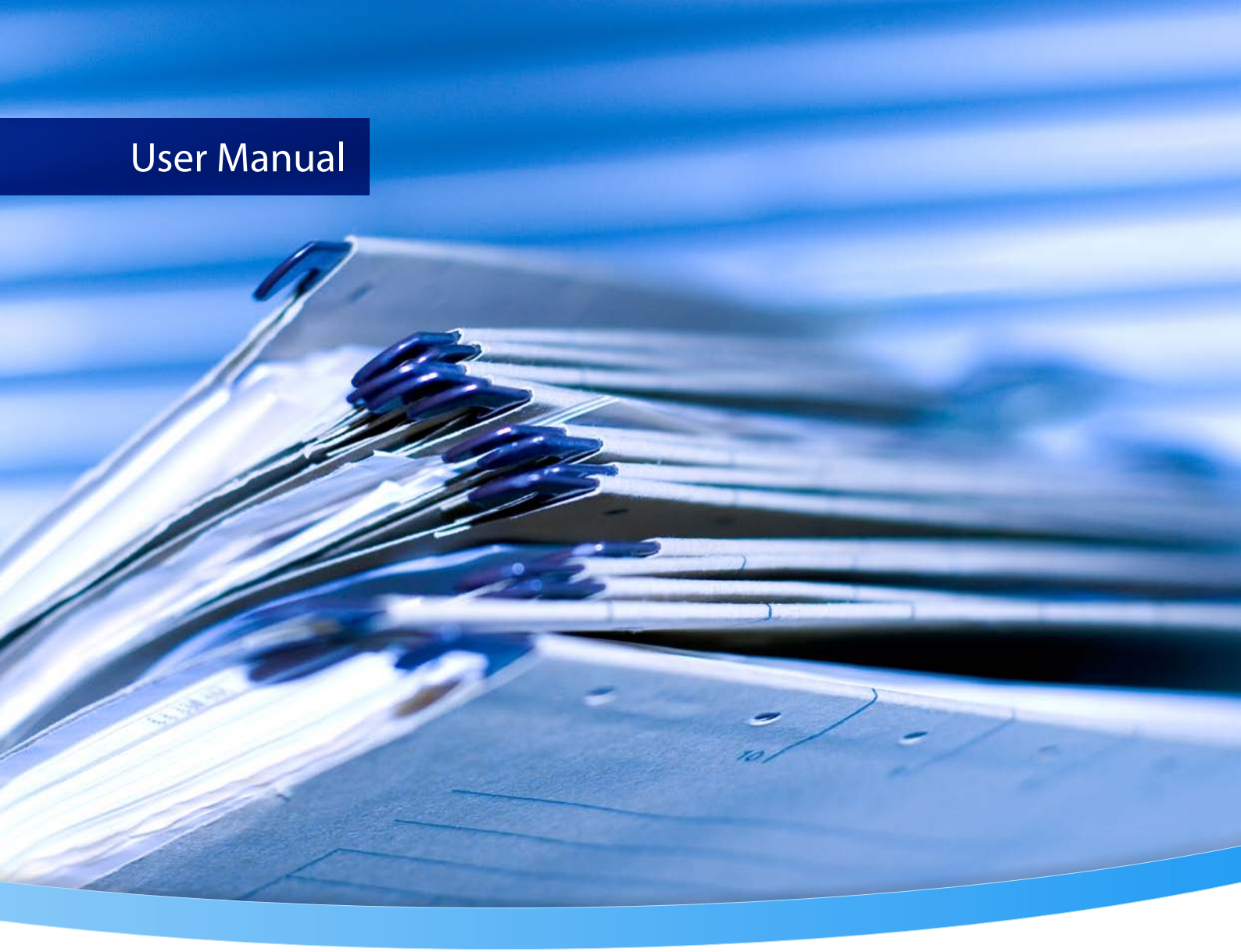
User Manual



3-Heights®
PDF to PDF/A Converter Shell

PDF/A conform

Version 6.27.1

pdftools

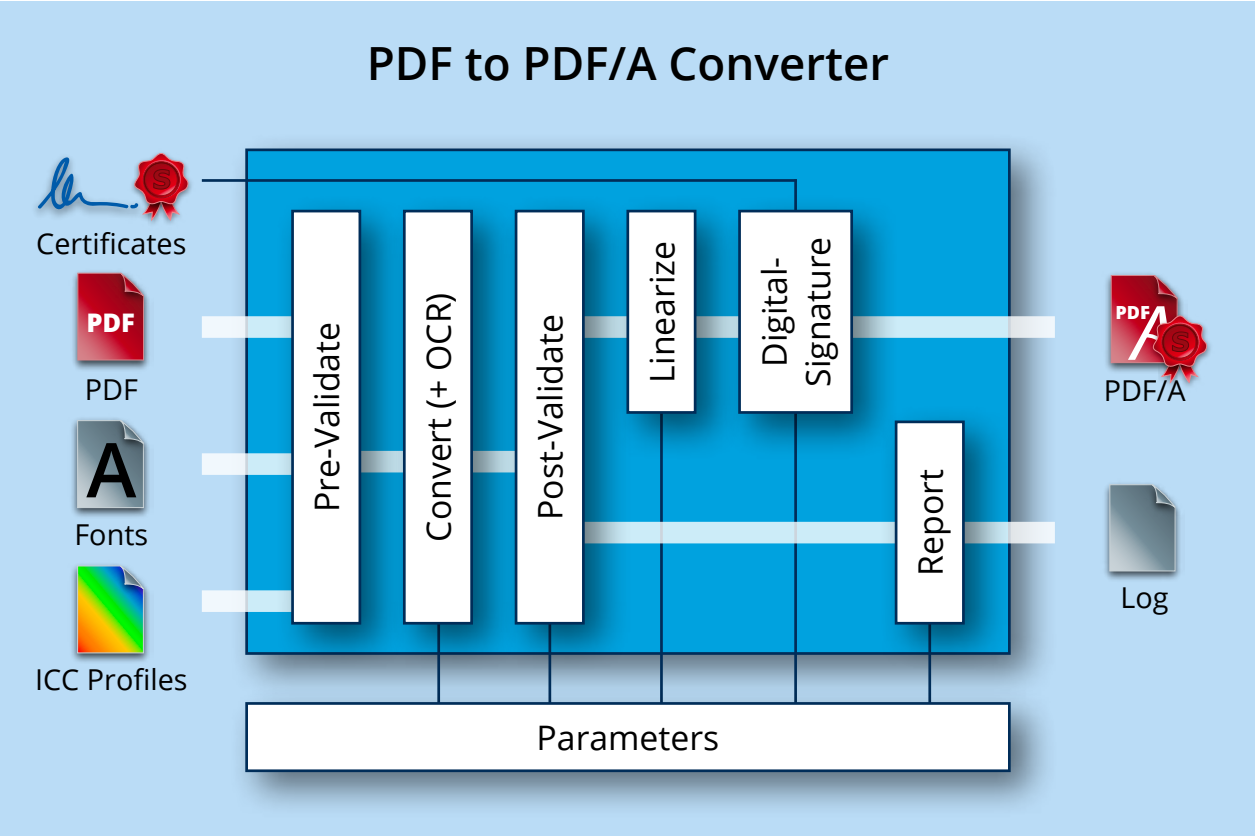
Contents

1 Introduction

1.1 Description

The 3-Heights® PDF to PDF/A Converter Shell converts PDF files into PDF/A files. PDF/A has been acknowledged world-wide as the ISO standard for long-term archiving since 2005. The tool analyzes and converts the input file, applying a digital signature where required.

The integrated validator then optionally checks conformity once again. This product is robust and powerful, and therefore designed for archive migrations of any size.



1.2 Functions

The 3-Heights® PDF to PDF/A Converter Shell accepts files from many different applications and automatically converts them into PDF/A. The level of conformity can be set to level A, U, or B. ICC color profiles for device-dependent color profiles and font types are embedded in the document. There is an option to provide the entire character set for fonts (no subsetting) to facilitate editing at a later stage. Missing fonts are reproduced as close to the original as possible via font recognition. Metadata can be generated automatically or added from external sources. The tool also detects and automatically repairs problems typical of the PDF format. A digital signature can be applied and a conformity check carried out at the end of the process. The optional 3-Heights® OCR Add-on and linearization for fast web display are valuable additional functions.

1.2.1 Features

- Convert PDF documents to PDF/A-1, PDF/A-2, PDF/A-3

    - Provide support for all PDF/A conformance levels.
    - Make color spaces device-independent, e.g. by embedding ICC profile or setting an output intent
    - Embed and subset fonts
    - Manage colorants (PDF/A-2 and later)
    - Recover corrupt documents
    - Repair corrupt data such as embedded font programs or images
    - Remove transparency (PDF/A-1 only)
    - Remove malicious content such as attached files (PDF/A-1 and PDF/A-2) and JavaScript actions
    - Remove multimedia content such as video and sound
    - Convert embedded and attached files (PDF/A-2 and later)
    - Repair metadata and make them consistent
- Control the conversion process
    - Perform pre- and post-validation
    - Get conversion reports
    - Write the application log to a log file
    - Automatically determine optimal conformance based on input file (optional)
    - Enables sophisticated error handling
- Create digital signatures, conforming to PDF/A
    - Apply PAdES-LTV (Long-Term Validation) signatures
    - Include embedded trust chain, timestamp, and revocation information (OCSP, CRL)
    - Supports various types of cryptographic providers:
        - Windows certificate store
        - Hardware such as hardware security module (HSM), smart cards, and USB tokens
        - Online signature services
            - myBica Digital Signing Service
            - Swisscom All-in Signing Service
            - GlobalSign Digital Signing Service
            - QuoVadis sealsign
    - Add an optional visual appearance of the signature (page, size, color, position, text, background image, etc.)
- Read encrypted input files
- Enhance output file
    - Set metadata
    - Perform linearization for fast web view
    - Use PDF file compression features (PDF/A-2 and later)
- Perform text recognition using OCR engine (optional)
    - Replace old OCR text or skip images with existing OCR text
    - Set the OCR language and options
    - Deskew and de-noise images
    - Detect barcodes
    - List OCR plugins
- Add embedded files (PDF/A-2) and associated files (PDF/A-3)
- Add embedded XML invoice data conforming to the ZUGFeRD or Factur-X specification (PDF/A-3)

1.2.2 Formats

Input formats

- PDF 1.x (PDF 1.0, …, PDF 1.7)
- PDF 2.0

Output formats

- PDF/A-1a, PDF/A-1b
- PDF/A-2a, PDF/A-2b, PDF/A-2u
- PDF/A-3a, PDF/A-3b, PDF/A-3u

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)
  - PAdES (ETSI EN 319 142) signature levels B-B, B-T, CMS
  - Legacy PAdES baseline signature (ETSI TS 103 172) B-Level and T-Level
  - Legacy PAdES (ETSI TS 102 778) Part 2 (PAdES Basic), Part 3 (PAdES-BES), and Part 4 (PAdES-LTV, Long-Term Validation)
  - Long-term signature profiles for PAdES (ISO 14533-3)
  - Cryptographic suites (ETSI TS 119 312)
  - ZUGFeRD 1.0, ZUGFeRD 2.0, Factur-X V1.0
- Quality assurance: veraPDF test corpus and Isartor test suite

1.3 Operating systems

The 3-Heights® PDF to PDF/A Converter 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
- macOS 10.10+ | x64

'+' indicates the minimum supported version.

1.4 Digital signatures

1.4.1 Overview

Digital signature is a large and slightly complex topic. This chapter gives an introduction to digital signatures and describes how the 3-Heights® PDF to PDF/A Converter Shell is used to apply them. It does however not describe all the technical details.

1.4.2 Terminology

Digital signature is a cryptographic technique of calculating a number (a digital signature) for a message. Creating a digital signature requires a private key from a certificate. Validating a digital signature and its authorship requires a public key. Digital Signature is a technical term.

Electronic signature is a set of electronic data that is merged or linked to other electronic data in order to authenticate it. Electronic Signatures can be created by means of a digital signature or other techniques. Electronic Signature is a legal term.

Abbreviations

CA	Certification Authority
CMS	Cryptographic Message Syntax
CRL	Certificate Revocation List
CSP	Cryptographic Service Provider
HSM	Hardware Security Module
OCSP	Online Certificate Status Protocol
PKCS	Public Key Cryptography Standards
QES	Qualified electronic signature
TSA	Timestamp Authority
TSP	Timestamp Protocol

1.4.3 Why digitally signing?

The idea of applying a digital signature in PDF is very similar to a handwritten signature: A person reads a document and signs it with its name. In addition to the name, the signature can contain further optional information, such as the date and location. A valid electronic signature is a section of data that can be used to:

- Ensure the integrity of the document
- Authenticate the signer of the document
- Prove existence of file prior to date (timestamp)

Digitally signing a document requires a certificate and its private key. How to access and use a certificate is described in Cryptographic provider.

