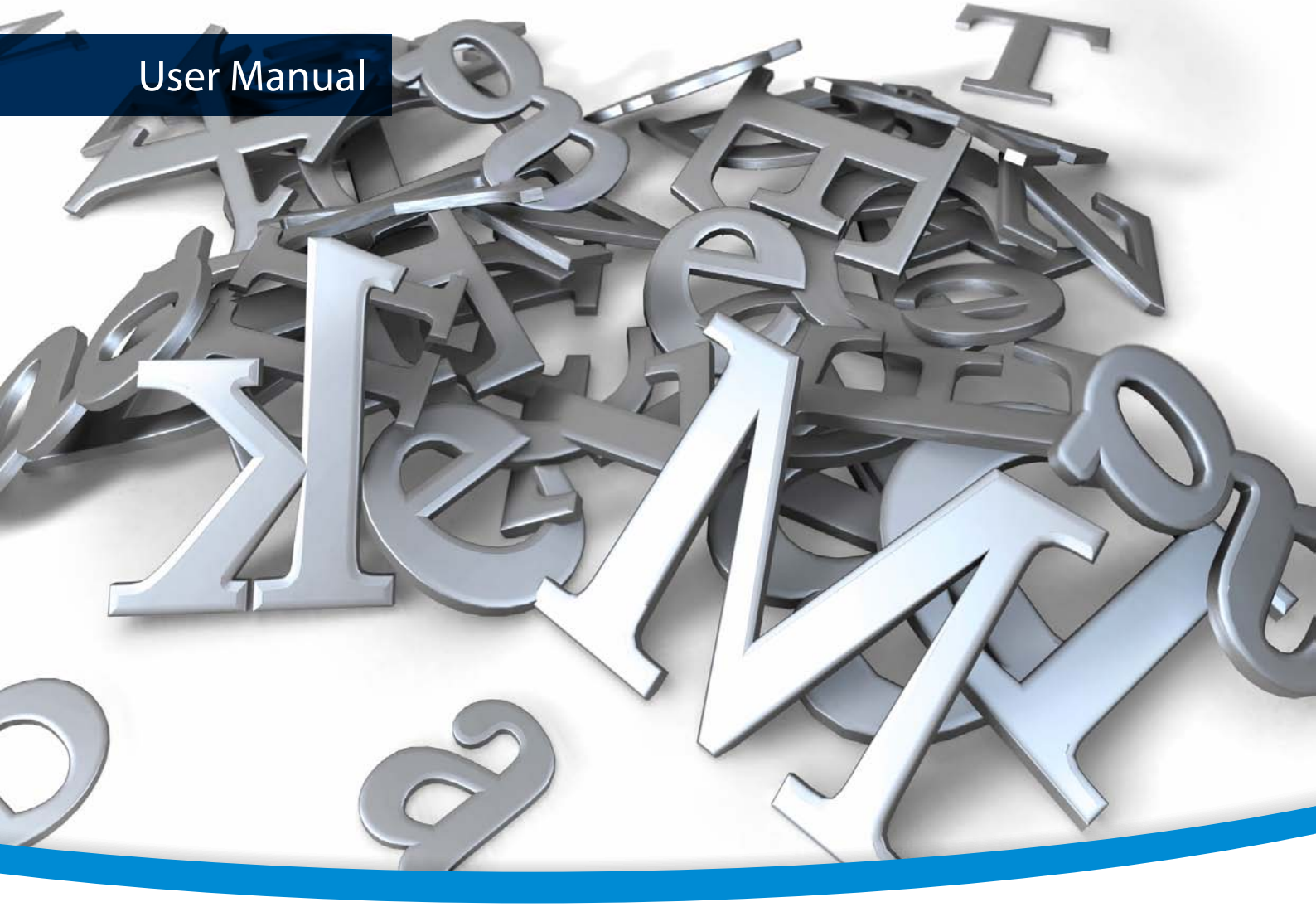
User Manual



# 3-Heights®
# PDF OCR Shell

PDF/A conform

**Version 6.27.8**

PDF-TOOLS.COM
Premium PDF Technology

# Contents

# 1 Introduction

## 1.1 Description

The 3-Heights® PDF OCR Shell 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
  - Conform to PDF/A
  - 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 Conformance

Standards:

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

## 1.3 Operating systems

The 3-Heights® PDF OCR Shell 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

'+' indicates the minimum supported version.

