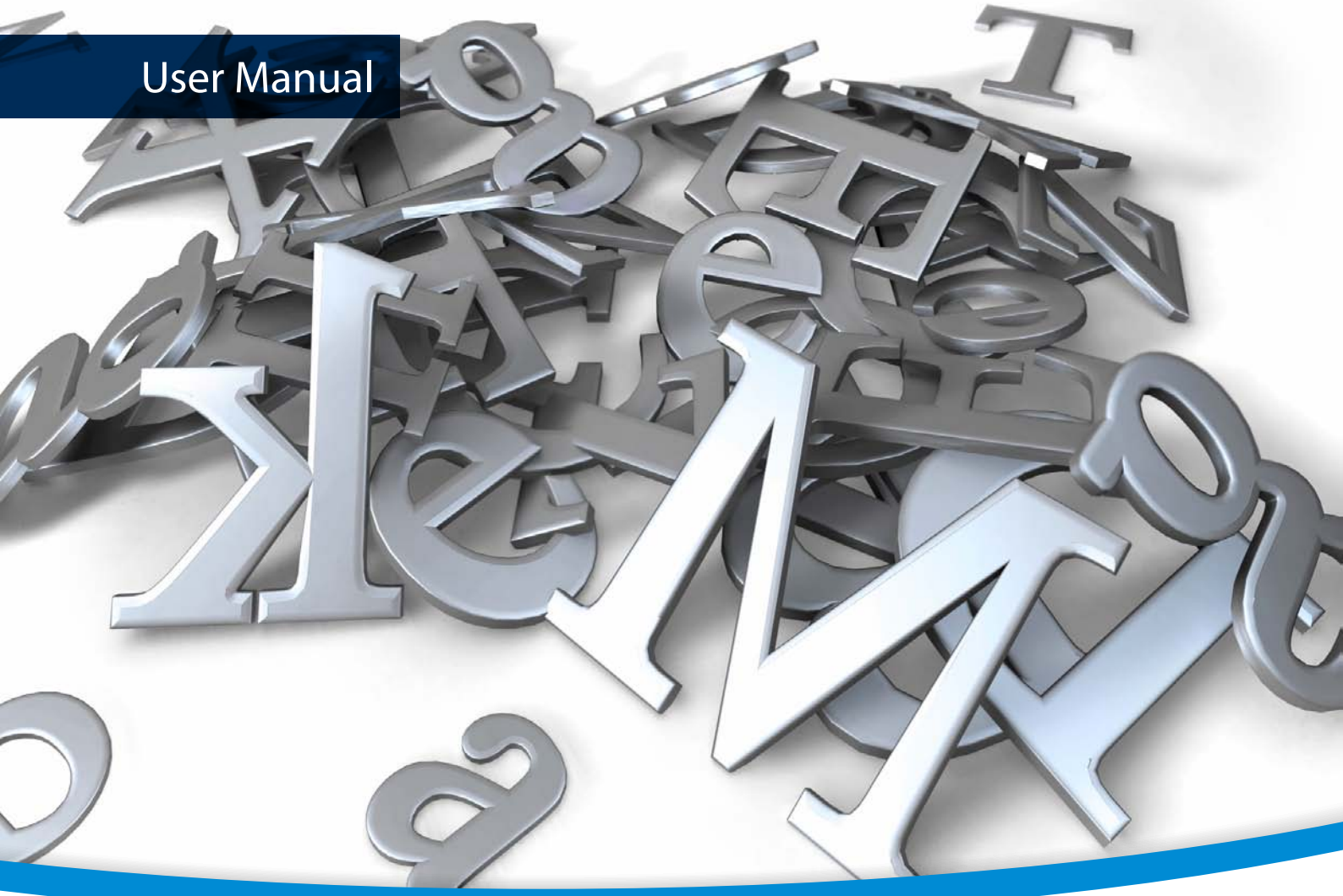
User Manual



3-Heights™ PDF OCR API

PDF/A conform

Version 4.12.26.5

PDF-TOOLS.COM
Premium PDF Technology

Contents

1 Introduction

1.1 Description

The 3-Heights™ PDF OCR API enhances PDF documents using information detected by an OCR engine.

All text in PDF documents can be made extractable, regardless of how text is included in the document. Specifically, text in images, text written with vector graphics or other graphical effects (e.g. transparency effects), or text using a font that does not provide extractable text (i.e. Unicode information) can be detected.

Tagging of OCR text for accessibility is supported. This is useful as preparation for PDF/A level A conversion or to process tagged documents.

Detected barcodes or QR codes can be extracted or embedded into the document's metadata.

The product is optimized for performance, which guarantees low latency and high document throughput. This is achieving by minimizing the number of required OCR operations. Furthermore, OCR operations are executed asynchronously, if supported by the OCR engine.

1.2 Functions

1.2.1 Features

- Make text extractable
  - Text contained in images
  - Text with fonts that have no Unicode information
  - Text written using vector graphics (e.g. in CAD drawings)
  - Any visible text, regardless of the type of graphics objects used
- Scan improvements
  - Deskew scanned images
  - Rotate pages according to the recognized rotation of scan
- Detect barcodes and QR codes
- Process embedded files
- Tagging of OCR text for accessibility
- High performance
  - Asynchronous processing
  - Page analysis and result caching to minimize OCR operations
- High quality
  - PDF/A compliant
  - High-fidelity conversion of existing page content
  - 3-Heights™ PDF Rendering Engine 2.0.
  - Automatic detection of optimal OCR resolution

1.2.2 Formats

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

1.2.3 Compliance

Standards:

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

1.3 Interfaces

The following interfaces are available: C, Java, .NET

1.4 Operating Systems

The 3-Heights™ PDF OCR API is available for the following operating systems:

- Windows 7, 8, 8.1, 10 – 32 and 64 bit
- Windows Server 2008, 2008 R2, 2012, 2012 R2, 2016 – 32 and 64 bit
- HP-UX 11i and later PA-RISC2.0 – 32 bit
- HP-UX 11i and later ia64 (Itanium) – 64 bit
- IBM AIX 6.1 and later – 64 bit
- Linux 2.6 – 32 and 64 bit
- Oracle Solaris 2.8 and later, SPARC and Intel
- FreeBSD 4.7 and later (32 bit) or FreeBSD 9.3 and later (64 bit, on request)
- macOS 10.4 and later – 32 and 64 bit

1.5 How to Best Read this Manual

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

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

2 Installation and Deployment

2.1 Windows

The 3-Heights™ PDF OCR API comes as a ZIP archive.

The installation of the software requires the following steps.