In a PDF document, a digital signature consists of two parts:

A PDF related part This part consists of the PDF objects required to embed the signature into the PDF document. This part depends on the signature type (document signature, MDP signature - see explanation). Information such as name of the signer, reason, date, and location is stored here. The signature may optionally have a visual appearance on a page of the PDF document, which can contain text, graphics, and images.

This part of the signature is entirely created by the 3-Heights® PDF to PDF/A Converter Shell.

A cryptographic part A digital signature is based on a cryptographic checksum (hash value) calculated from the content of the document that is being signed. If the document is modified at a later time, the computed hash value is no longer correct and the signature becomes invalid, i.e. the validation fails and reports that the document has been modified since the signature was applied. Only the owner of the certificate and its private key is able to sign the document. However, anybody can verify the signature with the public key contained in the certificate.

This part of the signature requires a cryptographic provider for some cryptographic data and algorithms.

The 3-Heights® PDF to PDF/A Converter Shell supports the following types of digital signatures:

Document signature A document signature type digital signature checks the integrity of the signed part of the document and authenticates the signer's identity. One or more document signatures can be applied. A signed document can be modified and saved by incremental updates. The state of the document can be re-created as it existed at the time of signing.

MDP signature A modification detection and prevention signature detects disallowed changes specified by the author. A document can contain only one MDP signature, which must be the first in the document. Other types of signatures may be present.

Document timestamp signature A timestamp signature provides evidence that the document existed at a specific time and protects the document's integrity. One or more document timestamp signatures can be applied. A signed document can be modified and saved by incremental updates.

1.4.4 What is an electronic signature?

There are different types of electronic signatures, which normally are defined by national laws, and therefore are different for different countries. The type of electronic signatures required in a certain process is usually defined by national laws. Quite advanced in this manner are German-speaking countries where such laws and an established terminology exist. The English terminology is basically a translation from German.

Three types of electronic signatures are distinguished:

- Simple Electronic Signature - "Einfache Elektronische Signatur"
- Advanced electronic signature (AdES) - "Fortgeschrittene Elektronische Signatur"
- Qualified electronic signature (QES) - "Qualifizierte Elektronische Signatur"

All applied digital signatures conform to PDF/A and PAdES.

Simple electronic signature

A simple electronic signature requires any certificate that can be used for digital signing. The easiest way to retrieve a certificate, which meets that requirement, is to create a self-signed certificate. Self-signed means it is signed by its owner. Therefore, the issuer of the certificate and the approver of the legitimacy of a document signed by this certificate is the same person.

Example:

Anyone can create a self-signed certificate issued by "Peter Pan" and issued to "Peter Pan". Using this certificate, a person can sign in the name of "Peter Pan".

If a PDF document is signed with a simple electronic signature and the document is changed after the signature had been applied, the signature becomes invalid. However, the person who applied the changes could, at the same time (maliciously), also remove the existing simple electronic signature and—after the changes—apply a new, equally looking Simple Electronic Signature and falsify its date. A simple electronic signature is neither strong enough to ensure the integrity of the document nor to authenticate the signer.

This drawback can overcome using an advanced or qualified electronic signature.

Advanced electronic signature

Requirements for advanced certificates and signatures vary depending on the country where they are issued and used.

An advanced electronic signature is based on an advanced certificate that is issued by a recognized certificate authority (CA) for the country, such as VeriSign, SwissSign, QuoVadis. To receive an advanced certificate, its owner must prove its identity, e.g. by physically visiting the CA and presenting its passport. The owner can be an individual or legal person or entity.

An advanced certificate contains the name of the owner, the name of the CA, its period of validity, and other information.

The private key of the certificate is protected by a PIN, which is only known to its owner.

This brings the following advantages over a simple electronic signature:

- The signature authenticates the signer.
- The signature ensures the integrity of the signed content.

Qualified electronic signature

Requirements for qualified certificates and signatures vary depending on the country where they are issued and used.

A qualified electronic signature is similar to an advanced electronic signature, but has higher requirements. The main differences are:

- It is based on a qualified certificate, which is provided as a hardware token (USB stick, smart card).
- For every signature, it is required to enter the PIN code manually. This means that only one signature can be applied at a time.
- Certificate revocation information (OCSP/CRL) can be acquired from an online service. The response (valid, revoked, etc.) must be embedded in the signature.
- A timestamp (TSP) that is acquired from a trusted time server (TSA) may be required.

This brings the following advantages over an advanced electronic signature:

- The signature ensures the certificate was valid at the time when the document was signed (due to the embedding of the OCSP/CRL response).
- The signature ensures the integrity of the time of signing (due to the embedding of the timestamp).
- Legal processes that require a QES are supported.

> **Note:** A timestamp can be added to any type of signature. OCSP/CRL responses are also available for some advanced certificates.

1.4.5 Creating electronic signatures

This is a simple example of how to create an electronic document signature. More detailed examples can be found in Creating digital signatures.

Preparation steps

1. Identify whether an Advanced electronic signature or a Qualified electronic signature is required. For most automated processes, an advanced signature is sufficient.
2. Identify regulatory requirements regarding the content and life-cycle of the signature:
   - Is a timestamp required to prove that the signature itself existed at a certain date and time?
   - Should validation information be embedded to allow the signature to be validated long time after its generation?
   - Should the integrity of the validation material be protected?
   - Is a specific signature encoding required?

   These requirements (or regulatory requirements) define the signature level that must be used.
3. Acquire a corresponding certificate from a CA.
   For automated processes, it is recommended you use a HSM, an online signing service, or soft certificates. Other hardware such as USB tokens or smart cards are often cheaper, but limited to local interactive single-user applications.
   When using an online signing service, ensure that it supports the required signature encoding.
4. Set up and configure the certificate's Cryptographic provider.
   - In case the certificate resides on hardware such as an USB token or a smart card, the required middleware (driver) needs to be installed.
   - In case the certificate is a soft certificate, it must be imported into the certificate store of a cryptographic provider.
5. Optional: Acquire access to a trusted time server (TSA) (preferably from the CA of your signing certificate).
6. Optional: Ensure your input documents conform to the PDF/A standard.
   It is recommended to sign PDF/A documents only, because this ensures that the file's visual appearance is well defined, as it can be reproduced flawlessly and authentically in any environment. Furthermore, PDF/A conformance is typically required if the file is to be archived. Because signed files cannot be converted to PDF/A without breaking its signatures, files must be converted before signing.

> **Note:** A detailed guidance on the use of standards for signature creation can be found in the technical report ETSI TR 119 100.

Application of the signature

Apply the signature by providing the following information:

1. The Cryptographic provider where the certificate is located
2. Values for the selection of the signing certificate (e.g. the name of the certificate)
3. Optional: Timestamp service URL (e.g. "http://server.mydomain.com:80/tsa")
4. Optional: Timestamp service credentials (e.g. username:password)
5. Optional: Add validation information
6. Optional: Visual appearance of the signature on a page of the document (e.g. an image).

Example: Steps to add an electronic document signature

The 3-Heights® PDF to PDF/A Converter Shell applies PDF/A conforming signatures. This means if a PDF/A document is digitally signed, it retains PDF/A conformance.

To add an electronic document signature with the 3-Heights® PDF to PDF/A Converter Shell, the following steps need to be done:

1. Provide the certificate name (Subject).
2. Apply settings for the signature, such as the reason text, or the visual appearance (color, position, etc).
3. Process the PDF document by a user which has access to the selected certificate, and thereby, add the signature.

The certificate name is provided with the switch `-cn`, the reason with the switch `-cr`, and the provider (including the PIN to access the certificate's private key) with the switch `-cp`. A sample command looks like this:

```
pdf2pdf
-cn "Philip Renggli"
-cp "cvp11.dll;0;secret-pin"
-cr "I reviewed the document"
-tsu "http://server.mydomain.com:80/tsa"
-ar 10 10 200 50
input.pdf output.pdf
```

The visual appearance of the digital signature on a page of the resulting output document looks as shown below:

2 Installation

2.1 Windows

The 3-Heights® PDF to PDF/A Converter 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 to PDF/A Converter 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:

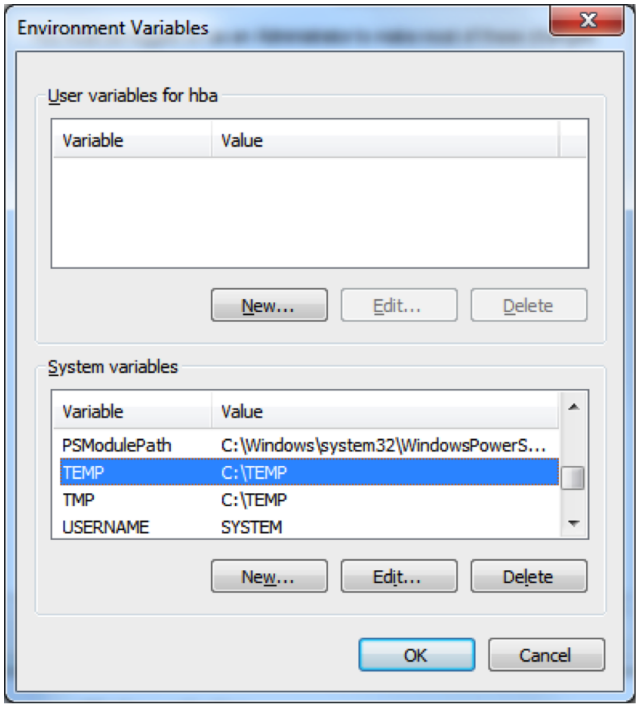
Subdirectory	Description
`bin`	Runtime executable binaries
`doc`	Documentation

5. (Optional) To easily use the 3-Heights® PDF to PDF/A Converter 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. Ensure the cache directory exists as described in Special directories.
8. Make sure your platform meets the requirements regarding color spaces and fonts described in Color spaces and Fonts, respectively.
9. If you want to sign documents, set up your cryptographic provider as described in Cryptographic provider.
10. (Optional) Download and install the 3-Heights® OCR Enterprise Add-on, and the OCR Engine as described in the respective manuals:
    - 3-Heights® OCR Add-on for ABBYY FineReader Engine v10: `OcrAbbyy10.pdf`
    - 3-Heights® OCR Add-on for ABBYY FineReader Engine v11: `OcrAbbyy11.pdf`
    - 3-Heights® OCR Add-on for ABBYY FineReader Engine v12: `OcrAbbyy12.pdf`
    - 3-Heights® OCR Service: `OcrService.pdf` from the separate product kit.

2.1.1 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 `pdf2pdf.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 to PDF/A Converter Shell:

File description

Name	Description
`bin/x64/pdf2pdf`	Main executable
`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 pdf2pdf
```

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/pdf2pdf /usr/bin
```

4. Optionally, register your license key using the license manager.
5. Ensure the cache directory exists as described in Special directories.
6. Make sure your platform meets the requirements regarding color spaces and fonts described in Color spaces and Fonts, respectively.
7. If you want to sign documents, set up your cryptographic provider as described in Cryptographic provider.
8. (Optional) Download and install the 3-Heights® OCR Enterprise Add-on, and the OCR Engine as described in the respective manuals:
   - 3-Heights® OCR Add-on for ABBYY FineReader Engine v10: `OcrAbbyy10.pdf`
   - 3-Heights® OCR Add-on for ABBYY FineReader Engine v11: `OcrAbbyy11.pdf`
   - 3-Heights® OCR Add-on for ABBYY FineReader Engine v12: `OcrAbbyy12.pdf`
   - 3-Heights® OCR Service: `OcrService.pdf` from the separate product kit.

2.3 Uninstall

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

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

2.4 Note about the evaluation license

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

2.5 Special directories

2.5.1 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

2.5.2 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.

2.5.3 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 `pdf2pdf.exe` resides.

macOS

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

Linux

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

3 License management

The 3-Heights® PDF to PDF/A Converter 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.