# 2 Installation

## 2.1 Windows

The 3-Heights® PDF OCR Shell comes as a ZIP archive or as an MSI installer.

To install the software, proceed as follows:

1. You need administrator rights to install this software.
2. Log in to your download account at https://www.pdf-tools.com. Select the product "PDF OCR Shell". 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.
   There is an MSI (*.msi) package and a ZIP (*.zip) archive available. The MSI (Microsoft Installer) package provides an installation routine that installs and uninstalls the product for you. The ZIP archive allows you to select and install everything manually.
   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 MSI installs the 64-bit version, whereas the ZIP archive contains both the 32-bit and the 64-bit version of the product. Therefore, on 32-bit systems, the ZIP archive must be used.
3. If you select an MSI package, start it and follow the steps in the installation routine.
4. 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:

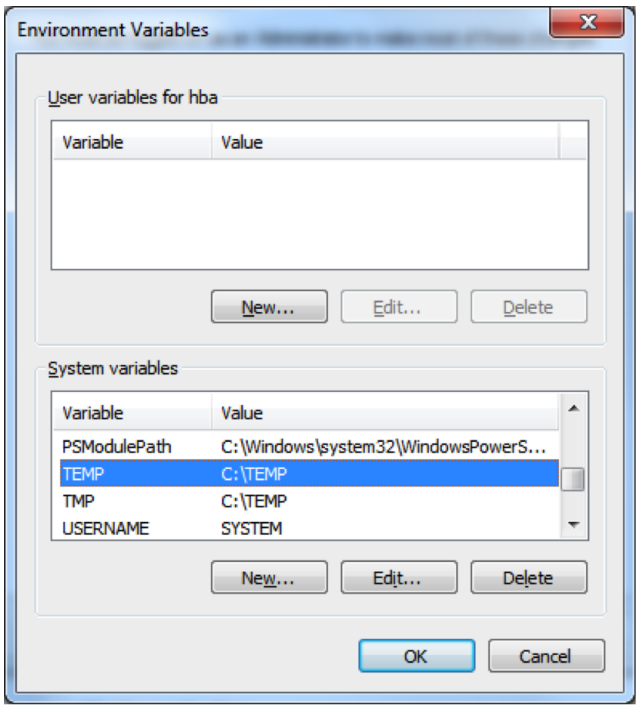
| Subdirectory | Description |
|---|---|
| bin | Runtime executable binaries |
| doc | Documentation |

5. (Optional) To easily use the 3-Heights® PDF OCR Shell from a shell, the directory needs to be included in the "Path" environment variable.
6. (Optional) Register your license key using the License management.
7. Make sure your platform meets the requirements regarding fonts described in Fonts.
8. 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.

### 2.1.1 How to set the environment variable "Path"

To set the environment variable "Path" in Windows, go to Start → Control Panel (classic view) → System → Advanced → Environment Variables.

Select "Path" and "Edit", then add the directory where pdfocr.exe is located to the "Path" variable. If the environment variable "Path" does not exist, create it.




## 2.2 Linux and macOS

This section describes installation steps required on Linux or macOS.

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

**File description**

| Name | Description |
|---|---|
| `bin/x64/pdfocr` | Main executable |
| `bin/x64/libPdfOcrAPI.so` | Shared library required by `pdfocr` |
| `bin/x64/*.ocr` | OCR plugin modules |
| `doc/*.*` | Documentation |

### 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 pdfocr
```

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 executable from one of the standard executable directories, e.g.