1. You need administrator rights to install this software.
2. Log in to your download account at http://www.pdf-tools.com. Select the product "PDF OCR API". If you have no active downloads available or cannot log in, please contact pdfsales@pdf-tools.com for assistance.
   You will find different versions of the product available. We suggest to download the version, which is selected by default. If another is required, it can be selected using the combo box.
   The product comes as a ZIP archive containing all files.
   There are 32 and 64-bit versions of the product available. While the 32-bit version runs on both, 32 and 64-bit platforms, the 64-bit version runs on 64-bit platforms only. The ZIP file contains both the 32-bit and the 64-bit version of the product.
3. Unzip the archive to a local folder, e.g. `C:\Program Files\PDF Tools AG\`.
   This creates the following subdirectories:

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

4. (Optional) Register your license key using the License Management.
5. Identify which interface you are using. Perform the specific installation steps for that interface described in chapter Interface Specific Installation Steps
6. Make sure your platform meets the requirements regarding fonts described in chapter Fonts.
7. Download and install the 3-Heights™ OCR Service, the OCR Service client plugin, 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 Add-On for Barcode and QR Code Recognition: OcrBarcodes.pdf
   - 3-Heights™ OCR Service: OcrService.pdf from the separate product kit.

2.2 Unix

This section describes installation steps required on all Unix platforms, which includes Linux, macOS, Oracle Solaris, IBM AIX, HP-UX, FreeBSD and others.

The Unix version of the 3-Heights™ PDF OCR API provides two interfaces:

- Java interface
- Native C interface

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

File Description

Name	Description
`bin/‹platform›/libPdfOcrAPI.so`	This is the shared library that contains the main functionality. The file's extension varies depending on the type of UNIX system. The directory `‹platform›` is either `x86` containing the 32-bit version of the library, or `x64` for the 64-bit version.
`bin/‹platform›/*.ocr`	These are OCR plugin modules.
`doc/*.*`	Documentation
`include/*.h`	Contains header files to include in your C/C++ project.
`jar/PdfOcrAPI.jar`	Java API archive.
`samples`	Example code.

2.2.1 All Unix Platforms

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:
   - *On Linux*:

     ```
     ldd libPdfOcrAPI.so
     ```

   - *On AIX*:

     ```
     dump -H libPdfOcrAPI.so
     ```

   In case the above reports any missing libraries you have three options:
   a. Download an archive that is linked to another 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. On Linux it usually suffices to install the package `libstdc++6`.
   c. Use GNU shared libraries provided by PDF Tools AG:
      1. Go to http://www.pdf-tools.com and navigate to "Support" →"Utilities".
      2. Download the GNU shared libraries for your platform.
      3. Install the libraries manually according your system's documentation. On Linux this typically involves copying them to your library directory, e.g. `/usr/lib` or `/usr/lib64`, and running `ldconfig`.
      4. Verify that the GNU shared libraries required by the product are available on your system now.
3. Create a link to the shared library from one of the standard library directories, e.g:

   ```
   ln -s /opt/pdf-tools.com/bin/‹platform›/libPdfOcrAPI.so /usr/lib
   ```

4. Optionally register your license key using the Command Line License Manager Tool.
5. Identify which interface you are using. Perform the specific installation steps for that interface described in chapter Interface Specific Installation Steps
6. Make sure your platform meets the requirements regarding fonts described in chapter Fonts.
7. Download and install the 3-Heights™ OCR Service, the OCR Service client plugin, 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 Add-On for Barcode and QR Code Recognition: `OcrBarcodes.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. We suggest that you create a file link for this purpose by using the following command:

```
ln libPdfOcrAPI.dylib libPdfOcrAPI.jnilib
```

2.3 Interfaces

The 3-Heights™ PDF OCR API provides three different interfaces. The installation and deployment of the software depend on the interface you are using. The table below shows the supported interfaces and examples with which programming languages they can be used.

Interface	Programming Languages
`.NET`	The MS software platform .NET can be used with any .NET capable programming language such as: - C# - VB .NET - J# - others This interface is available in the Windows version only.
`Java`	The Java interface is available on all platforms.
`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 developer kit (SDK) contains all files that are used for developing the software. The role of each file with respect to the four different interfaces is shown in table Files for Development. The files are split in four categories:

Req. This file is required for this interface.

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

Doc. This file is for documentation only.

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

Files for Development

Name	.NET	Java	C
bin\‹platform›\PdfOcrAPI.dll	Req.	Req.	Req.
bin*NET.dll	Req.		
bin*NET.xml	Doc.		
bin\‹platform›*.ocr	Req.	Req.	Req.
doc*.pdf	Doc.	Doc.	Doc.
doc\javadoc*.*		Doc.	
include\pdfocrapi_c.h			Req.
include*.*			Opt.
jar\PdfOcrAPI.jar		Req.	
lib\‹platform›\PdfOcrAPI.lib			Req.[1]
samples*.*	Doc.	Doc.	Doc.

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

File Description

Name	Description
bin\‹platform›\PdfOcrAPI.dll	This is the DLL that contains the main functionality (required).
bin*NET.dll	The .NET assemblies are required when using the .NET interface. The files bin*NET.xml contain the corresponding XML documentation for MS Visual Studio.
bin\‹platform›*.ocr	These are OCR plugin DLLs.[2]
doc*.*	Various documentations.
include*.*	Contains files to include in your C / C++ project.
lib\‹platform›\PdfOcrAPI.lib	On Windows operating systems, the object file library needs to be linked to the C/C++ project.
jar\PdfOcrAPI.jar	The Java API archive.
samples*.*	Contains sample programs in different programming languages.

[1] Not required for Unix-like operating systems and Mac.
[2] These files must reside in the same directory as PdfOcrAPI.dll.

2.3.2 Deployment

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

Files for Deployment

Name	.NET	Java	C
bin\‹platform›\PdfOcrAPI.dll	Req.	Req.	Req.
bin*NET.dll	Req.		
bin\‹platform›*.ocr	Req.	Req.	Req.
jar\PdfOcrAPI.jar		Req.	

The deployment of an application works as described below:

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

2.4 Interface Specific Installation Steps

2.4.1 Java Interface

The 3-Heights™ PDF OCR API requires Java version 6 or higher.

For compilation and execution When using the Java interface, the Java wrapper `jar\PdfOcrAPI.jar` needs to be on the CLASSPATH. This can be done by either adding it to the environment variable CLASSPATH, or by specifying it using the switch `-classpath`:

```
javac -classpath ".;C:\Program Files\PDF Tools AG\jar\PdfOcrAPI.jar" sample.java
```

For execution Additionally the library `PdfOcrAPI.dll` needs be in one of the system's library directories[3] or added to the Java system property `java.library.path`. This can be achieved 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` depending on the platform of the Java VM[4].

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

Note that on Unix-type systems, the path separator usually is a colon and hence the above changes to something like:

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

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

2.4.2 .NET Interface

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

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

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

The native DLL `PdfOcrAPI.dll` is called by the .NET assembly `PdfOcrNET.dll`.

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

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

Troubleshooting: TypeInitializationException

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

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

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

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

The following sections describe in more detail, how to resolve the respective issue.