3.1 License features

The functionality of the 3-Heights® PDF to PDF/A Converter Shell contains one area to which the following license feature is assigned:

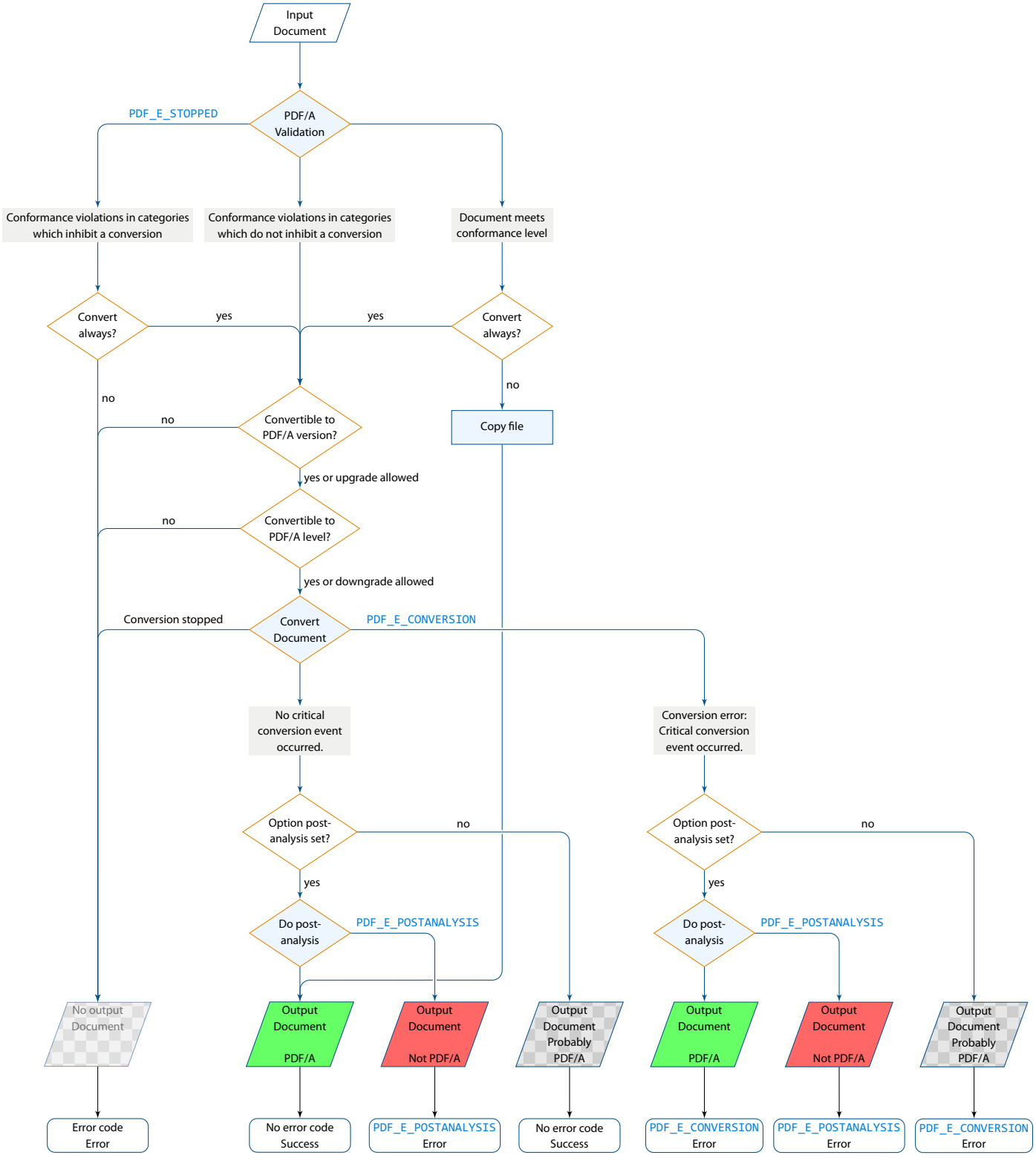
`Signature` Signature creation

The presence of this feature in a given license key can be checked in the license manager. The Interface reference specifies in more detail which functions are included in this license feature.

4 User guide

4.1 Process description

The workflow of the PDF to PDF/A Conversion is outlined in the graphic below.



1. **License check**: The license is checked.
2. **Pre-analysis**: The input document is analyzed. If the document already conforms to the requested standard, it is copied.
   If the required PDF/A level (e.g. level U or A) cannot be met, the conversion is aborted with an error[1].
   If the target standard is PDF/A-1 and a file contains transparency or other elements that cannot be converted to PDF/A-1, the target standard is upgraded to PDF/A-2 if the option -ad is set.
   If the input document contains non-convertible elements, the conversion is stopped, except if the convert-always option is enabled.
3. **Conversion**: The actual conversion is performed.
   The conversion is stopped, e.g. if an OCR error occurs, a required font is not found in the installed font directories, or linearization fails. In this case, no meaningful output document is created.
   If actions had to be taken that might have altered the visual appearance of the file or crucial data had to be removed, a conversion error is generated (see chapter Conversion errors below).
4. **Post-analysis**: Finally, the resulting PDF document is validated. If the resulting document does not meet the requested standard, a post-analysis error is raised.

4.1.1 Conversion steps

The goal of the conversion is to create a document that conforms to the PDF/A ISO standard.

If the analysis of the document indicates a conversion to the requested standard is possible, the following steps are performed:

- Embed and subset non-embedded font programs
- Replace device specific color spaces with CIE-based color spaces
- Add a GTS_PDFA output intent
- Remove prohibited entries
- Remove entries with a default value
- Remove entries with unknown values
- Add mandatory entries
- Add XMP metadata if missing, or fix inconsistent XMP metadata
- Apply implicit optimization functions (e.g. replace and subset embedded fonts)
- Apply implicit repair functions (to conform with ISO19005-1 chapter 6.1)

If the analysis indicates a conversion is not possible, a “best effort” conversion can be forced. In this case, the output may or may not be PDF/A conformant. Use the post-analysis feature to detect whether the output is conformant. It is also possible that the output file may look visually different to the input file due to the forced conversion.

4.1.2 Conversion errors

The conversion error (return code 5) indicates that actions had to be taken during conversion that may have altered the visual appearance of the file or crucial data had to be removed.

> **Note:** The resulting document conforms to PDF/A nonetheless.

The following issues may result in a conversion error:

- Optional content (layers) removed (PDF/A-1 only)
- Prohibited annotation type converted to stamp
- Prohibited action removed

[1] Automatic downgrades can be deactivated using the option -ad.

- Embedded files removed
- Transparency removed (PDF/A-1 only)
- Character from show string removed because glyph missing in font
- Unconvertible metadata

All conversion events are written to the log file. The description and location let you identify potential problems quickly.

For a complete list of conversion events that can lead to a conversion error, see the option `-cem`.

Handling conversion errors

You should check which conversion errors are tolerable in your process and which must be considered critical. Set the option `-cem` to include critical errors only.

Conversion errors can often be resolved by optimizing the 3-Heights® PDF to PDF/A Converter Shell's options or installation:

1. Conversion errors can be minimized by converting to PDF/A-2 instead of PDF/A-1. PDF/A-2 allows some features of newer versions of the PDF Reference, e.g. transparency, optional content (layers), or embedded files.
2. If fonts were substituted, the missing fonts should be installed (see Fonts).
3. For documents with non-convertible XMP metadata, you should update the PDF creating software to generate valid XMP metadata.
4. How signed documents can be converted to PDF/A is explained in the following blog post by Dr. Hans Bärfuss.

In case of a conversion error, the output file is best presented to the user so they can decide whether or not the conversion result is acceptable. The conversion log file is helpful to display a meaningful message, e.g. "Embedded files have been removed during PDF/A conversion." All conversion events are written to the log file and include the cause of the conversion error.

For fully automated processes or documents that the user cannot accept, a fallback conversion can be added. For different conversion errors, different fallback conversions may be required:

1. If embedded files were removed, they can be extracted using the product 3-Heights® PDF Extract and then converted to PDF/A by the 3-Heights® PDF Document Converter.
2. Other conversion errors can be dealt with by creating an image-based PDF using the 3-Heights® PDF to Image Converter, which renders all pages and replaces their content with the resulting image. Optionally, some data such as bookmarks, links, or document metadata can be preserved.

4.1.3 Post-analysis

The post-analysis step checks whether the output file conforms to the requested standard. A post-analysis error (return code 6) indicates that the output file is not PDF/A.

In case of a post-analysis error, the conversion can be repeated with the option `-rd`. The log file then indicates why the post-analysis failed. Often the issue can be resolved. For example, by installing missing fonts (see Fonts) or with the option `-ad`.

If the file cannot be converted, a meaningful fallback could be the conversion to an image-based PDF as described above in chapter Handling conversion errors.

4.2 What is PDF/A?

PDF/A is an ISO standard for using the PDF format for the long-term archiving of electronic documents. This chapter provides a brief overview. For additional information, see `https://www.pdf-tools.com/en/resources/pdf-iso-standards/`.

4.2.1 PDF/A-1

The PDF/A-1 format is described in the international standard ISO-19005-1. It is based on the PDF 1.4 reference and has some additional requirements. It is beneficial to have a general understanding of PDF/A. Here is a brief overview of how to create a PDF/A document from a non-PDF/A document.

1. A PDF/A has requirements about metadata and the structure of the file. The PDF to PDF/A Converter takes care of this and no user intervention is required. However, you can provide the XMP metadata if desired.
2. In PDF/A, colors (including grayscale and black/white) must not be represented in a device color space (DeviceRGB, DeviceCMYK, DeviceGray). You can provide suitable default color space profiles to substitute the device color spaces, one for RGB, CMYK, and grayscale, respectively. In addition, or alternatively, one color space profile can be embedded as output intent. For the latter, device colors are automatically managed by the output intent if the color can be represented in the space given by the color space profile in the output intent.
   If the converter encounters unmanaged colors (for example, because no color space profile was set), then a calibrated color space is generated automatically, one RGB, and one grayscale, for RGB and grayscale colors, respectively. If unmanaged CMYK colors are encountered, a default CMYK output intent is embedded.
3. Fonts used in visible text must be embedded. This is automatically done by the converter.
4. For PDF/A-1a: The original document structure information is retained when converting the file to PDF/A. However, new tags are not added and the structure is not changed. To create a PDF/A-1a conforming file, the original file must have been created with the required structure and tagging. Otherwise, a PDF/A-1b file is produced.

4.2.2 What is the difference between PDF/A-1b and PDF/A-1a?

PDF/A-1a has additional specifications on top of those for PDF/A-1b. These are:

1. Font encoding must meet additional requirements, e.g. include a ToUnicode mapping (ISO 19005-1, chapter 6.3.8)
2. The document must contain a logical structure (ISO 19005-1, chapter 6.8)

The idea of the PDF/A-1a requirements is mainly to provide support for disabled people, i.e. by providing the required information needed for applications that support text-to-speech features.

The logical structure of the document is a description of the content of the pages. This description has to be provided by the creator of the document. It consists of a fine granular hierarchical tagging that distinguishes between the actual content and artifacts (such as page numbers, footers, layout artifacts, etc.). The tagging provides a meaningful description. Examples are “This is a Header”, “This color image shows a small sailing boat at sunset”, etc. This information cannot be generated automatically; it needs to be provided. This is one of the reasons why not every PDF document can be converted to PDF/A-1a.

4.2.3 PDF/A-2

PDF/A-2 is described in ISO 19005-2. It is based on ISO 32000-1, the standard for PDF 1.7. PDF/A-2 is meant as an extension to PDF/A-1. The second part complements the first part and does not replace it. The most important differences between PDF/A-1 and PDF/A-2 are:

- The list of compression types has been extended by JPEG2000
- Transparent contents produced by graphic programs are allowed
- Optional contents (also known as layers) can be made visible or invisible
- Multiple PDF/A files can be bundled in one file (collection, package)
- The additional conformity level U (Unicode) allows for creating searchable files without having to fulfill the strict requirements of the conformity level A (accessibility)
- File size can be reduced using compressed object and XRef streams

Documents that contain features described above, in particular layers or transparency, should therefore be converted to PDF/A-2 rather than PDF/A-1.

4.2.4 PDF/A-3

PDF/A-3 is described in ISO 19005-3. It is based on ISO 32000-1, the standard for PDF 1.7. PDF/A-3 is an extension to PDF/A-2. The third part shall complement the second part and not replace it. The only two differences between PDF/A-2 and PDF/A-3 are:

- Files of any format and conformance may be embedded. Embedded files need not be suitable for long-term archiving.
- Embed files can be associated with any part of the PDF/A-3 file.

4.3 Color spaces

4.3.1 Colors in PDF

The PDF format supports a range of color spaces:

Device color spaces (DeviceGray, DeviceRGB, and DeviceCMYK) These are also referred to as uncalibrated color spaces, because they cannot be used to specify color values such that colors are reproducible in a predictable way on multiple output devices.

CIE-based color spaces (CalGray, CalRGB, Lab, ICCBased) These are also referred to as device-independent color spaces, because they are inherently capable of specifying colors that can be reliably reproduced on multiple output devices.

Special color spaces (Separation and DeviceN) These require an alternate color space from one of the previous two groups to allow the PDF consumer to simulate the color on devices that do not support the special color space.