```
ln -s /opt/pdf-tools.com/bin/x64/pdfocr /usr/bin
```

4. Create a link to the shared library from one of the standard library directories, e.g.

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

5. Optionally, register your license key using the license manager.
6. Make sure your platform meets the requirements regarding fonts described in 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.

## 2.3 Uninstall

If you have used the MSI for the installation, go to Start → 3-Heights® PDF OCR Shell… → Uninstall …

If you have used the ZIP file for the installation, undo all the steps done during installation.

## 2.4 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® PDF OCR Shell is run under that user and the user profile is loaded.

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

### 2.4.1 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 Shell. 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.

### 2.4.2 Microsoft core fonts on Linux or macOS

Many PDF documents use Microsoft core fonts like Arial, Times New Roman, and other fonts commonly used on Windows. Therefore, it is recommended to install these fonts to your default font directories. Many Linux distribu-

tions offer an installable package for these “Microsoft TrueType core fonts”. For instance, on Debian based systems, the package is called `ttf-mscorefonts-installer`.

Alternatively, you can download the fonts from here:

https://corefonts.sourceforge.net/

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

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

## 2.5 Note about the evaluation license

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

## 2.6 Special directories

### 2.6.1 Directory for temporary files

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

### 2.6.2 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.

### 2.6.3 Font directories

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

If two fonts with the same name are found, the latter one takes precedence, i.e. user fonts 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 `pdfocr.exe` resides.

#### Linux

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

# 3 License management

The 3-Heights® PDF OCR Shell requires a valid license in order to run correctly. If no license key is set or the license is not valid, then the executable will fail and the return code is set to 10.

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

# 4 Getting started

## 4.1 Basics

### 4.1.1 Usage

The usage of the 3-Heights® PDF OCR Shell is:

`pdfocr [options] input.pdf output.pdf`

A simple command to process a document requires four parameters: The name of the PDF input file, the PDF output file, an OCR engine name, and a processing mode.

**Example:** Read the input document `input.pdf`, create a new document `output.pdf`, and use the OCR Service to process images.

```
pdfocr -ocr service -oim update input.pdf output.pdf
```

In order to list all available features type `pdfocr` without any parameters.

## 4.2 Specify the folder of the output file

The output folder can simply be added in front of the output file name.

```
pdfocr input.pdf myfolder\output.pdf
```

or absolute (Windows):

```
pdfocr input.pdf C:\myfolder\output.pdf
```

## 4.3 Process description

The following is a simplified process description of the 3-Heights® PDF OCR Shell:

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 (`-oim`), text (`-otm`), and pages (`-opm`) 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 barcode mode.
2. If its location is on an image on the page, it is processed according to the image ocr mode.
3. If it corresponds to text on the page, it is processed according to the text ocr mode.
4. Otherwise it is added as OCR text to the page, if the page ocr mode is not none.

## 4.4 Example

**Example:** Example that shows how to add OCR text to images.

```
pdfocr -ocr service -ocp "PredefinedProfile=DocumentConversion_Accuracy" -ocl "..." -oim
update input.pdf output.pdf
```

## 4.5 Use cases

This chapter describes some common use cases.

### 4.5.1 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 or 12

The following profile configuration `abbyy_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:

```
pdfocr -ocr service -ocp "Profile=C:\path\to\abbyy_text.ini" -ocl "..." ...
```

> **Note:** It is important that `C:\path\to\abbyy_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 `-oim`):

   ```
   pdfocr ... -oim update ...
   ```

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

   ```
   pdfocr ... -otm 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 `-opm`):

   ```
   pdfocr ... -opm ifNoText ...
   ```

### Process document

```
pdfocr -ocr service -ocp "Profile=C:\path\to\abbyy_text.ini" -ocl "..." -oim update ^
  input.pdf output.pdf
```

## 4.5.2 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 Shell 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 or 12

The following profile configuration `abbyy_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:

```
pdfocr -ocr service -ocp "Profile=C:\path\to\abbyy_tagging.ini" -ocl "..." ...
```

> **Note:** It is important that `C:\path\to\abbyy_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:

   ```
   pdfocr ... -oim update ...
   ```

2. (Optional) Enable scan enhancements:

   ```
   pdfocr ... -oca -occs ...
   ```

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

   ```
   pdfocr ... -tm update ...
   ```

## Process document

```
pdfocr -v -ocr service -ocp "Profile=C:\path\to\abbyy_tagging.ini" -ocl "..." -oim update
-oca -occs -tm update input.pdf output.pdf
```

## Error handling

Tagging errors are classified as warnings by the 3-Heights® PDF OCR Shell. Because tagging is crucial for this process, tagging warnings returned must be checked and treated as errors. In case of OCR warnings, the return code of 3-Heights® PDF OCR Shell will be 7. See How to handle conversion warnings for more information.