Troubleshooting: DllNotFoundException

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

Resolve this by either:

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

Troubleshooting: BadImageFormatException

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

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

AnyCPU This means, that the application will run as a 32-bit process on 32-bit Windows and as 64-bit process on 64-bit Windows. When using AnyCPU one has to use a different native DLL, depending on the platform of Windows. This can be ensured either when installing the application (by installing the matching native DLL) or at application start-up (by determining the application's platform and ensuring the matching native DLL is loaded).

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

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

2.4.3 C Interface

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

2.5 Uninstall, Install a New Version

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

Installing a new version does not require to previously uninstall the old version. The files of the old version can directly be overwritten with the new version. If using the COM interface, the new DLL must be registered, un-registering the old version is not required.

2.6 Fonts

Fonts are required, if OCR is preformed and OCR text is added to a PDF document. Hereby 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.

2.6.1 Font Cache

A cache of all fonts in all Font Directories is created. If fonts are added or removed from the font directories, the cache is updated automatically.

In order to achieve optimal performance, make sure that the cache directory is writable for the 3-Heights™ PDF OCR 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.

2.7 Note about the Evaluation License

With the evaluation license the 3-Heights™ PDF OCR API automatically adds a watermark to the output files.

2.8 Special Directories

2.8.1 Directory for temporary files

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

Unix

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

2.8.2 Cache Directory

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

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

Windows

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

Linux, macOS and other Unixes

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

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

2.8.3 Font Directories

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

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

Windows

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

macOS

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

Linux and other Unixes

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

3 License Management

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

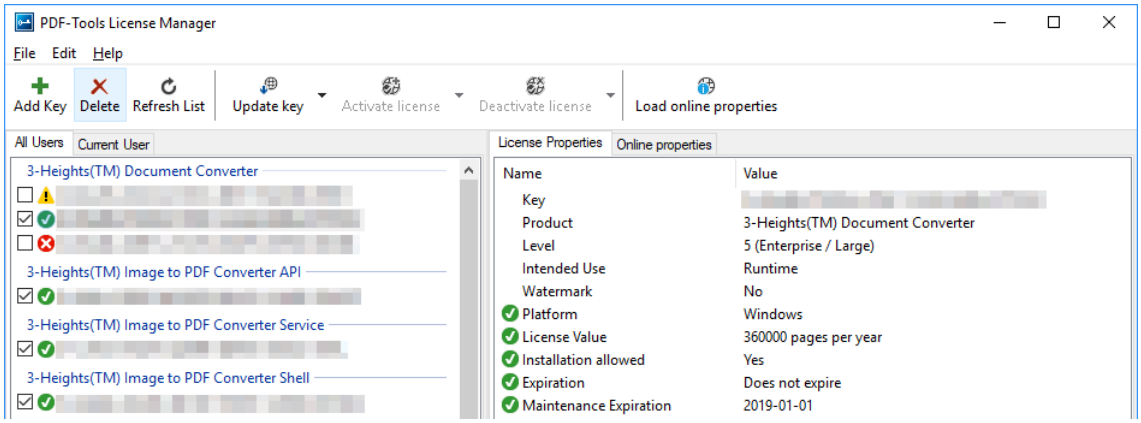
3.1 License Installation and Management

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

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

3.1.1 Graphical License Manager Tool

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



List all installed license keys

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

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

Add and delete license keys

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

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

Display the properties of a license

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

3.1.2 Command Line License Manager Tool

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

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

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

```
licmgr
```

List all installed license keys

```
licmgr list
```

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

Example:

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

Add and delete license keys

Install new license key:

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

Delete old license key:

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

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

g For all users

u Current user

Display the properties of a license

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

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

Example:

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

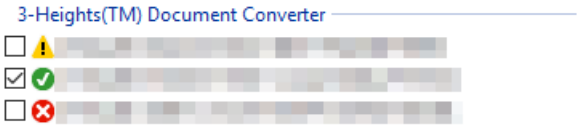
3.2 License Selection and Precedence

3.2.1 Selection

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

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

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



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

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

3.2.2 Precedence

License keys are considered in the following order:

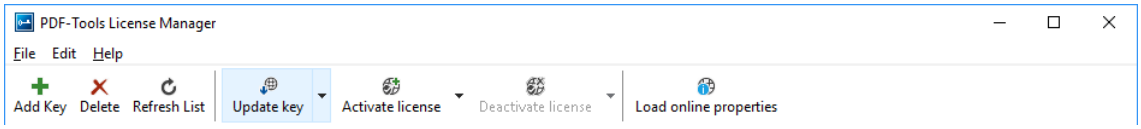
1. License key passed at runtime.
2. License selected for the current user
3. License selected for the current user (legacy key format)
4. License selected for all users
5. License selected for all users (legacy key format)

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

3.3 Key Update

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

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



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

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

3.4 License activation

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

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

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

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

3.4.1 Activation

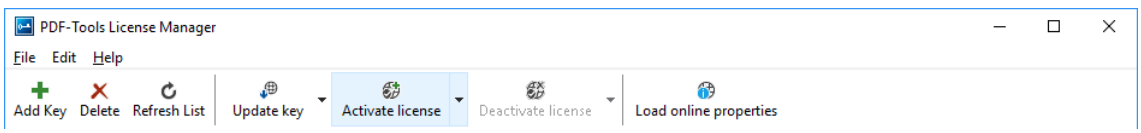
The License can be activated directly within the license manager. Every activation increases the activation count of the license by 1.

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

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

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

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



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

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

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

Note that the key has to be installed first.

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

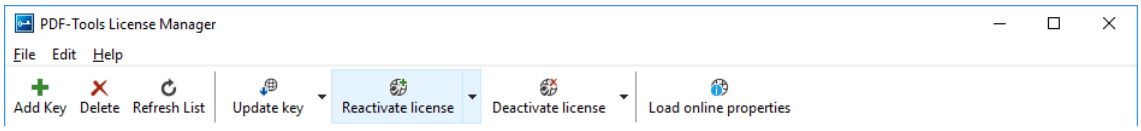
```
licmgr activate -cd 1-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
licmgr activate -c "custom comment" 1-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

3.4.2 Reactivation

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

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

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



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

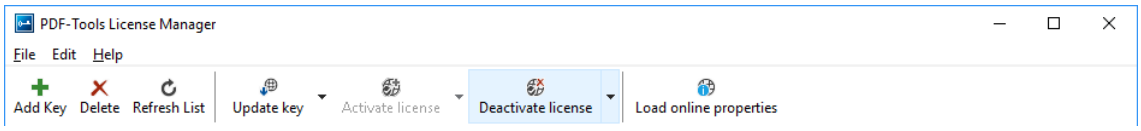
```
licmgr activate 1-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

3.4.3 Deactivation

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

The process for deactivation is similar to the activation process.