Colors can occur in the following objects of a PDF/A document:

- Raster images (also inline images)
- Text and vector objects such as lines and curves
- Annotations
- Shading patterns
- Transparency blending (PDF/A-2 and later)

ICC color profiles

An ICC (International Color Consortium) profile is a file format that can be used to describe the color characteristics of a particular device. For example, for the correct color reproduction when an image from a scanner or camera is displayed on a device such as a monitor or printer. Color profiles are usually provided with the operating system (OS). On a Windows System, they can be found at the following location:

```
%SystemRoot%\system32\spool\drivers\color
```

Alternatively, additional profiles can be found here:

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

Please note that most color profiles are copyrighted, therefore you should read the license agreements on the above links before using the color profiles. The PDF to PDF/A Converter tries to locate color profiles automatically in the `%SystemRoot%\system32\spool\drivers\` color folder as needed. On Linux or macOS, you can store the

color profiles contained in the `colorprofiles.zip` download in a folder of your choice, and set the environment variable `PDF_ICC_PATH` to point to that folder.

PDF/A requirements

In PDF/A, the usage of uncalibrated color spaces (DeviceGray, DeviceRGB, and DeviceCMYK) is prohibited because colors that are specified in this way cannot be reproduced reliably on multiple output devices. Therefore, when converting to PDF/A, all device color spaces should be replaced by CIE-based color spaces. There is one exception to this rule: An uncalibrated color is tolerated if the output intent holds an ICC color profile with which the color can be represented. (E.g. a grayscale color can be represented in an RGB color profile, but a CMYK color cannot.)

The 3-Heights® PDF to PDF/A Converter Shell uses the following strategy:

- For each device color space (DeviceGray, DeviceRGB, and DeviceCMYK), an ICC color profile can be specified to be used as substitute for the respective device color space.
- If an output intent is present, it is copied.
- Optionally, an ICC color profile can be set to be used in the output intent.
- During conversion, if a device color space is encountered then the following is done:
  - If an output intent was set that is capable of managing this color, no action is needed.
  - Otherwise, if an ICC color profile is set to substitute this device color space, then this color profile is used.
  - Otherwise, for DeviceRGB and DeviceGray color spaces: A calibrated color space (CalRGB[2] and CalGray respectively) is generated and used as a substitute.
  - Otherwise, for DeviceCMYK color spaces:
    - If the output intent is not set, then a default CMYK ICC color profile is used for the output intent.
    - If the output intent holds a non-CMYK ICC color profile, then a default CMYK ICC color profile is generated and used as a substitute for DeviceCMYK.

The above strategy is motivated by the fact that CalRGB and CalGray color spaces occupy very little memory in comparison to ICC color profiles. Also note that the primary purpose of the output intent in a PDF document is to describe the characteristics of the device on which a document is intended to be rendered. Traditionally, the target device is a printer, which motivates CMYK output intents. The default CMYK color profile `USWebCoatedSWOP.icc` is provided in the sub-directory `bin\icc`.

4.4 Fonts

The PDF/A standard requires all fonts to be embedded in the PDF file. This ensures that the future rendering of the textual content of a conforming file matches, on a glyph by glyph basis, the appearance of the file as originally created.

Hence, if non-embedded fonts in a PDF are used, the font must be embedded. For this, a matching font has to be found in the Font directories. The option `-fd` should be used to define additional directories. The default font directories are listed in Font directories.

It is important that the Font directories contain all fonts that are used for the input files.

Fonts should be added to one of the Font directories if the post-analysis returns validation errors such as:

```
"output.pdf", 9, 20, 0x00418704, "The font ShinGo must be embedded.", 1
```

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

[2] The generated CalRGB color space is an approximation to the ICC color profile `sRGB Color Space Profile.icm`.

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`.

4.4.1 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 to PDF/A Converter 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.

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

Alternatively, you can download the fonts from here:

`https://corefonts.sourceforge.net/`

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

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

4.4.3 Font configuration file fonts.ini

The font configuration file is optional. It can be used to control the embedding of fonts.

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

Windows: In a directory named `Fonts`, which must be a direct subdirectory of where `pdf2pdf.exe` resides.

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

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

`fonts.ini` uses the INI file format and has two sections. The section `[fonts]` is ignored by the 3-Heights® PDF to PDF/A Converter Shell, so you may remove it. In the section `[replace]`, font replacement rules of the form `key=value` can be defined. The key specifies the font that is to be replaced. The key should match the name of the font mentioned in the pre-analysis of the 3-Heights® PDF to PDF/A Converter Shell, e.g. `"ShingGo"` for:

```
"file.pdf", 9, 20, 0x00418704, "The font ShinGo must be embedded.", 1
```

The value should match the true type name of an installed font. Do not replace any standard fonts (Helvetica, Arial, Times, TimesNewRoman, Courier, CourierNew, Symbol, and ZapfDingbats).

Please note that this feature should be used with care. Replacing a font with another might change the visual appearance of the file because of different glyph shapes, metrics, or glyphs that are not available in the replacement font. Embedding another font might also have legal implications.

Example: Replace MS-Mincyo with MS-Mincho

```
[replace]
MS-Mincyo=MS-Mincho
```

This rule defines that to embed a font program for font MS-Mincyo, the font MS-Mincho should be used. This rule is useful, because both names are possible transliterations of the same Japanese font. However, the official transliteration used by the actual font is MS-Mincho.

4.5 Cryptographic provider

In order to use the 3-Heights® PDF to PDF/A Converter Shell's cryptographic functions such as creating digital signatures, a cryptographic provider is required. The cryptographic provider manages certificates and their private keys, and implements cryptographic algorithms.

The 3-Heights® PDF to PDF/A Converter Shell can use various different cryptographic providers. The following list shows the provider that can be used for each type of signing certificate.

USB token or smart card These devices typically offer a PKCS#11 interface, which is the recommended way to use the certificate →PKCS#11 provider.

On Windows, the certificate is usually also available in the Windows Cryptographic Provider.

In any case, signing documents is only possible in an interactive user session.

Hardware Security Module (HSM) HSMs always offer very good PKCS#11 support →PKCS#11 provider

For more information and installation instructions, see the separate document `TechNotePKCS11.pdf`.

Soft certificate Soft certificates are typically PKCS#12 files that have the extension `.pfx` or `.p12` and contain the signing certificate, as well as the private key and trust chain (issuer certificates). Soft certificate files cannot be used directly. Instead, they must be imported into the certificate store of a cryptographic provider.

- *All platforms*: The recommended way of using soft certificates is to import them into a store that offers a PKCS#11 interface and use the PKCS#11 provider. For example:
  - A HSM
  - openCryptoki on Linux

  For more information and installation instructions of the stores, see the separate document `TechNotePKCS11.pdf`.
- *Windows*: If no PKCS#11 provider is available, soft certificates can be imported into Windows certificate store, which can then be used as cryptographic provider →Windows Cryptographic Provider

Signature service Signature services are a convenient alternative to storing certificates and key material locally. The 3-Heights® PDF to PDF/A Converter Shell can use various different services. The configuration is explained in the following sections of this documentation:

- myBica Digital Signing Service
- Swisscom All-in Signing Service
- GlobalSign Digital Signing Service
- QuoVadis sealsign

4.5.1 PKCS#11 provider

PKCS#11 is a standard interface offered by most cryptographic devices such as HSMs, USB tokens, or sometimes even soft stores (e.g. openCryptoki).

More information on and installation instructions of the PKCS#11 provider of various cryptographic devices can be found in the separate document TechNotePKCS11.pdf.

Configuration

Provider Option -cp

The provider configuration string has the following syntax:

"‹PathToDll›;‹SlotId›;‹Pin›"

‹PathToDll› Path to driver library filename, which is provided by the manufacturer of the HSM, UBS token, or smart card. Examples:

- The CardOS API from Atos (Siemens) uses siecap11.dll
- The IBM 4758 cryptographic coprocessor uses cryptoki.dll
- Devices from Aladdin Ltd., use etpkcs11.dll
- For SafeNet Luna, HSM use cryptoki.dll an Windows or libCryptoki2_64.so on Linux/UNIX.
- For Securosys SA, Primus HSM or CloudsHSM, use primusP11.dll[3] on Windows and libprimusP11.so[3] on Linux.
- For Google Cloud HSM (Cloud KMS), use libkmsp11.so[4].

‹SlotId› (optional). If it is not defined, it is searched for the first slot that contains a running token.

‹Pin› (optional). If it is not defined, the submission for the PIN is activated via the pad of the token.

If this is not supported by the token, the following error message is raised when signing: "Private key not available."

Example:

```
-cp "C:\Windows\system32\siecap11.dll;4;123456"
```

Interoperability support

The following cryptographic token interface (PKCS#11) products have been successfully tested:

- SafeNet Protect Server
- SafeNet Luna
- SafeNet Authentication Client
- IBM OpenCrypTokI
- CryptoVision
- Siemens CardOS
- Utimaco SafeGuard CryptoServer
- Securosys SA CloudsHSM[3]

Selecting a certificate for signing

The 3-Heights® PDF to PDF/A Converter Shell offers different ways to select a certificate. The product tries the first of the following selection strategies, for which the required values have been specified by the user.

1. **Certificate fingerprint**

[3] It is recommended to use version 1.7.32 or newer of the Primus HSM PKCS#11 Provider.
[4] Must be used as described in PKCS#11 devices that contain private keys only.

Option `-cfp`

- SHA-1 fingerprint of the certificate. The fingerprint is 20 bytes long and can be specified in hexadecimal string representation, e.g. "b5 e4 5c 98 5a 7e 05 ff f4 c6 a3 45 13 48 0b c6 9d e4 5d f5". In Windows certificate store, this is called "Thumbprint", if "Thumbprint algorithm" is "sha1".

2. **Certificate issuer and serial number**
   Options `-ci` and `-cno`
   - Certificate issuer (e.g. "QV Schweiz CA"). In Windows certificate store, this is called "Issued By".
   - Serial number of the certificate (hexadecimal string representation, e.g. "4c 05 58 fb"). This is a unique number assigned to the certificate by its issuer. In Windows certificate store, this is the field called "Serial number" in the certificate's "Details" tab.
3. **Certificate name and issuer (optional)**
   Options `-cn` and `-ci`
   - Common Name of the certificate (e.g. "PDF Tools AG"). In Windows certificate store, this is called "Issued To".
   - Optional: Certificate issuer (e.g. "QV Schweiz CA"). In Windows certificate store, this is called "Issued By".

Using PKCS#11 stores with missing issuer certificates

Some PKCS#11 devices contain the signing certificate only. However, to embed revocation information, it is important that the issuer certificates, i.e. the whole trust chain, is available as well.

On Windows, missing issuer certificates can be loaded from the Windows certificate store. Missing certificates can be installed as follows:

1. Get the certificates of the trust chain. You can download them from the website of your certificate provider or do the following:
   a. Sign a document and open the output in Adobe Acrobat.
   b. Go to "Signature Properties" and then view the signer's certificate.
   c. Select a certificate of the trust chain.
   d. Export the certificate as "Certificate File" (extension `.cer`).
   e. Do this for all certificates of the trust chain.
2. Open the exported files by double clicking on them in Windows Explorer.
3. Click "Install Certificate...".
4. Select "automatically select the certificate store based on the type of certificate" and finish import.

PKCS#11 devices that contain private keys only

Some PKCS#11 devices, such as the Google Cloud HSM (Cloud KMS), can only store private keys and no certificates. In such cases, it is possible to supply the required certificates externally using the option `-cps`.

Name	Type	Required	Value
Certificate	Bytes	Required	The signing certificate in either PEM (.pem, ASCII text) or DER (.cer, binary) form. This certificate must be selected as the signing certificate as described in Selecting a certificate for signing.

PrivateKeyUri	String	Required	The RFC 7512 URI specifying the private key object in the store. The following URI formats are supported: **`pkcs11:object=‹label›`** To specify the CKA_LABEL object attribute of the private key. The `‹label›` is a text string that is converted to UTF-8 and percent-decoded before matching the CKA_LABEL attribute. Example: `"pkcs11:object=Signing Certificate"` **`pkcs11:id=‹id›`** To specify the CKA_ID object attribute of the private key. The value of the `‹id›` can be percent-encoded to match CKA_ID attributes with binary data. Example: `"pkcs11:id=%C8%48%EC%66%00%17%01%BA%AE%06"` This private key object must belong to the certificate that was specified by the session property **Certificate**.
TrustChain	Bytes	Recommended	The certificates of the trust chain in either PEM (.pem, ASCII text) or DER (.cer, binary) form. Multiple certificates can be concatenated into a single byte stream. Supplying the certificates is highly recommended and required, if revocation information (CRL, OCSP) should be embedded (see Option `-co`).

```
pdf2pdf -v -cp "myPKCS11.dll;0;pin" -cn "Signing Certificate" ^
-cpf Certificate signing-certificate.cer ^
-cps PrivateKeyUri "pkcs11:object=Signing Certificate" ^
-cpf TrustChain trust-chain.cer ^
input.pdf signed.pdf
```

4.5.2 Cryptographic suites

Message digest algorithm

The default hash algorithm to create the message digest is `SHA-256`. Other algorithms can be chosen by setting the provider session property `MessageDigestAlgorithm`, for which supported values are:

SHA-1 This algorithm is considered broken and therefore strongly discouraged by the cryptographic community.