## 4.5.3 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:

```
pdfocr -ocr barcodes -ocp "BarcodeTypes=QRCode" ...
```

#### Abbyy FineReader 11 or 12 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:

```
pdfocr -ocr service -ocp "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:

   ```
   pdfocr ... -opm all ...
   ```

2. **Enable barcode extraction**: Write all detected barcodes in XML format to the file `barcodes.xml`:

   ```
   pdfocr ... -obx barcodes.xml ...
   ```

   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 Shell.

### Process document

```
pdfocr -ocr barcodes -ocp "BarcodeTypes=QRCode" -opm all -obx barcodes.xml input.pdf
output.pdf
```

## 4.6 How to handle conversion warnings

There are two types of problems that may occur when processing a file.

Some problems are severe and hinder further processing. As a result, the processing is aborted and the return code of the 3-Heights® PDF OCR Shell will indicate the error.

Problems that do not hinder further processing are classified as warnings. In this case, the the return code of 3-Heights® PDF OCR Shell will be 7. Using the option -v, all warnings can be shown.

**Example:** The following command generates two warnings, because the input file is signed and the specified OCR resolution is too low for an optimal recognition.

```
pdfocr -ocr service -ocp "PredefinedProfile=DocumentConversion_Accuracy" -ocl "..." -oim
update -ocd 200 150 250 -v input.pdf output.pdf
Processing file input.pdf
OCR warnings:
 - ocr: Max OCR DPI is 250 but should be at least 300 for optimal image OCR. (page 1)
 - signed: The document signature of "Peter Pan" had to be removed. (page 1)
Done.
```

For some processes, certain warnings might be critical, e.g. as described in How to make text extractable. In order to simplify the processing of warnings, they are divided into categories.

The format of the warnings shown using -v is as follows:

```
- ‹category›: ‹message› [(page ‹number›)]
```

The following categories are defined:

**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.

**signed** Processing a signed file changes it, such that all signatures become invalid. Therefore, all signatures are removed and this warning is generated.

# 5 Interface reference

Switches are options that are provided with the command to define how the document should be processed.

Switches can occur in two forms: As stand-alone option, such as `-v` (verbose mode) or they may require a parameter, such as `-pw password` (set password to read encrypted input document).

The last two parameters of the command line should always be the input and the output document.

Switches are parsed from left to right. If the same switch is applied multiple times the last set value is applied.

## 5.1 OCR engine configuration

### 5.1.1 `-le` List OCR engines

**List OCR engines** `-le`

OCR engines are accessed through the corresponding OCR interface DLLs. The following 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 DLLs are provided by the 3-Heights® PDF OCR Shell. The OCR engine is provided as a separate product, such as 3-Heights® OCR Enterprise Add-on.

Here is an example of listing available OCR engines:

```
pdfocr -le
List of available OCR engines:
- abbyy11
- abbyy10
- service
End of list.
```

In order to make use of the OCR engine, the OCR interface DLL and the OCR engine must be installed. The `-le` switch lists all available OCR interface DLLs. It does not verify the corresponding OCR engine is installed and can be initialized. The OCR engine is actually accessed when using the switch `-ocr`.

### 5.1.2 `-ocr` Load OCR engine

**Load OCR engine** `-ocr ‹name›`

If a PDF document has to be made fully text searchable, even if the text is part of a raster image, then the images that are contained in the PDF document must be run through an OCR engine. With this switch, the user can select an OCR engine, e.g. `Abbyy11`, and instruct the tool to embed the recognized text as a hidden layer on top of the

image. If the add-in is not found or the engine cannot be initialized (because it is not installed or the license key is not valid), then an error message is issued.

The name of the OCR engine can be retrieved using the switch `-le`. If the switch `-ocr` is not used, no OCR is applied.

**Example:** The following switch sets the OCR engine to the OCR Service

```
pdfocr -ocr service input.pdf output.pdf
```

See also documentation for the 3-Heights® OCR Add-on.

### 5.1.3 `-ocl` Set OCR language

**Set OCR language** `-ocl ‹languages›`

To optimize the performance of the OCR engine, it can be given hints what languages are used. The default language of the Abbyy FineReader 11 OCR Engine is English. This switch can only be used if the switch `-ocr` is set. This setting depends on the OCR engine.

The following switch set the languages to English and German:

```
pdfocr -ocr abbyy11 -ocl "English, German" input.pdf output.pdf
```

See also documentation for the 3-Heights® OCR Add-on.

### 5.1.4 `-ocp` Set OCR parameters

**Set OCR parameters** `-ocp ‹params›`

Using this switch, OCR engine specific parameters (key/value pairs) can be set to optimize the performance.

The following switch sets a predefined profile (i.e. a configuration setting), which is optimized for creating electronic archives with high accuracy:

```
pdfocr -ocr abbyy11 -ocp "PredefinedProfile = DocumentArchiving_Accuracy"
input.pdf output.pdf
```

See also documentation for the 3-Heights® OCR Add-on.

## 5.2 Image OCR options

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

**Note:** The options `-oca` and `-occs` 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 `-oim` set.
3. The required information is provided by the OCR engine, which depends on the type and settings of the engine.

### 5.2.1 `-oim` Image OCR mode

**Image OCR mode** `-oim ‹mode›`

The mode according to which images are processed. See chapter Process description for a description on how setting this option affects OCR processing.

Available values for `‹mode›` are:

**`none`** **(default)** Do not process images.

**`update`** Only process images that have no OCR text.

**`replace`** Process all images and remove existing OCR text.

**`remove`** Remove existing OCR text.

**`ifNoText`** Process images only if document contains no text.

### 5.2.2 `-oca` Rotate scan

**Rotate scan** `-oca`

Rotate scan according to the orientation detected by the OCR engine.

### 5.2.3 `-occs` Deskew scan

**Deskew scan** `-occs`

Deskew scan according to the angle detected by the OCR engine.

## 5.3 Text OCR options

### 5.3.1 `-otm` Text OCR mode

**Text OCR mode** `-otm ‹mode›`

The mode according to which text is processed. See chapter Process description for a description on how setting this option affects OCR processing.

Available values for `‹mode›` are:

**`none`** **(default)** 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.

### 5.3.2 -ots Text OCR skip

**Text OCR skip** `-ots ‹list›`

Defines text that can be skipped from text OCR processing.

The value for `‹list›` is a comma-separated list of the following values:

**none** **(default)** Do not skip any text in text ocr processing.

**knownSymbolic** Skip text of all fonts that are known to be symbolic, e.g. “ZapfDingbats” or “Wingdings”.

For many symbols of these fonts there exist no Unicodes. Also, even the ones that have Unicodes, such as “✓” or “→”, cannot be recognized by most OCR engines. Therefore, OCR processing of these fonts usually does not produce a meaningful result and could be skipped.

**pua** Skip text with Unicodes from Private Use Areas (PUA), i.e. accept Unicodes from PUA as meaningful.

Unicodes from the PUA are typically used for symbols, for which no Unicodes exist. OCR processing of these symbols does not produce a result and could be skipped.

On the other hand, some bad PDF creators use PUA for normal text. For these cases, OCR processing should be performed.

### 5.3.3 -otu ToUnicode source

**ToUnicode source** `-otu ‹list›`

Defines additional ToUnicode sources, i.e. in addition to OCR processing.

The value for `‹list›` is a comma-separated list of the following values:

**none (default)** Do not use any additional sources. Only use ToUnicode information contained in the PDF document as described in the PDF Reference.

**knownSymbolicPua** Use Unicodes from Private Use Areas (PUA) for all fonts that are known to be symbolic, e.g. “ZapfDingbats” or “Wingdings”.

**fallbackAllPua** Use Unicodes from Private Use Areas (PUA) for all characters for which no better Unicode can be determined.

**`installedFont`** If on the system a font of the same name is installed, use Unicodes of matching glyphs.

## 5.4 Page OCR options

### 5.4.1 `-opm` Page OCR mode

**Page OCR mode** `-opm ‹mode›`

The mode according to which pages are processed. See chapter Process description for a description on how setting this option affects OCR processing.

Available values for `‹mode›` are:

**`none`** **(default)** 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.

### 5.4.2 `-tm` Tagging mode

**Tagging mode** `-tm ‹mode›`

The mode according to which tagging information is processed.

Available values for `‹mode›` are:

**`none`** Do not add tagging information.

**`update`** Update existing tagging information. A warning is generated, if no tagging information can be added. Therefore, this value is recommended if tagging information is crucial to your process.

**`auto`** **(default)** Determine tagging mode automatically. Use `update` for scans and born-digital documents with tagging, and `none` otherwise.

## 5.5 Barcode options

### 5.5.1 `-obx` Process recognized barcodes

**Process recognized barcodes** `-obx ‹stream›`

The mode according to which barcodes are processed.

`‹stream›` may be either the string `xmp`, in which case recognized barcodes are embedded into the document's XMP metadata, or a file name, in which case recognized barcodes are written to the file in XML format. 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 Shell.

## 5.6 General options

### 5.6.1 -ocd OCR resolution

**OCR resolution** `-ocd ‹d› ‹dmin› ‹dmax›`

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

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: `-ocd 300 200 400`

### 5.6.2 -pef Process embedded files

**Process embedded files** `-pef`

Process embedded files recursively. Otherwise embedded files are copied as-is.

### 5.6.3 -pw Read an encrypted PDF file

**Read an encrypted PDF file** `-pw ‹password›`

A PDF document that has a user password (the password to open the document) can only be processed when either the user or the owner password is provided. The password can be provided using the option `-pw` followed by the password.

**Example:** The input PDF document is encrypted with a user password. Either the user or the owner password of the input PDF is "mypassword". The command to process such an encrypted file is:

```
pdfocr -pw mypassword input.pdf output.pdf
```

When a PDF is encrypted with a user password and the password is not provided or is incorrect, the 3-Heights® PDF OCR Shell cannot read and process the file. Instead it generates the following error message:

```
Password wasn't correct.
```

### 5.6.4 -v Verbose mode

**Verbose mode** `-v`

This option turns on the verbose mode.

### 5.6.5 `-lk` Set license key

**Set license key** `-lk ‹key›`

Pass a license key to the application at runtime, instead of using one that is installed on the system.

```
pdfocr -lk X-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX ...
```

This is required in an OEM scenario only.

## 5.7 Frequent error source

It may happen that you type a command, or copy it from somewhere and it doesn't work even though it seems to be correct. A common reason is that the dash (-) which is used for most parameters is accidently mistaken by an em dash (—).

## 5.8 Return codes

All return codes other than `0` indicate an error in the processing.

In case of an error, an error message is written to the standard error stream (stderr). It is highly recommended to check the command's return value and in case of an error, log the error output.

**Return codes**

| Value | Description |
|---|---|
| `0` | Success. |
| 1 | Couldn't open input file. |
| 2 | PDF output file could not be created. |
| 3 | Error with given options, e.g. too many parameters. |
| 4 | PDF input file is encrypted and password is missing or incorrect. |
| 5 | OCR infrastructure error.<br>The file could not be processed because of an error in the OCR infrastructure, e.g. page credits of OCR engine used. |
| 6 | OCR processing error.<br>The file could not be processed because of an error related to the input file or the processing options set. |
| 7 | OCR warnings occurred.<br>I.e. the file has been processed successfully, but warnings occurred. Use `-v` to get a list of them. If the warnings reported are not relevant for your process, the return code may be ignored and treated the same as `0`. See How to handle conversion warnings for more information. |

### Return codes

| | |
|---|---|
| `10` | License error, e.g. invalid license key. |

# 6 Version history

## 6.1 Changes in versions 6.19–6.27

- **Update** license agreement to version 2.9

## 6.2 Changes in versions 6.13–6.18

No functional changes.

## 6.3 Changes in versions 6.1–6.12

- **New** version of OCR plugin `"barcodes"` with QR Code recognition improvements.
- **Improved** search algorithm for installed fonts: User fonts under Windows are now also taken into account.
- **New** option `-ots` to define text that can be skipped from text OCR processing.
- **New** option `-otu` to enable additional sources of ToUnicode information, i.e. in addition to OCR processing.
- **Changed** default of tagging mode option `-tm` to `auto`.

## 6.4 Changes in version 5

- **Improved** wording of warning messages. The new messages do not contain the words “warning” nor “error”, such that they can be used as both warning and error message.
- **New** warning, if signatures were removed.
- **New** version of OCR plugin `"barcodes"` with QR Code recognition improvements.
- **New** additional supported operating system: Windows Server 2019.
- **Improved** option `-obx` to extract the type of recognized barcodes.

## 6.5 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** parameter `addResults` for page ocr mode `-opm`.

# 7 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.



### Contact

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