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



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

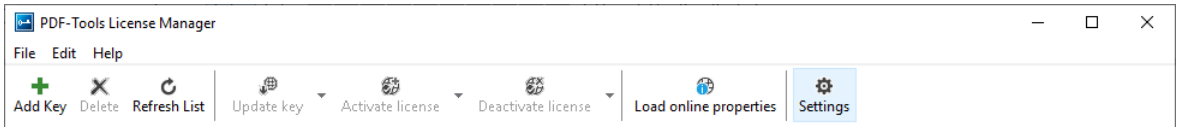
```
licmgr deactivate 1-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

3.5 Proxy Setting

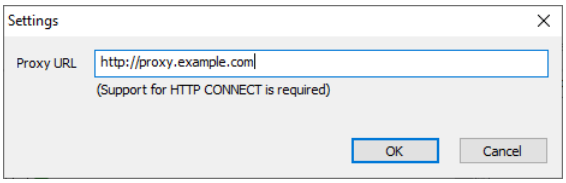
A proxy URL can be configured for computers that cannot access the internet without a web proxy.

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

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



and enter the proxy URL in the respective field:



3.6 Offline Usage

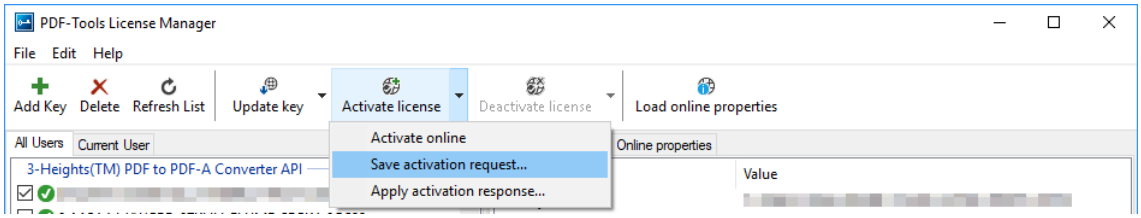
The following actions in the license manager need access to the internet:

- License Activation
- License Reactivation
- License Deactivation
- Key Update

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

3.6.1 First Step: Create a Request File

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



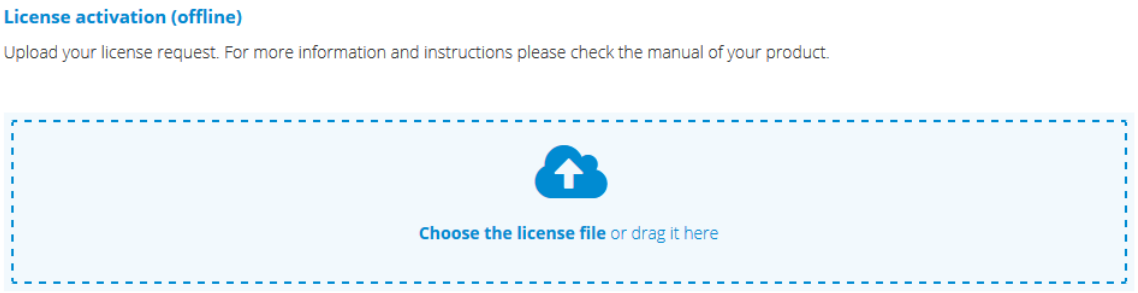
With the Command Line Interface use the `-fs` option to specify the destination path of the request file:

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

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

3.6.2 Second Step: Use Form on Website

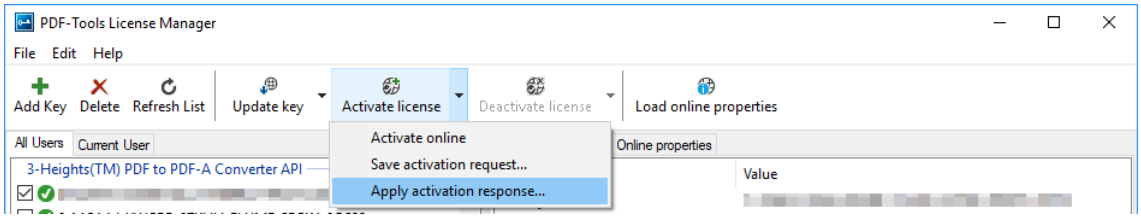
Open the following website in a web browser: http://www.pdf-tools.com/pdf20/en/mypdftools/licenses-kits/license-activation/ Upload the request by dragging it onto the marked area:



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

3.6.3 Third Step: Apply the Response File

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



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

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

3.7 License Key Versions

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

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

3.8 License Key Storage

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

3.8.1 Windows

The license keys are stored in the registry:

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

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

3.8.2 macOS

The license keys are stored in the file system:

- /Library/Application Support/PDF Tools AG (for all users)
- ~/Library/Application Support/PDF Tools AG (for the current user)

3.8.3 Unix/Linux

The license keys are stored in the file system:

- /etc/opt/pdf-tools (for all users)
- ~/.pdf-tools (for the current user)

> **Note:** The user, group and permissions of those directories are set solely by the license manager tool. It may be necessary to change permissions to make the licenses readable for all users. Example:
>
> ```
> chmod -R go+rx /etc/opt/pdf-tools
> ```

3.9 Troubleshooting

3.9.1 License key cannot be installed

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

Possible causes:

- The license manager application is an older version that only supports the legacy key format.

Solution

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

3.9.2 License is not visible in license manager

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

Possible causes:

- The license manager application is an older version that only supports the legacy key format.

Solution

Use a current version of the license manager application.

3.9.3 License is not found at runtime

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

Possible causes:

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

Solution

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

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

3.9.4 Eval watermark is displayed where it should not

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

Possible causes:

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

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

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

Solution

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

3.9.5 Activation is not recognized

The license is installed and activated in the license manager, but the software does not recognize it as activated.

The error message is: `"The license has not been activated."`

Possible causes:

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

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

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

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

- The software was not restarted after activating the license.

Solution

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

3.9.6 Activation is invalidated too often

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

Possible causes:

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

Solution

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

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

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

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

3.9.7 Connection to the licensing service fails

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

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

Possible causes:

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

Solution

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

If this is not possible, try Offline Usage instead.

3.9.8 Offline usage fails due to a request/response mismatch

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

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

Possible causes:

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

Solution

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

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

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

4 User's Guide

4.1 Overview of the API

4.1.1 Process Description

The following is a simplified process description of the 3-Heights™ PDF OCR API:

1. **Open document**: The input document is opened.
2. **Process document**: The document is processed and the result written to the output.
   1. **Process pages**
      1. Analyze page: The page's content is analyzed in order to determine, whether processing by the OCR engine is required or not (see chapter Page Analysis below).
      2. OCR page (optional)
         1. Determine optimal OCR resolution
         2. Render page: The page is converted to an image using the 3-Heights™ PDF Rendering Engine 2.0.
         3. Send image to OCR engine.
         4. Process OCR results (see chapter OCR Result Processing below).
      3. Copy page: The page is copied to the output. If available, information from the OCR engine is added.
   2. **Process embedded files**: Embedded files can be processed recursively or copied as-is.

Page Analysis

The page's content is analyzed. The modes of ocr parameters specified for images (`ImageOcrParams`), text (`TextOcrParams`), and pages (`PageOcrParams`) are evaluated. The page is processed by the OCR engine if required by any of the modes.

OCR Result Processing

Each OCR result object is processed as follows:

1. If it is a barcode, it is processed according to the `BarcodeMode`.
2. If its location is on an image on the page, it is processed according to the `ImageOcrMode`.
3. If it corresponds to text on the page, it is processed according to the `TextOcrMode`.
4. Otherwise it is added as OCR text to the page, if the `PageOcrMode` is not `None`.

4.1.2 Example

Example: C# example that shows how to add OCR text to images.