SHA-256 (default)

SHA-384

SHA-512

RIPEMD-160

Signing algorithm

The signing algorithm can be configured by setting the provider session property `SigAlgo`. Supported values are:

RSA_RSA (default) This is the RSA PKCS#1v1.5 algorithm, which is widely supported by cryptographic providers.

RSA_SSA_PSS This algorithm is sometimes also called RSA-PSS.

Signing will fail if the algorithm is not supported by the cryptographic hardware. The device must support either the signing algorithm CKM_RSA_PKCS_PSS (i.e. RSA_SSA_PSS) or CKM_RSA_X_509 (i.e. raw RSA).

> **Note:** Setting the signing algorithm only has an effect on signatures created by the cryptographic provider itself. All signed data acquired from external sources may use other signing algorithms, specifically the issuer signatures of the trust chain, the timestamp's signature, or those used for the revocation information (CRL, OCSP). It is recommended to verify that the algorithms of all signatures provide a similar level of security.

4.6 Windows Cryptographic Provider

This provider uses Windows infrastructure to access certificates and to supply cryptographic algorithms. Microsoft Windows offers two different APIs, the Microsoft CryptoAPI and Cryptography API Next Generation (CNG).

Microsoft CryptoAPI Provides functionality for using cryptographic algorithms and for accessing certificates stored in the Windows certificate store and other devices, such as USB tokens, with Windows integration.

Microsoft CryptoAPI does not support some new cryptographic algorithms, such as SHA-256.

Cryptography API: Next Generation (CNG) CNG is an update to CryptoAPI. It extends the variety of available cryptographic algorithms, e.g. by the SHA-256 hashing algorithms. If possible, the 3-Heights® PDF to PDF/A Converter Shell performs cryptographic calculations with CNG instead of CryptoAPI.

CNG is available only if:

- The operating system is at least Windows Vista or Windows Server 2008.
- The provider of the signing certificate's private key, e.g. the USB token or smart card, supports CNG.

If CNG is not available, the CryptoAPI's cryptographic algorithms are used. In any case, CryptoAPI is used for the certificate accessing functionalities.

> **Default message digest algorithm:** Since version 4.6.12.0 of the 3-Heights® PDF to PDF/A Converter Shell, the default message digest algorithm is SHA-256. As a result, signing will fail if CNG is not available (error message "Private key not available."). To use SHA-1, the provider session property `MessageDigestAlgorithm` can be used. Use of SHA-1 is strongly discouraged by the cryptographic community.

4.6.1 Configuration

Provider Option `-cp`

The provider configuration string has the following syntax:

`"[‹ProviderType›:]‹Provider›[;‹PIN›]"`

The `‹ProviderType›` and `‹PIN›` are optional. The corresponding drivers must be installed on Windows. If CNG is available, `‹ProviderType›` and `‹Provider›` are obsolete and can be omitted.

Optionally, when using an advanced certificate, the PIN code (password) can be passed as an additional, semi-column separated parameter ‹PIN›. This does not work with qualified certificates, because they always require the PIN code to be entered manually every time.

If ‹Provider› is omitted, the default provider is used. The default provider is suitable for all systems where CNG is available.

Examples: Use the default provider with no PIN.

```
Provider = ""
```

Examples: "123456" being the PIN code.

```
Provider = ";123456"
```

```
Provider = "Microsoft Base Cryptographic Provider v1.0;123456"
```

```
Provider = "PROV_RSA_AES:Microsoft Enhanced RSA and AES Cryptographic" _
  + "Provider;123456"
```

Certificate store Option -csn

The value for the certificate store depends on the OS. Supported values are: "CA", "MY" and "ROOT". For signature creation, the default store "MY" is usually the right choice.

Store location Option -csl

Either of the following store locations:

- "Local machine"
- "Current user" (default)

Usually, personal certificates are stored in the "current user" location and company-wide certificates are stored under "local machine".

The "current user" store is only available, if the user profile has been loaded. This may not be the case in certain environments, such as within an IIS web application or COM+ applications. Use the store of the local machine if the user profile cannot be loaded. For other services, it is sufficient to log on as the user. Some cryptographic hardware (such as smart cards or USB tokens) require an interactive environment. As a result, the private key might not be available in the service session, unless the 3-Heights® PDF to PDF/A Converter Shell is run interactively.

Certificates in the "Local Machine" store are available to all users. However, in order to sign a document, you need access to the signing certificate's private key. The private key is protected by Windows ACLs and typically readable for Administrators only. Use the Microsoft Management Console (mmc.exe) to grant access to the private key for other users as follows:

Add the Certificates Snap-in for the certificates on local machine. Right-click on the signing certificate, click on "All Tasks" and then "Manage Private Keys..." where you can set the permissions.

4.6.2 Selecting a certificate for signing

Within the certificate store selected by Store location and Certificate store, the selection of the signing certificate works the same as with the PKCS#11 provider. For more information, see Selecting a certificate for signing.

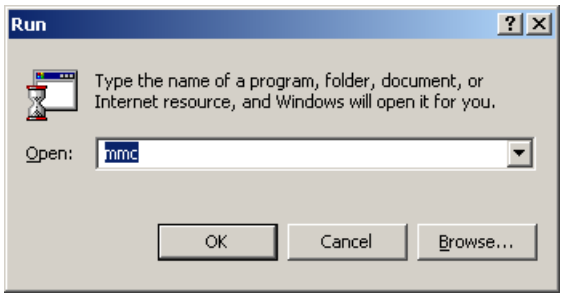
4.6.3 Certificates

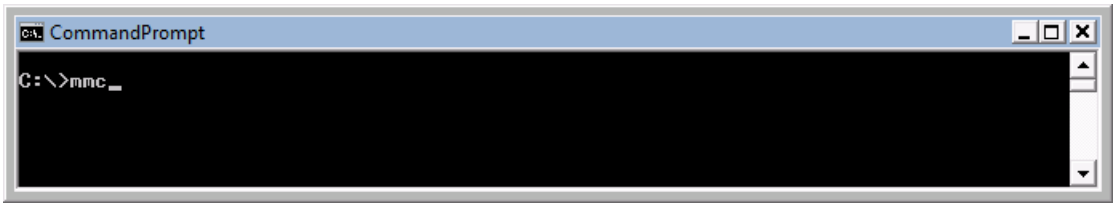
To sign a PDF document, a valid existing certificate name must be provided and its private key must be available.

There are various ways to create or obtain a certificate. How this is done is not described in this document. This document describes the requirements for and how to use the certificate.

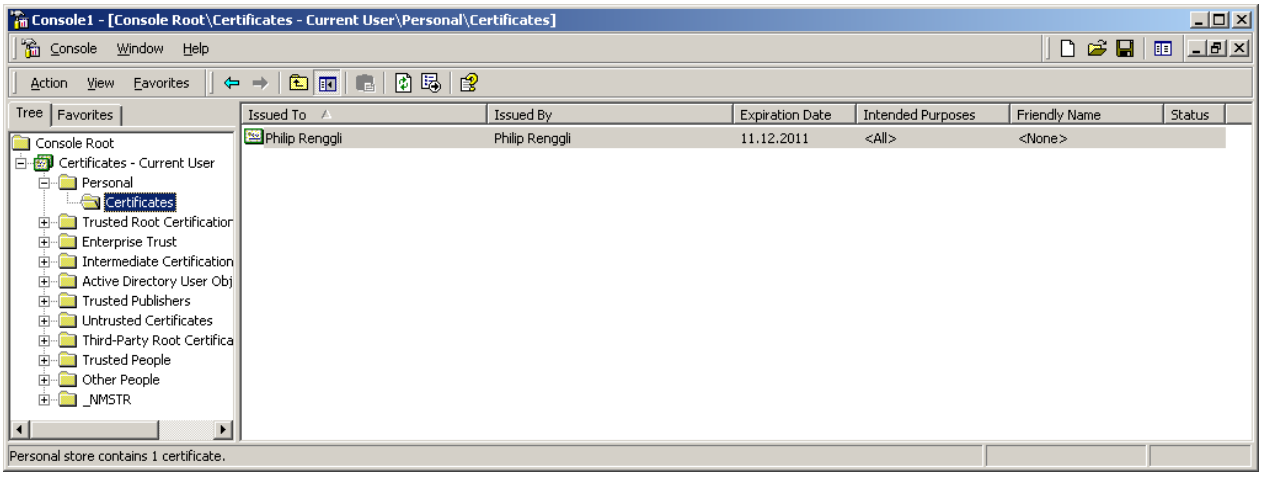
On the Windows operating system, certificates can be listed by the Microsoft Management Console (MMC), which is provided by Windows. To see the certificates available on the system, perform the following steps:

1. To launch the MMC, go to Start → Run…→ type “mmc”, or start a Command Prompt and type “mmc”.

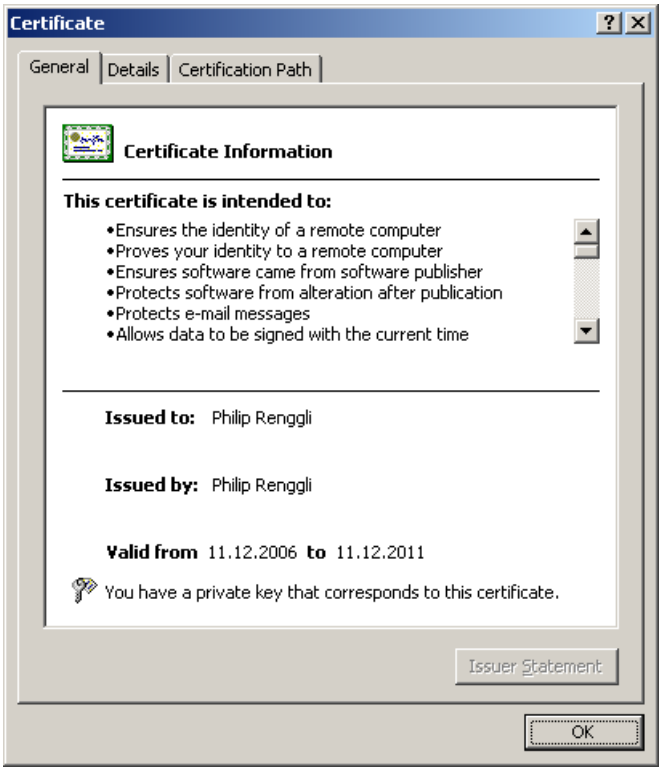




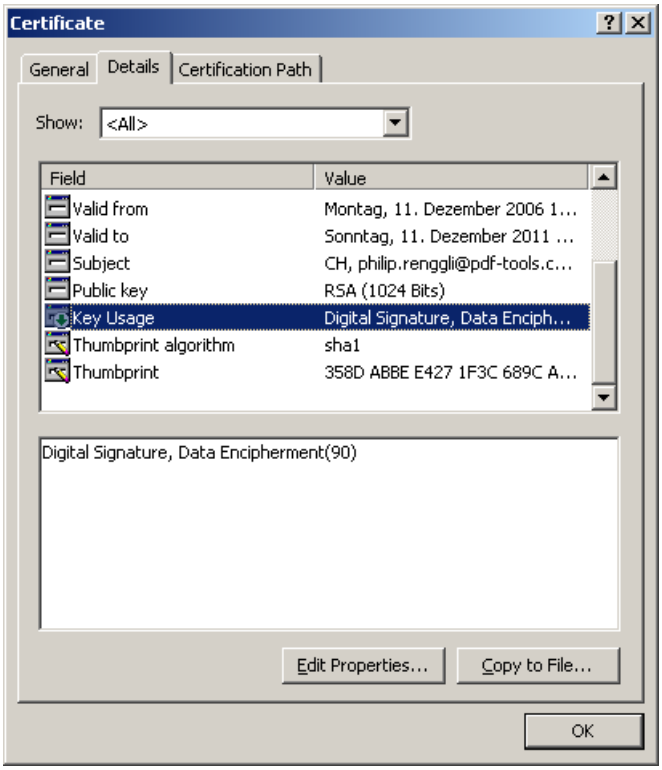
2. Under “File” →“Add/Remove Snap-in”.
3. Choose “Certificates” and click the “Add” button.
4. In the next window choose to manage certificates for “My user account”.
5. Click “Finish”.
6. The certificate must be listed under the root “Certificates - Current User”. For example, as shown in the screen-shot below:



7. Double-click the certificate to open. The certificate name corresponds to the value “Issued to”.

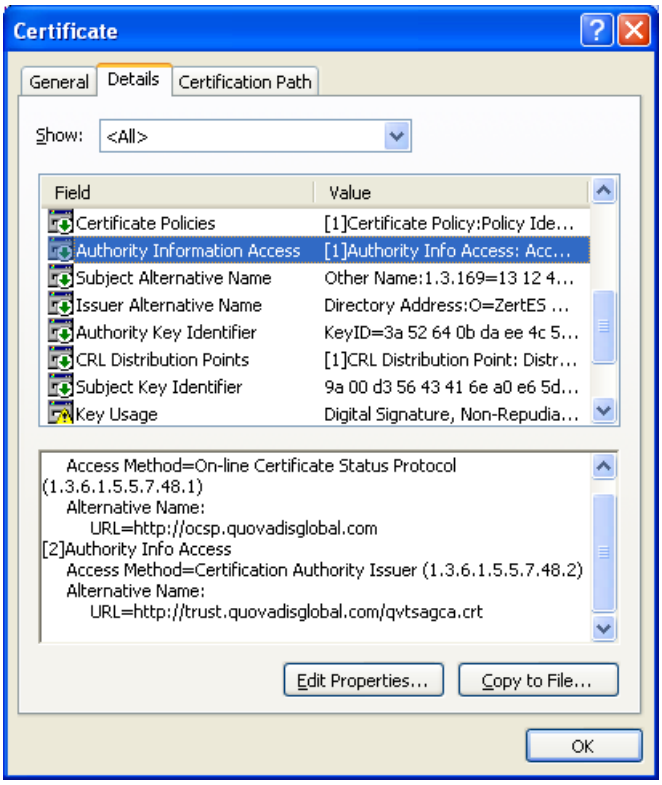


8. In the Details tab of the certificate, there is a field named “Key Usage”. This field must contain the value “Digital Signature”. Additional values are optional. See the figure below.
   You must have the private key that corresponds to this certificate.



4.6.4 Qualified certificates

A qualified certificate can be obtained from a certificate authority (CA). Besides the requirements listed in the previous chapter, it has the additional requirement to contain the key “Authority Information Access”, which contains the information about the OCSP server.



4.6.5 Cryptographic suites

The message digest algorithm and the signing algorithm can be chosen as described for the PKCS#11 provider in Cryptographic suites.

The `MessageDigestAlgorithm` can only be set to a value other than `SHA-1` if the private key’s provider supports CNG.

The `SigAlgo` can only be set to `RSA_SSA_PSS` if the private key’s provider supports CNG.

4.7 myBica Digital Signing Service

Provider Option `-cp`

The provider configuration string contains the URL to the service endpoint, typically, `https://sign.mybica.ch/DS/DS`.

Provider configuration The provider can be configured using provider session properties.

There are two types of properties:

- “String” Properties:
  String properties are set using option `-cps`.
- “File” Properties:
  File properties are set using option `-cpf`.

Name	Type	Required	Value
Identity	String	Required	The identity of your signing certificate. Example: `My Company:Signing Cert 1`
DSSProfile	String	Required	Must be set to `http://www.pdf-tools.com/dss/profile/pades/1.0`
SSLClientCertificate	File	Required	SSL client certificate in PKCS#12 Format (`.p12`, `.pfx`). File must contain the certificate itself, all certificates of the trust chain and the private key.
SSLClientCertificatePassword	String	Optional	Password to decrypt the private key of the SSL client certificate.
SSLServerCertificate	File	Recommended	Certificate of the server or its issuer (CA) certificate (`.crt`). The certificate may be in either PEM (ASCII text) or DER (binary) form. Note: If this property is not set, the server certificate's trustworthiness cannot be determined. As a result, the connection is not guaranteed to be secure.
RequestID	String	Recommended	Any string that can be used to track the request. Example: An UUID like `AE57F021-C0EB-4AE0-8E5E-67FB93E5BC7F`

Signature configuration The signature can be customized using standard options of the 3-Heights® PDF to PDF/A Converter Shell.

Description	Required	Value	Setting
Common Name	Required	The name of the signer must be set[5].	Option `-cn`.
Timestamp	optional	Use the value `urn:ietf:rfc:3161` to embed a timestamp.	Option `-tsu`
Signature Format	Optional	To set the signature format	Option `-st`. Must be `adbe.pkcs7.detached`
Revocation Info	Recommended	To embed OCSP responses or CRL.	Option `-co`[6]

Visual Appearance	Optional	See Creating a visual appearance of a signature.

Proxy configuration If a proxy is used for the connection to the service, see Using a proxy for more information.

4.8 QuoVadis sealsign

Provider Option `-cp`

The provider configuration string contains the URL to the QuoVadis sealsign service.

- Demo service:
  `https://services.sealsignportal.com/sealsign/ws/BrokerClient`
- Productive service:
  `https://qvchsvsws.quovadisglobal.com/sealsign/ws/BrokerClient`

Provider configuration The provider can be configured using provider session properties that can be set using the options `-cps` or `-cpf`.

Name	Type	Required	Value
Identity	String	Required	The account ID is the unique name of the account specified on the server. Example: `Rigora`
Profile	String	Required	The profile identifies the signature specifications by a unique name. Example: `Default`
secret	String	Required	The secret is the password which secures the access to the account. Example: `NeE=EKEd33FeCk70`
clientId	String	Required	A client ID can be used to help separating access and creating better statistics. If specified in the account configuration it is necessary to provide this value. Example: `3949-4929-3179-2818`
pin	String	Required	The PIN code is required to activate the signing key. Example: `123456`

[5] This parameter is not used for certificate selection, but for the signature appearance and description in the PDF only.
[6] The recommendation is to not use the option `-co`.

MessageDigestAlgorithm	String	Optional	The message digest algorithm to use. Default: `SHA-256` Alternatives: `SHA-1`, `SHA-384`, `SHA-512`, `RIPEMD-160`, `RIPEMD-256`

Signature configuration The signature can be customized using standard options.

Description	Required	Value	Setting
Common Name	Required	The name of the signer must be set[7].	Option `-cn`.
Timestamp	-	Not available.	
Revocation Info	Recommended	To embed OCSP responses or CRL.	Option `-co`[8]
Visual Appearance	Optional	See Creating a visual appearance of a signature.	

Proxy configuration If a proxy is used for the connection to the service, see Using a proxy for more information.

4.9 Swisscom All-in Signing Service

4.9.1 General properties

To use the signature service, the following general properties have to be set:

Description	Required	Value	Setting
Common Name	Required	Name of the signer[9].	Option `-cn`
Provider	Required	The service endpoint URL of the REST service. Example: `https://ais.swisscom.com/AIS-Server/rs/v1.0/sign`	Option `-cp`
Timestamp	optional	Use the value `urn:ietf:rfc:3161` to embed a timestamp.	Option `-tsu`
Signature Format	Optional	To set the signature format	Option `-st`. Supported values are `adbe.pkcs7.detached`, `ETSI.CAdES.detached`.

[7] This parameter is not used for certificate selection, but for the signature appearance and description in the PDF only.
[8] The recommendation is to not use the option `-co`.

Revocation Info	Optional	To embed OCSP responses	Option `-co`. Supported with `adbe.pkcs7.detached` only.

If a proxy is used for the connection to the service, see Using a proxy for more information.

4.9.2 Provider session properties

In addition to the general properties, a few provider specific session properties have to be set.

There are two types of properties:

- "String" Properties:
  String properties are set using option `-cps`.
- "File" Properties:
  File properties are set using option `-cpf`.

Name	Type	Required	Value
DSSProfile	String	Required	Must be set to `http://ais.swisscom.ch/1.0`
SSLClientCertificate	File	Required	SSL client certificate in PKCS#12 Format (`.p12`, `.pfx`). File must contain the certificate itself, all certificates of the trust chain and the private key.
SSLClientCertificatePassword	String	Optional	Password to decrypt the private key of the SSL client certificate.
SSLServerCertificate	File	Recommended	Certificate of the server or its issuer (CA) certificate (`.crt`). The certificate may be in either PEM (ASCII text) or DER (binary) form. Note: If this property is not set, the server certificate's trustworthiness cannot be determined. As a result, the connection is not guaranteed to be secure.
Identity	String	Required	The Claimed Identity string as provided by Swisscom: `‹customer name›:‹key identity›`
RequestID	String	Recommended	Any string that can be used to track the request. Example: An UUID like `AE57F021-C0EB-4AE0-8E5E-67FB93E5BC7F`

4.9.3 On-demand certificates

To request an on-demand certificate, the following additional property has to be set:

[9] This parameter is not used for certificate selection, but for the signature appearance and description in the PDF only.

Name	Type	Required	Value
SwisscomAllInOnDemandDN	String	Required	The requested distinguished name. Example: `cn=Hans Muster,o=ACME,c=CH`

4.9.4 Step-up authorization using Mobile-ID

To use the step-up authorization, the following additional properties have to be set:

Name	Type	Required	Value
SwisscomAllInMSISDN	String	Required	Mobile phone number. Example: `+41798765432`
SwisscomAllInMessage	String	Required	The message to be displayed on the mobile phone. Example: `Pipapo halolu`.
SwisscomAllInLanguage	String	Required	The language of the message. Example: `DE`

Those properties have to comply with the Swisscom Mobile-ID specification.

4.10 GlobalSign Digital Signing Service

Provider Option `-cp`

The provider configuration string contains the URL to the service endpoint.

`https://emea.api.dss.globalsign.com:8443/v2`

Provider configuration The provider can be configured using provider session properties.

There are two types of properties:

- "String" Properties:
  String properties are set using option `-cps`.
- "File" Properties:
  File properties are set using option `-cpf`.

Name	Type	Required	Value
api_key	String	Required	Your account credentials' key parameter for the login request.
api_secret	String	Required	Your account credentials' secret parameter for the login request.

Identity	String	Required	Parameter to create the signing certificate. Example for an account with a static identity: `{}` Example for an account with a dynamic identity: `{ "subject_dn": { "common_name": "John Doe" } }`
SSLClientCertificate	File	Required	SSL client certificate in PKCS#12 Format (`.p12`, `.pfx`). File must contain the certificate itself, all certificates of the trust chain and the private key.
SSLClientCertificatePassword	String	Optional	Password to decrypt the private key of the SSL client certificate.
SSLServerCertificate	File	Recommended	Certificate of the server or its issuer (CA) certificate (`.crt`). The certificate may be in either PEM (ASCII text) or DER (binary) form. Note: If this property is not set, the server certificate's trustworthiness cannot be determined. As a result, the connection is not guaranteed to be secure.

Signature configuration The signature can be customized using standard options of the 3-Heights® PDF to PDF/A Converter Shell.

Description	Required	Value	Setting
Common Name	Required	The name of the signer must be set[10].	Option `-cn`.
Timestamp	recommended	Use the value `urn:ietf:rfc:3161` to embed a timestamp.	Option `-tsu`
Signature Format	Optional	To set the signature format	Option `-st`. Supported values are `adbe.pkcs7.detached`, `ETSI.CAdES.detached`.
Revocation Info	Recommended	To embed OCSP responses or CRL.	Option `-co`[11]
Visual Appearance	Optional	See Creating a visual appearance of a signature.	

Proxy configuration If a proxy is used for the connection to the service, see Using a proxy for more information.

[10] This parameter is not used for certificate selection, but for the signature appearance and description in the PDF only.
[11] The recommendation is to not use the option `-co`.

Creating the SSL client certificate

When creating a new account, GlobalSign will issue an SSL client certificate `clientcert.crt`. The following command creates a PKCS#12 file `certificate.p12` that can be used for the `SSLClientCertificate`:

```
openssl pkcs12 -export -out certificate.p12 -inkey privateKey.key -in clientcert.crt
```

Getting the SSL server certificate

The SSL server certificate can either be found in the technical documentation of the "Digital Signing Service" or downloaded from the server itself:

1. Get the server's SSL certificate:

   ```
   openssl s_client -showcerts -connect emea.api.dss.globalsign.com:8443 ^
     -cert clientcert.crt -key privateKey.key
   ```

2. The certificate is the text starting with "`-----BEGIN CERTIFICATE-----`" and ending with "`-----END CERTIFICATE-----`". Use the text to create a text file and save it as `server.crt`.
3. Use `server.crt` or one of its CA certificates for the `SSLServerCertificate`.

Advice on using the service

There are rate limits for both creating new identities and for signing operations. If multiple documents must be signed at once, use the API version of the 3-Heights® PDF to PDF/A Converter Shell, which can re-use the same session (and hence its signing certificates) for signing.

Due to the short-lived nature of the signing certificates, it is important to embed revocation information immediately. For example, by using the option `-dss` of the 3-Heights® PDF Security Shell or not `-co`. Furthermore, it is highly recommended to embed a timestamp to prove that the signature was created during the certificate's validity period.

%

5 Creating digital signatures

This chapter describes the steps that are required to create different types of digital signatures. A good introductory example can be found in Creating electronic signatures.

5.1 Creating a PAdES signature

The PAdES European standard (ETSI EN 319 142) recommends that one of the following four baseline signature levels be used:

PAdES-B-B A digital signature.

PAdES-B-T A digital signature with a timestamp token.

PAdES-B-LT A digital signature with a timestamp token and signature validation data. The signature is a long-term signature or "LTV enabled".

PAdES-B-LTA A digital signature with a timestamp token and signature validation data protected by a document timestamp.

The lifecycle of digital signatures and the usage of these signature levels are described in more detail in chapter 8.11.6 "Digital signatures lifecycle" of ETSI TR 119 100.

> **Note:** The Decision 2015/1506/EU of the eIDAS Regulation (Regulation (EU) N°910/2014) still refers to the previous legacy PAdES baseline signature standard ETSI TS 103 172. However, the signatures as created by the 3-Heights® PDF to PDF/A Converter Shell are compatible.

The Compatibility of PAdES signature levels shows how the signature levels described above and as created by the 3-Heights® PDF to PDF/A Converter Shell conform with other standards.

Compatibility of PAdES signature levels

ETSI EN 319 142	ETSI TS 102 778	ETSI TS 103 172	ISO 14533-3
PAdES-B-B	PAdES-BES (Part 3)	PAdES B-Level	-
PAdES-B-T	PAdES-BES (Part 3)	PAdES T-Level	PAdES-T
PAdES-B-LT	PAdES-BES (Part 3)	PAdES LT-Level	PAdES-A
PAdES-B-LTA	PAdES-LTV (Part 4)	PAdES LTA-Level	PAdES-A

Requirements

For general requirements and preparation steps, see Creating electronic signatures.

Requirements

Level	Signing Certificate	Timestamp	Product
PAdES-B-B	any	no	3-Heights® PDF to PDF/A Converter Shell
PAdES-B-T	any	required	3-Heights® PDF to PDF/A Converter Shell
PAdES-B-LT	advanced or qualified certificate	required	**3-Heights® PDF Security**
PAdES-B-LTA	advanced or qualified certificate	required	**3-Heights® PDF Security**

Make sure the trust store of your cryptographic provider contains all certificates of the trust chain, including the root certificate. Also include the trust chain of the timestamp signature, if your TSA server does not include them in the timestamp.

A proper error handling is crucial in order to ensure the creation of correctly signed documents. The output document was signed successfully, if and only if the 3-Heights® PDF to PDF/A Converter Shell returns code 0 (success).

Note on linearization: Because signature levels PAdES-B-LT and PAdES-B-LTA must be created in a two-step process, the files cannot be linearized. When creating signature levels PAdES-B-B or PAdES-B-T that may later be augmented, linearization should not be used.

PAdES vs. CAdES: CAdES is an ETSI standard for the format of digital signatures. The format used in PAdES is based on CAdES, which is why the format is called `ETSI.CAdES.detached` (see `-st`). Because PAdES defines additional requirements suitable for PDF signatures, mere CAdES conformance is not sufficient.

5.1.1 Create a PAdES-B-B signature

Input document Any PDF document.

Cryptographic provider A cryptographic provider that supports the creation of PAdES signatures.

```
pdf2pdf -cp "myPKCS11.dll;0;pin" -cn "..." -st "ETSI.CAdES.detached" -co ^
input.pdf pades-b-b.pdf
```

5.1.2 Create a PAdES-B-T signature

Input document Any PDF document.

Cryptographic provider A cryptographic provider that supports the creation of PAdES signatures.

```
pdf2pdf -cp "myPKCS11.dll;0;pin" -cn "..." -st "ETSI.CAdES.detached" -co ^
-tsu "http://server.mydomain.com/tsa" input.pdf pades-b-t.pdf
```

5.2 Creating a visual appearance of a signature

Each signature may have a visual appearance on a page of the document. The visual appearance is optional and has no effect on the validity of the signature. Because of this and because a visual appearance may cover important content of the page, the 3-Heights® PDF to PDF/A Converter Shell creates invisible signatures by default.

To create a visual appearance, a non-empty signature rectangle must be set. For example, by setting the option `-ar 10 10 200 50` the following appearance is created:

Different properties of the visual appearance can be specified.

Page and position See options `-ap` and `-ar`.

Text Two text fragments can be set using two different fonts, font sizes, and colors see options `-at1`, `-at2`, `-af1`, `-af2`, `-afs1`, and `-afs2`.

Background image See options `-abg`.

5.3 Miscellaneous

5.3.1 Caching of CRLs, OCSP, and timestamp reponses

To improve the speed when mass signing, the 3-Heights® PDF to PDF/A Converter Shell provides a caching algorithm to store CRL (Certificate Revocation List), OCSP (Online Certificate Status Protocol), TSP (Timestamp Protocol) and data from signature services. This data is usually valid over period of time that is defined by the protocol, which is normally at least 24 hours. Caching improves the speed, because there are situations when the server does not need to be contacted for every digital signature.

The following caches are stored automatically by the 3-Heights® PDF to PDF/A Converter Shell at the indicated locations within the Cache directory:

Certificates	`<CacheDirectory>/Certificates/hash.cer`
CRL	`<CacheDirectory>/CLRs/server.der`
OCSP responses	`<CacheDirectory>/OCSP Responses/server-hash.der`
Service data	`<CacheDirectory>/Signature Sizes/hash.bin`
Timestamp responses[12]	`<CacheDirectory>/Time Stamps/server.der`

[12] The sizes of the timestamp responses are cached only. Cached timestamp responses cannot be embedded but used for the computation of the signature length only.

The caches can be cleared by deleting the files. Usage of the caches can be deactivated by setting the option `-nc`. The files are automatically updated if the current date and time exceeds the "next update" field in the OCSP or CRL response, respectively, or the cached data was downloaded more than 24 hours ago.

5.3.2 Using a proxy

The 3-Heights® PDF to PDF/A Converter Shell can use a proxy server for all communication to remote servers, e.g. to download CRL or for communication to a signature service. The proxy server can be configured using the provider session property `Proxy`. The property's value must be a string with the following syntax:

`http[s]://[‹user›[:‹password›]@‹host›[:‹port›]`

Where:

- `http` / `https`: Protocol for connection to proxy.
- `‹user›:‹password›` (optional): Credentials for connection to proxy (basic authorization).
- `‹host›`: Hostname of proxy.
- `‹port›`: Port for connection to proxy.

For SSL connections, e.g. to a signature service, the proxy must allow the HTTP CONNECT request to the signature service.

Example: Configuration of a proxy server that is called "myproxy" and accepts HTTP connections on port 8080.

```
-cps "Proxy" "http://myproxy:8080"
```

5.3.3 Configuring a proxy server and firewall

For the application of a timestamp or online verification of certificates, the signature software requires access to the server of the certificates' issuer (e.g. `http://ocsp.quovadisglobal.com` or `http://platinum-qualified-g2.ocsp.swisssign.net/`) via HTTP. The URL for verification is stored in the certificate; the URL for timestamp services is provided by the issuer. If these functions are not configured, no access is required.

In organizations where a web proxy is used, it must be ensured that the required MIME types are supported. These are:

OCSP

- `application/ocsp-request`
- `application/ocsp-response`

Timestamp

- `application/timestamp-query`
- `application/timestamp-reply`

Signature services

- Signature service-specific MIME types.

5.3.4 Setting the signature build properties

In the signature build properties dictionary, the name of the application that created the signature can be set using the provider session properties `Prop_Build.App.Name` and `Prop_Build.App.REx`. The default values are "3-Heights® PDF to PDF/A Converter Shell" and its version.

6 Validating digital signatures

6.1 Validating a qualified electronic signature

There are basically three items that need to be validated:

1. Trust chain
2. Revocation information (optional)
3. Timestamp (optional)

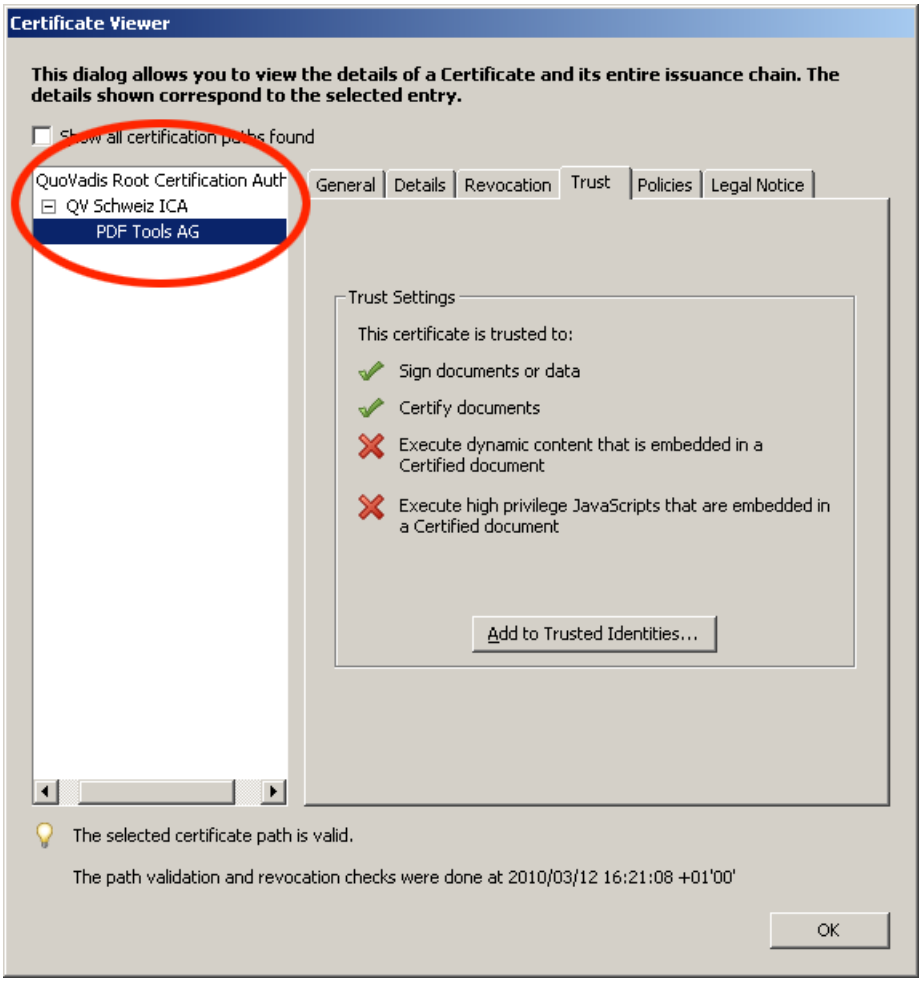
Validation can be done in different ways, e.g. Adobe Acrobat, from which the screenshots below are taken.

6.1.1 Trust chain

Before the trust chain can be validated, ensure the root certificate is trusted. There are different ways to add a certificate as trusted root certificate. The best way on Windows is this:

1. Retrieve a copy of the certificate containing a public key. This can be done be requesting it from the issuer (your CA) or by exporting it from an existing signature to a file (`CertExchange.cer`). Ensure you are not installing a malicious certificate!
2. Add the certificate to the trusted root certificates. If you have the certificate available as file, you can simply double-click it to install it.