```
void OcrProcessImages(Stream input, Stream output)
{
    // Open input document from stream
    using (var document = Document.Open(input, null))
    {
        // Create OCR engine
        using (var engine = Engine.Create(engineName))
        {
            // Configure OCR engine
```

```
            engine.SetParameters(engineParams);
            engine.SetLanguages(engineLanguages);

            // Set process parameters
            var ocr = new OcrParams();
            ocr.Engine = engine;

            var imageOcr = new ImageOcrParams();
            imageOcr.Mode = ImageOcrMode.UpdateText;

            // Process document and write result to output stream
            using (var warnings = document.Process(output, null, ocr, imageOcr,
                                                   null, null, null))
            { }
        }
    }
}
```

4.2 Use Cases

This chapter describes some common use cases.

4.2.1 How to make text extractable

This example shows how text in a PDF document can be made extractable. This is suitable both for born-digital and scanned documents.

> **Note:** For tagged input documents, the configuration described in How to tag scans for accessibility (PDF/A level A) should be used.

OCR Engine Configuration

Abbyy FineReader 11

The following profile configuration `abbyy11_text.ini` is optimized to extract as much text as possible:

```
[PagePreprocessingParams]
CorrectOrientation=TRUE
[PageAnalysisParams]
DetectVerticalEuropeanText=TRUE
[ObjectsExtractionParams]
DetectTextOnPictures = TRUE
```

Use the profile using the OCR engine running on the OCR Service:

```
using (var engine = Engine.Create("service"))
{
    engine.SetParameters(@"Profile=C:\path\to\abbyy11_text.ini");
    engine.SetLanguages(languages);
    ...
```

```
}
```

Note: It is important that `C:\path\to\abbyy11_text.ini` is the path to the configuration file on the OCR Service and the OCR Service process has read permissions.

Processing Configuration

1. **Detect text contained in images**: For documents that contain images, processing of images can be activated by setting an image OCR mode (see `ImageOcrMode`):

```
var imageOcr = new ImageOcrParams();
imageOcr.Mode = ImageOcrMode.UpdateText;
```

2. **Make text extractable**: For documents that contain non-extractable text, processing of text can be activated by setting a text OCR mode (see `TextOcrMode`):

```
var textOcr = new TextOcrParams();
textOcr.Mode = TextOcrMode.Update;
```

3. **Make other visible text extractable**: For documents that contain other forms of visible text, pages can be OCR processed by setting a page OCR mode (see `PageOcrMode`):

```
var pageOcr = new PageOcrMode();
pageOcr.Mode = PageOcrMode.IfNoText;
```

Process Document

```
var ocr = new OcrParams();
ocr.Engine = engine;

using (var warnings = document.Process(output, null, ocr, imageOcr,
                                       textOcr, pageOcr, null))
{
    ...
}
```

4.2.2 How to tag scans for accessibility (PDF/A level A)

"Tagging" adds structural information to a PDF. This information can be used e.g. to read the document to the visually impaired.

The 3-Heights™ PDF OCR API supports tagging scans, e.g. such that they can be converted to PDF/A level A.

Tagging of scans that contain no figures or pictures, will conform to the PDF/UA specification (ISO 14289-1). For figures and pictures an alternative representation or replacement text must be included. Because this information is not provided by the OCR engine, tagging of such files cannot conform to PDF/UA.

Prerequisites

1. The OCR engine must provide structural information for OCR results. We recommend Abbyy FineReader 11 or newer.
2. If the 3-Heights™ OCR Service is used, it must be newer than 4.11.21.0.

OCR Engine Configuration

Abbyy FineReader 11

The following profile configuration `abbyy11_tagging.ini` is suitable:

```
[PagePreprocessingParams]
CorrectOrientation=TRUE
```

This is essentially the predefined profile "DocumentConversion_Accuracy", but can also handle rotated pages.

Use the profile using the OCR engine running on the OCR Service:

```
using (var engine = Engine.Create("service"))
{
    engine.SetParameters(@"Profile=C:\path\to\abbyy11_tagging.ini");
    engine.SetLanguages(languages);
    ...
}
```

> **Note:** It is important that `C:\path\to\abbyy11_tagging.ini` is the path to the configuration file on the OCR Service and the OCR Service process has read permissions.

Processing Configuration

1. Activate processing of images by setting an image OCR mode:

```
var imageOcr = new ImageOcrParams();
imageOcr.Mode = ImageOcrMode.UpdateText;
```

2. (Optional) Enable scan enhancements:

```
imageOcr.RotateScan = true;
imageOcr.DeskewScan = true;
```

3. Activate creation of tagging information by setting the tagging mode:

```
var pageOcr = new PageOcrMode();
pageOcr.Tagging = TaggingMode.Update;
```

Process Document

```
var ocr = new OcrParams();
ocr.Engine = engine;

using (var warnings = document.Process(output, null, ocr, imageOcr,
                                       null, pageOcr, null))
{
    ...
}
```

Error Handling

Tagging errors are classified as warnings by the 3-Heights™ PDF OCR API. Because tagging is crucial for this process, tagging warnings returned by `Process` must be checked and treated as errors.

```
foreach (var warning in warnings)
{
    if (warning.Code == WarningCode.Tagging)
        throw new Exception(warning.Message);
}
```

4.2.3 How to detect barcodes

This example shows how to detect and extract barcodes and QR codes.

OCR Engine Configuration

There are two OCR engines available that support barcode recognition.

Barcodes OCR Engine

The OCR engine "`barcodes`" is a specialized plugin for barcode and QR code recognition.

No engine parameters are required. However, in order to speed up the recognition process, it can be limited to a specific set of code types:

```
using (var engine = Engine.Create("barcodes"))
{
    engine.SetParameters(@"BarcodeTypes=QRCode");
    ...
}
```

Abbyy FineReader 11 Engine

The predefined profile "BarcodeRecognition_Accuracy" is optimized for this purpose and will detect all supported types of barcodes and QR codes.

Use the profile using the OCR engine running on the OCR Service:

```
using (var engine = Engine.Create("service"))
```

```
{
    engine.SetParameters(@"PredefinedProfile=BarcodeRecognition_Accuracy");
    ...
}
```

> **Note:** This profile detects barcodes only and cannot be used to make any text extractable. However, with a profile file barcode and text recognition can be performed in one step

Processing Configuration

1. **Activate OCR processing**: Activate OCR processing of all pages:

```
var pageOcr = new PageOcrMode();
pageOcr.Mode = PageOcrMode.All;
```

2. **Enable barcode extraction**: Write all detected barcodes in XML format to a memory stream:

```
var barcodes = new BarcodeParams();
barcodes.Mode = BarcodeMode.Extract;
barcodes.XmlOutput = new MemoryStream();
```

The format of the barcodes XML file is documented in the XML schema `barcodes.xsd` located in the documentation folder of the 3-Heights™ PDF OCR API.

Process Document

```
var ocr = new OcrParams();
ocr.Engine = engine;

using (var warnings = document.Process(output, null, ocr, null,
                                       null, pageOcr, barcodes))
{ }

barcodes.XmlOutput.Seek(0, SeekOrigin.Begin);
XElement barcodes = XElement.Load(barcodes.XmlOutput, LoadOptions.PreserveWhitespace);

foreach (var barcode in barcodes.Elements(barcodes.Name.Namespace + "barcode"))
{
    ...
}
```

4.3 How to optimize the performance

Minimize the number of OCR operations

A page is processed by the OCR engine only, if required by either the `ImageOcrMode`, `TextOcrMode`, or `PageOcrMode`. Therefore, these modes should be set carefully, such that no unnecessary OCR operations are performed.

Use one or more OCR Service instances

With the OCR Service, multiple pages can be OCR processed concurrently. For this reason, it is recommended to use the OCR Service plugin.

Parallel processing is controlled by:

- the number of running OCR Services
- the number of parallel processes configured in each OCR Service
- the version of the OCR Service, which should be 4.11.20.0 or higher

Optimize OCR engine performance

By setting apt OCR engine parameters, the performance can be improved. Specifically, it is recommended to deactivate all features that are not required or use a specialized predefined profile. Consult the manual of the OCR engine for more information.

Mass OCR processing

When processing multiple documents, the same `Engine` instance can and should be used.

The 3-Heights™ PDF OCR API is fully thread-safe. So multiple documents can be processed concurrently, as longs as each instance of `Engine` and `Document` is used in one thread at the time only.

Note that some OCR engines are not thread-safe. This is one of the reasons, why the use of an OCR Service is recommended.

4.4 Thread safety

The 3-Heights™ PDF OCR API is fully thread-safe with one rule:

An object may only be accessed in one thread concurrently.

4.5 Garbage collection and closing objects

Every interface object is considered being a resource that must be closed when not needed anymore.

In garbage collected languages, interface objects are

In garbage collected languages such as Java or C#, interface objects can be closed explicitly with the `close()` or `Dispose()` method respectively, or implicitly by the garbage collector.

Whenever possible, objects should be closed explicitly. Not closing objects will increase resource consumption.

5 Programming Interfaces

Where possible and useful the 3-Heights™ PDF OCR API uses language specific features. This means that some parts of the API use different syntax for different programming languages.

5.1 .NET Interface

5.1.1 Error handling

Errors are reported using exceptions.

The three logic error codes are mapped to the corresponding native exception classes:

`IllegalArgument` maps to `System.ArgumentException`

`IllegalState` maps to `System.InvalidOperationException`

`UnsupportedOperation` maps to `System.NotSupportedException`

The rest of the error codes is modeled using a single exception class `PdfTools.ErrorCodeException` that provides access to the underlying error code and message.

5.1.2 Streams

The native stream interface `System.IO.Stream` is used.

5.1.3 Lists

Lists implement the native list interface `System.Collections.Generic.IList<T>`.

5.2 Java Interface

5.2.1 Properties

Properties are modeled with setter and getter methods.

5.2.2 Error handling

Errors are reported using exceptions.

The three logic error codes are mapped to the corresponding native runtime exception classes and are not checked:

`IllegalArgument` maps to `java.lang.IllegalArgumentException`

`IllegalState` maps to `java.lang.IllegalStateException`

`UnsupportedOperation` maps to `java.lang.UnsupportedOperationException`

The rest of the error codes is modeled using a single checked exception class `com.pdf_tools.ErrorCodeException` that provides access to the underlying error code and message.

5.2.3 Streams

The native stream interfaces cannot be used, because they are lacking two important features:

- The PDF file format is based on random access. Native Java streams have only limited support for this.
- The ability to read from an output stream is crucial for processing large files.

Instead we provide a custom stream interface `com.pdf_tools.Stream`, which has a similar interface as `java.io.RandomAccessFile`.

An implementation for files is provided, backed by `java.io.RandomAccessFile`.

5.2.4 Lists

Lists implement the native Java list interface `java.util.List`.

5.3 C Interface

5.3.1 Namespaces, classes and methods

In most languages, namespaces and classes are used to model the interfaces.

The exception is C, where this is modeled with function prefixes and functions operating on handles. The prefix of all functions of the 3-Heights™ PDF OCR API is `PdfOcr`, for types it is `TPdfOcr` and for enum values `ePdfOcr`.

5.3.2 Library Initialization

The first method called must be `PdfOcrInitialize`. Failing to invoke this function will result in undefined behavior. Similarly, the last method must be `PdfOcrUninitialize`.

5.3.3 Objects

Objects in the C interface are represented by object handles. After use, all object handles returned by the 3-Heights™ PDF OCR API must be closed with `PdfOcrClose` and released with `PdfOcrRelease`.

5.3.4 Properties

Properties are modeled with setter and getter methods.

5.3.5 Error handling

Errors are reported by returning special values, e.g. `NULL` or `FALSE`.

More information on the error can be retrieved by using the functions `PdfOcrGetLastError` and `PdfOcrGetLastErrorMessage`.

5.3.6 Strings

All functions involving strings are provided in two different flavors:

- UTF-16 function with suffix `W`, using `WCHAR` as parameter type.

- Multibyte character set function with suffix A, using char as parameter type. The concrete character set that is used depends on the platform:
  - On Windows, the current ANSI code page (CP_ACP) is assumed.
  - On Unix, the current C encoding (LC_CTYPE) is used.

In addition to the effective function names with suffix, there's a macro without suffix for each function pair: It either resolves to the W variant (if _UNICODE is defined), or to the A variant (if _UNICODE is **not** defined).

Example: Simplified signature of an API function:

```
void PdfOcrSetStringA(const char* szString);  // Multibyte encoding
void PdfOcrSetStringW(const WCHAR* szString); // UTF-16
#ifdef _UNICODE
#define PdfOcrSetString PdfOcrSetStringW
#else
#define PdfOcrSetString PdfOcrSetStringA
#endif
```

String return values

Functions that return a string are treated specially in C. Instead of returning the string, those functions take a buffer and size as last parameters and write into that buffer. The return value is the amount of data written to the buffer.

To determine the required buffer size, the function has to be called with NULL as argument.

Example: String return value (error handling is omitted)

```
size_t PdfOcrGetStringA(char* pBuffer, size_t nBufferSize);
```

```
size_t nBufferSize = PdfOcrGetStringA(NULL, 0);
char* pBuffer = malloc(nBufferSize * sizeof char);
nBufferSize = PdfOcrGetStringA(pBuffer, nBufferSize);
```

5.3.7 Streams

Streams are modeled by means of a set of callbacks and a context pointer, grouped in a struct TPdfStreamDescriptor.

An implementation for FILE* is provided in the header file pdfdecl.h (search for function PdfCreateFILEStreamDescriptor).

5.3.8 Lists

Lists are implemented like any other interfaces. Every list type provides a subset of the following properties and methods:

Count

Property (get): int Count

The number of elements of the list.

Get

Method: `ElementType Get(int index)`
Error codes: `IllegalState`, `IllegalArgument`, `UnsupportedOperation`

Get an element of the list.

Append

Method: `Append(ElementType element)`
Error codes: `IllegalState`, `IllegalArgument`, `UnsupportedOperation`

Append an element to the list.

6 Interface Reference

> **Note:** This chapter describes the C# interface of the 3-Heights™ PDF OCR API. Other interfaces however work similarly, i.e. they have calls with similar names and the call sequence to be used is the same as with C#.
> See chapters Java Interface and C Interface for more information.

6.1 Common Elements

> **Note:** In this section common static methods and properties are listed. They can be called in every interface.

6.1.1 CheckLicense

Method: `void CheckLicense()`
Static
Error code: `License`

Check if the product is properly licensed.

This method can be used to perform a license checky without actually opening or creating a document, e.g. when starting a GUI application or a service.

Error Code:

`License` The product is not properly licensed.

6.1.2 LicenseKey

Property (set): `String LicenseKey`
Static
Error code (set): `License`

Set the license key.

> **Note:** License keys that require activation can only be installed in the license manager. Setting them at runtime is not supported.

Error Code:

`License` The license key is not valid.

6.1.3 ProductVersion

Property (get): `String ProductVersion`
Static

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

6.2 Engine Interface

Typically when processing documents, an OCR engine is required. They can be created using the method `Create`.

OCR engines can be reused to process multiple files. However, one OCR engine can only be used to process one file at the time.

Note that some OCR engines must be disposed in the same thread where they have been created.

Note that of some OCR engines only one instance can be created per process.

6.2.1 Create

Method: `Engine Create(String name)`
Static
Error code: `Infrastructure`

Create a new OCR engine object.

The list of available OCR engines can be retrieved using `Engines`.

Optionally the `name` argument may be followed by `@` and engine creation parameters, e.g. `"service@http://localhost:7982/"`.

Parameter:

`name` `[String]` The engine name and optional parameters.

Returns:

The newly created engine instance.

Error Code:

`Infrastructure` If the `name` argument is invalid.

6.2.2 Engines

Property (get): `Engine[] Engines`
Static

Get the list the names of all available OCR engines.

6.2.3 SetLanguages

Method: `void SetLanguages(String languages)`
Error code: `Infrastructure`

Set OCR engine specific language settings, e.g. "German,English". This can be used to improve detection accuracy.

Note that for some engines it is crucial to set the used languages correctly. For example, Abbyy FineReader 11 will only detect characters used in these languages. So if `languages` is set to "English" only, umlauts such as "äöü" are reconigzed as "aou".

Error Code:

`Infrastructure` If the `languages` argument is invalid.

6.2.4 SetParameters

Method: `void SetParameters(String parameters)`
Error code: `Infrastructure`

Set OCR engine specific parameters.

OCR engine specific parameters (key/value pairs) can be set to optimize the performance or activate optional recognition features.

Error Code:

`Infrastructure` If the `parameters` argument is invalid.

6.2.5 RemainingPageCredits

Property (get): `int RemainingPageCredits`

Get the number of remaining page credits.

This is relevant for OCR engines used with a license that limits the number of pages. In all other cases, a count of 2147483647 or higher will be reported.

6.3 Document Interface

6.3.1 Open

Method: `Document Open(Stream stream, String password)`
Static
Error codes: `Password`, `IO`, `Corrupt`, `Conformance`, `IllegalArgument`

Open a PDF document.

Parameters:

`stream` `[Stream]` The stream where the PDF document is stored.

Random read access is required.

`password` `[String]` The password to open the PDF document.

Returns:

The newly created document instance.

Error Codes:

`Password` The file is encrypted and the `password` is not valid.

`IO` Error reading from the stream.

`Corrupt` The file is corrupt or not a PDF document.

`Conformance` The document's conformance is not supported.

`IllegalArgument` If the `stream` argument is `null`.

6.3.2 Process

Method: `Warning[] Process(Stream stream, EncryptionParams encryption, OcrParams ocr, ImageOcrParams imageOcr, TextOcrParams textOcr, PageOcrParams pageOcr, BarcodeParams barcodes)`
Error codes: `IO`, `IO`, `Conformance`, `Infrastructure`, `Processing`, `IllegalState`, `IllegalArgument`

Process the document.

See chapter Process Description for more information on how documents are processed.

Parameters:

`stream` `[Stream]` The stream to which the output is written. `stream` must support both random read and write access.

`encryption` `[EncryptionParams]` Optional encryption parameters.

`ocr` `[OcrParams]` Optional OCR parameters.

`imageOcr` `[ImageOcrParams]` Optional image OCR parameters.

`textOcr` `[TextOcrParams]` Optional text OCR parameters.

`pageOcr` `[PageOcrParams]` Optional page OCR parameters.

`barcodes` `[BarcodeParams]` Optional parameter specifying how recognized barcodes should be processed.

This parameter has only an effect, if pages containing barcodes are processed and detected by the OCR engine. This depends on the type and settings of the engine in `ocr` as well as the parameters `imageOcr`, `textOcr`, and `pageOcr`.

Returns:

A list of `Warning` objects that occurred during processing of the document.

Error Codes:

`IO` Parameter XmlOutput required but not valid.

`Conformance` If encryption parameters are set for a PDF/A file.

`Infrastructure` If the OCR engine is not operational, e.g. due to a license issue or TCP connection problems.

`Processing` If the document cannot be processed.

`IllegalState` If the document has already been closed.

`IllegalArgument` If an OCR engine is required but no valid one is specified.

6.4 `Warning` Interface

A warning object describes an event that occurred in `Process`. Warnings are non-critical in a way, that the operation must not be aborted by the 3-Heights™ PDF OCR API. Nonetheless, it is recommended to review warnings that occur and decide, which must be treated as critical errors for a particular process.

6.4.1 Code

Property (get): `WarningCode Code`

The code describing the type of the warning.

6.4.2 Message

Property (get): `String Message`

The message describing the warning.

6.4.3 PageNo

Property (get): `int PageNo`

The number of the page on which the warning occurred.

6.5 Structures

6.5.1 BarcodeParams Struct

See chapter How to detect barcodes for an example of how to use barcode parameters.

`Mode` [`BarcodeMode`] The mode according to which barcodes are processed.

default: `BarcodeMode.None`

`XmlOutput` [`Stream`] The stream to which recognized barcodes are written. The format of the resulting barcodes XML file is documented in the XML schema `barcodes.xsd` located in the documentation folder of the 3-Heights™ PDF OCR API.

This property is required, if and only if the property `Mode` is `BarcodeMode.Extract`.

default: `null`

6.5.2 EncryptionParams Struct

`OwnerPassword` [`string`] The owner password of the document.

This password is also referred to as the author's password and grants full access to the document. Not only can the document be opened and read, it also allows for changing the document's security settings (access permission and passwords).

`UserPassword` [`string`] The user password of the document.

protects the document against unauthorized opening and reading. If a PDF document is protected by a user password, either the user or owner password must be provided to open and read the document. If a document has a user password, it must have an owner password as well. If no owner password is defined, the owner password is the same as the user password.

`UserPermissions` [`Permission`] The user permissions.

What operations in a PDF document are granted is controlled via these permission flags. In order to set permission flags, the PDF document must be encrypted and have an owner password. The owner password is required to initially set or later change the permission flags.

6.5.3 ImageOcrParams Struct

The image OCR parameters control under what conditions and how images should be processed.

> **Note:** The properties RotateScan and DeskewScan have an effect only, if:
> 1. The page is a scan and not born-digital.
> 2. The page is processed by the OCR engine, which depends on the ImageOcrMode set.
> 3. The required information is provided by the OCR engine, which depends on the type and settings of the engine.

RotateScan [bool] Whether scanned pages should be rotated according to the orientation detected by the OCR engine.

default: false

DeskewScan [bool] Whether scanned pages should be deskewed according to the angle detected by the OCR engine.

default: false

Mode [ImageOcrMode] The mode according to which images are processed.

See chapter Process Description for a description on how setting this property affects OCR processing.

default: ImageOcrMode.None

6.5.4 OcrParams Struct

Engine [Engine] The OCR engine (see Engine).

This parameter may not be null for most combinations of ImageOcrMode, TextOcrMode, and PageOcrMode.

default: null

OcrDPI [double] The default resolution in DPI used for OCR.

Each page's optimal OCR resolution is determined automatically, such that all images and text can be recognized. The default resolution is chosen, if it is within the range of optimal resolutions. The range of allowed resolutions can be chosen using OcrMinDPI and OcrMaxDPI.

The range should be in the range of resolutions supported by the OCR engine. Most OCR engines are optimized for resolutions around 300 DPI. Selecting a resolution that is too low will hinder the detection of small text. An excessively high resolution will reduce performance because of the higher resource requirements to render the page images and perform OCR on them.

If the optimal resolution of a page is not within the range, an OCR warning is generated.

default: 300

OcrMinDPI [double] The minimum resolution in DPI used for OCR.

default: 200

OcrMaxDPI [double] The maximum resolution in DPI used for OCR.

default: 400

ProcessEmbeddedFiles [bool] Whether embedded files should be processed recursively. Otherwise embedded files are copied as-is.

default: `false`

6.5.5 PageOcrParams Struct

Mode [PageOcrMode] The mode according to which pages are processed.

See chapter Process Description for a description on how setting this property affects OCR processing.

default: `PageOcrMode.None`

Tagging [TaggingMode] The mode according to which tagging information is processed.

default: `TaggingMode.None`

6.5.6 TextOcrParams Struct

Mode [TextOcrMode] The mode according to which text is processed.

See chapter Process Description for a description on how setting this property affects OCR processing.

default: `TextOcrMode.None`

6.6 Enumerations

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

6.6.1 BarcodeMode Enumeration

None Do not add barcode information.

Embed Embedded recognized barcodes into the document's XMP metadata.

Extract Write recognized barcodes to the stream specified by `XmlOutput`.

6.6.2 ImageOcrMode Enumeration

None Do not process images.

UpdateText Only process images that have no OCR text.

ReplaceText Process all images and remove existing OCR text.

RemoveText Remove existing OCR text.

IfNoText Process images only if document contains no text.

6.6.3 PageOcrMode Enumeration

None Do not process pages.

All Process all pages that are not empty.

`IfNoText` Process all pages that contain content but no text.

`AddResults` Do not trigger processing of pages. But if pages are OCR processed, e.g. due to another OCR mode, add results as OCR text to pages.

6.6.4 `Permission` Enumeration

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

`Print` Low resolution printing

`Modify` Changing the document

`Copy` Content copying or extraction

`Annotate` Annotations

`FillForms` Filling of form fields

`SupportDisabilities` Support for disabilities

`Assemble` Document assembly

`DigitalPrint` High resolution printing

6.6.5 `TaggingMode` Enumeration

`None` Do not add tagging information.

`Update` Update existing tagging information.

6.6.6 `TextOcrMode` Enumeration

`None` Do not process text.

`Update` Only process text that is not extractable.

For all characters that have no meaningful Unicode, OCR processing is used to determine the Unicode. This is the recommended mode to make text extractable.

Note that making text extractable requires many OCR operations. The reason is that of all characters multiple instances must be recognized, to deal with erroneous OCR recognitions.

`Replace` Process all text.

OCR is used to determine the Unicode of all characters, that is even if they seemingly have Unicode information. This is useful for documents that possibly contain wrong Unicode information. Wrong Unicode information is typically created by flawed PDF creators or to obfuscate text (i.e. to prevent copy-and-paste or search operations).

For documents that contain correct Unicode information, this mode produces the same result as the mode Update. The rare exceptions are special fonts for which the OCR engine produces wrong results, which might happen for some decorative or handwritten fonts. The main disadvantage of the mode Replace over Update is, that more OCR operations are required.

6.6.7 `WarningCode` Enumeration

`Ocr` The warning is related to OCR recognition.

`Tagging` The warning is related to tagging. It must be considered critical when tagging documents for accessibility as described in How to tag scans for accessibility (PDF/A level A). Note that this warning does not mean, that OCR text has not been added, but merely that there was an issue with tagging it.

`Text` The warning is related to making text extractable. This is critical when making text extractable as described in How to make text extractable.

7 Version History

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

7.1 Patches in Version 4.12

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

Patch 4.12.26.4

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

7.2 Changes in Version 4.12

- **Introduced** license features `Service`, `Ocr`, and `Barcode`.
- **Improved** image ocr mode to cache OCR text of images. Images that occur on multiple pages must be OCR processed once only.
- **New** specialized OCR plugin `"barcodes"` to recognize barcodes and QR codes without an additional OCR engine.
- **New** OCR plugin `"abbyy12"` for the ABBYY FineReader 12 engine.
- **Improved** memory consumption of product.
- **New** detecion of OCR text that is under images.
- **Improved** ocr result processing, notably the accuracy of associating OCR text to existing text on page.
- **New** HTTP proxy setting in the GUI license manager.
- **New** `PageOcrMode` `AddResults`.

8 Licensing, Copyright, and Contact

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

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

Licensing and Copyright

Contact

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