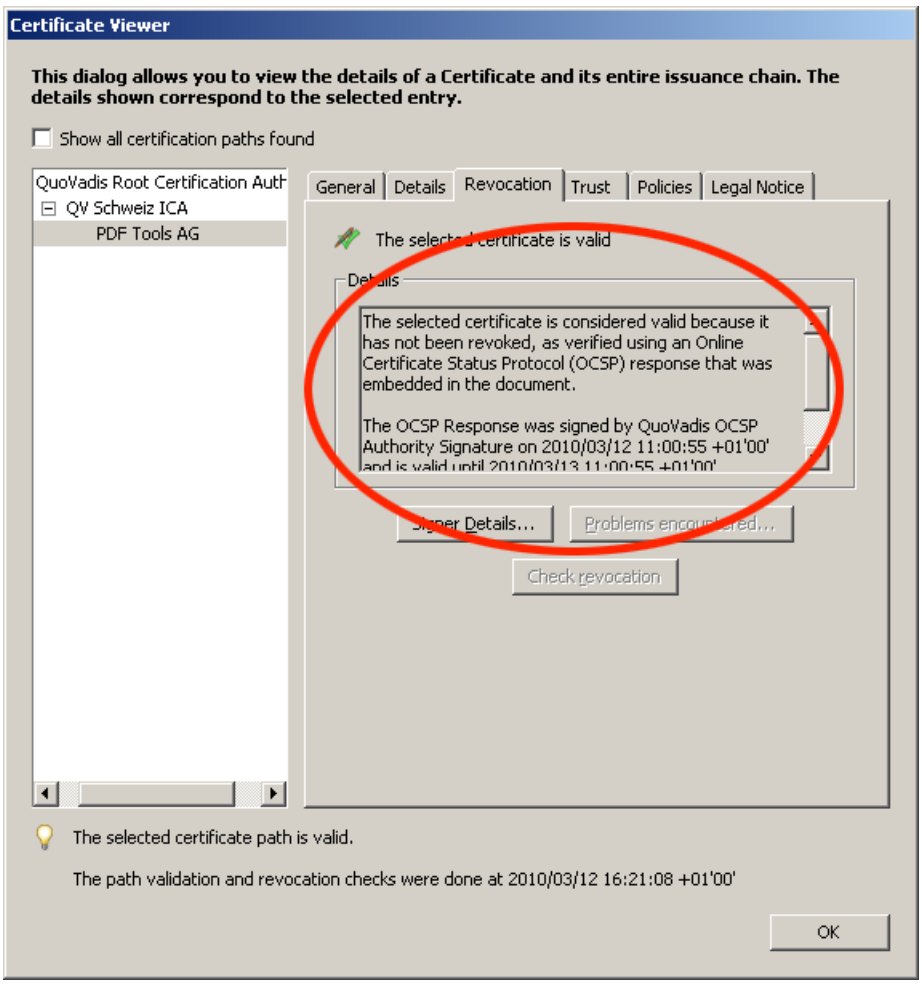
After that you can validate the signature, e.g. by open the PDF document in Adobe Acrobat, right-click the signature and select “Validate”, then select “Properties”, and select the tab “Trust”. There the certificate should be trusted to “sign documents or data”.



6.1.2 Revocation information

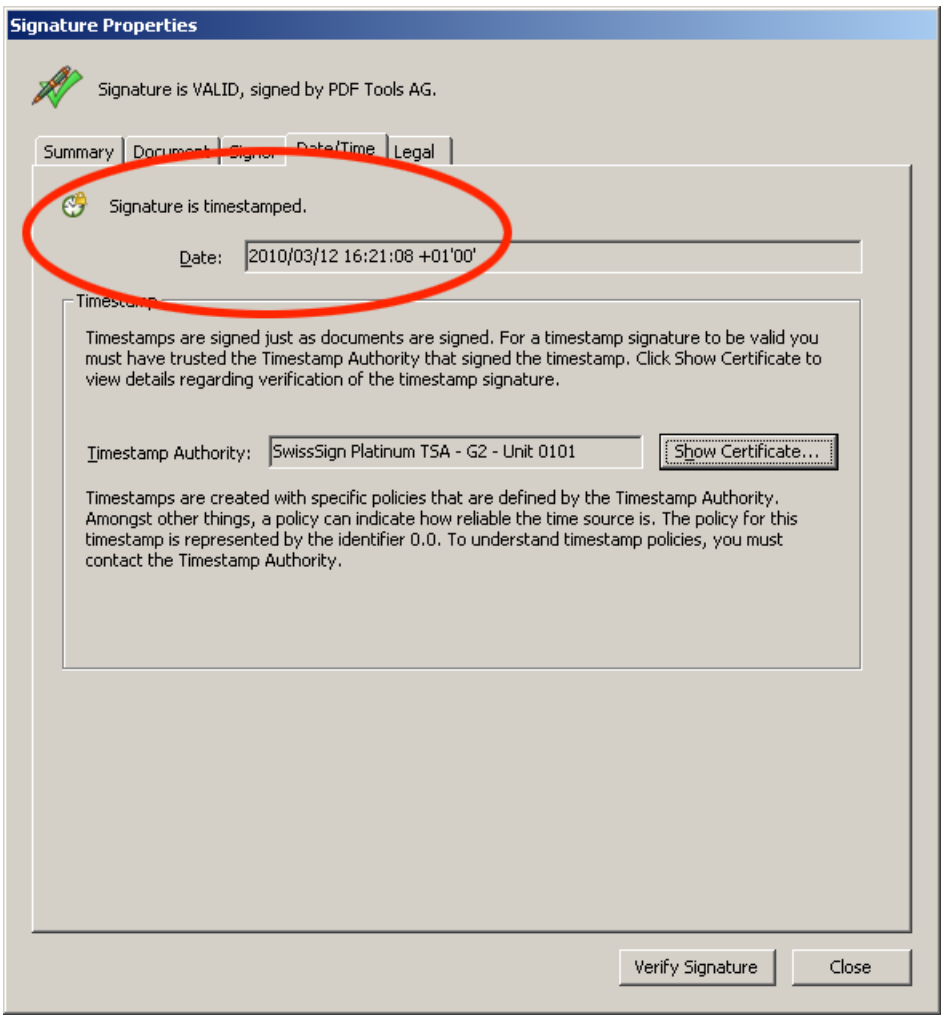
An OCSP response or CRL must be available. This is shown in the tab “Revocation”. The details should mention that “the certificate is considered *valid*”.

The presence of revocation information must be checked for the signing certificate and all certificates of its trust chain, except for the root certificate.



6.1.3 Timestamp

The signature can optionally contain a timestamp. This is shown in the tab “Date/Time”. The certificate of the time-stamp server must also be trusted, i.e. its trust chain should be validated as described in the section Trust Chain above.



6.2 Validating a PAdES LTV signature

Verifying if a signature conforms to the PAdES LTV standard is similar to validating a Qualified Electronic Signature.

The following must be checked:

1. Trust chain
2. Revocation information
3. Timestamp
4. LTV expiration date
5. Other PAdES requirements

6.2.1 Trust chain

Trust chain validation works the same as for validating Qualified Electronic Signatures.

6.2.2 Revocation information

Revocation information (OCPS response or CRL) must be valid and embedded into the signature. In the details, verify that the revocation check was performed using data that was "was *embedded in the signature* or *embedded in the document*". Revocation information that "was contained in the *local cache*" or "was requested *online*" is not embedded into the signature and does not meet PAdES LTV requirements. If Adobe Acrobat claims that revocation

information is contained in the local cache, even though it is embedded into the document, restart Adobe Acrobat and validate the signature again.

6.2.3 Timestamp

A timestamp must be embedded and validated as described for validating Qualified Electronic Signatures. If a document contains multiple timestamps, all but the latest one must contain revocation information.

6.2.4 LTV expiration date

The long-term validation ability expires with the expiration of the signing certificate of the latest timestamp.

The lifetime of the protection can be further extended beyond the life of the last timestamp applied by adding further DSS information to validate the previous last timestamp as well as a new timestamp. This process is described in Creating a PAdES signature.

6.2.5 Other PAdES requirements

Certain other PAdES requirements, such as requirements on the PKCS#7 CMS, cannot be validated using Adobe Acrobat. For this, use the 3-Heights® PDF Security API for validation.

7 Interface reference

7.1 General settings

7.1.1 -ad Allow downgrade of PDF/A conformance level

Allow downgrade of PDF/A conformance level `-ad`

If this option is set, automatic downgrade of the PDF/A conformance level is allowed, e.g. from PDF/A-1a to PDF/A-1b.

The level is downgraded under the following conditions:

- Downgrade to level B: If a file contains text that is not extractable (i.e. missing ToUnicode information).
  Example: Downgrade PDF/A-2u to PDF/A-2b.
- Downgrade to level U (PDF/A-2 and PDF/A-3) or B (PDF/A-1): Level A requires logical structure information ("tagging") information, so if a file contains no such information, its level is downgraded.
  Logical structure information in a PDF defines the structure of content, such as titles, paragraphs, figures, reading order, tables or articles. Logical structure elements can be "tagged" with descriptions or alternative text. "Tagging" allows the contents of an image to be described to the visually impaired.
  It is not possible for the 3-Heights® PDF to PDF/A Converter Shell to add meaningful tagging information. Adding tagging information without prior knowledge about the input file's structure and content is neither possible nor allowed by the PDF/A standard. For that reason, the conformance level is automatically downgraded to level B or U.
  Example: Downgrade PDF/A-1a to PDF/A-1b.

If this option is not set and an input file cannot be converted to the requested standard, e.g. because of missing "tagging" information, the conversion is aborted and the error code 4 is returned.

7.1.2 -au Allow upgrade from PDF/A-1 to PDF/A-2

Allow upgrade from PDF/A-1 to PDF/A-2 `-au`

If this option is set, automatic upgrade of the PDF/A version is allowed. If the target standard is PDF/A-1 and a file contains elements that cannot be converted to PDF/A-1, the target standard is upgraded to PDF/A-2. This avoids significant visual differences in the output file.

For example, the following elements may lead to an automatic upgrade:

- Transparency
- Optional content groups (Layers)
- Real values that exceed the implementation limit of PDF/A-1
- Embedded OpenType font files
- Predefined CMap encodings in Type0 fonts

If this option is not set, the conformance is not upgraded. Depending on the value of the conversion error mask (`-cem`), the conversion will fail with a conversion error (return code 5).

7.1.3 -af Add associated file

Add associated file `-af ‹i;r;m;d;f›`

Add a file to the document's embedded files. For PDF/A-3, the embedded file is associated with an object of the document, i.e. it is an associated file.

The file is embedded as-is. Embedding files is not allowed for PDF/A-1 and restricted to PDF/A conforming files for PDF/A-2.

Description of the parameter ‹i;r;m;d;f›

i The object to associate the embedded file with. `-1` for none, `0` for document, number greater than `0` for respective page. Default: `0` for PDF/A-3 and `-1` otherwise.

r The relationship of the embedded file to the object associated, PDF/A-3 only. Allowed values are `Source`, `Data`, `Alternative`, `Supplement`, and `Unspecified`. Default: `Unspecified`.

m Mime-type of the embedded file. Default: `application/octet-stream`. Other common values are `application/pdf`, `application/xml`, or `application/msword`.

d A description of the embedded file. This is presented to the user when viewing the list of embedded files.

f The path (or URL) to the file to be embedded.

```
pdf2pdf -cl pdfa-3a -af "0;Source;application/msword;The source ^
document;input.doc" input.pdf output.pdf
```

7.1.4 -ai Add an XML invoice file (Factur-X or ZUGFeRD)

Add an XML invoice file (Factur-X or ZUGFeRD) `-ai ‹type›;‹file›;‹afrel›`

Add an XML invoice file (Factur-X or ZUGFeRD).

> **Note:** This feature requires the conformance to be set to PDF/A-3.

If the specified XML invoice file cannot be added during conversion, conversion is aborted with an error. This can happen either if the invoice type cannot be determined unambiguously from the XML and there is no clear preference, or if the chosen invoice type is in direct contradiction to the XML itself.

Other than those basic checks, the XML invoice is not validated against any standard or schema.

Parameters:

`‹type›` The type of invoice.

Possible values `zf` or `fx` for ZUGFeRD/Factur-X with automatic profile selection, `zf1bas` (BASIC), `zf1com` (COMFORT), `zf1ext` (EXTENDED) for specific ZUGFerd 1.0 profiles, `zf2min` (MINIMUM), `zf2bwl` (BASIC WL), `zf2bas` (BASIC), `zf2en` (EN 16931), `zf2ext` (EXTENDED), `zf2xr` (XRECHNUNG) for specific ZUGFerd 2 (2.0 / 2.1) profiles, `zf20min` (MINIMUM), `zf20bwl` (BASIC WL), `zf20bas` (BASIC), `zf20en` (EN 16931), `zf20ext` (EXTENDED)

for specific ZUGFerd 2.0 profiles, `zf21min` (MINIMUM), `zf21bwl` (BASIC WL), `zf21bas` (BASIC), `zf21en` (EN 16931), `zf21ext` (EXTENDED), `zf21xr` (XRECHNUNG) for specific ZUGFerd 2.1 profiles, `fx1min` (MINIMUM), `fx1bwl` (BASIC WL), `fx1bas` (BASIC), `fx1en` (EN 16931), `fx1ext` (EXTENDED) for specific Factur-X 1.0 profiles.

‹file› The path to the XML invoice file.

‹afrel› Optional AFRelationship value.

The AFRelationship determines the relation of the invoice file to the PDF. Allowed values are `Source`, `Data`, `Alternative`, `Supplement` and `Unspecified`.

If no value is provided, a sensible default value is chosen.

Some invoice standards restrict the set of allowed values for certain profiles.

```
pdf2pdf -cl pdfa-3b -ai "zf;invoice.xml" input.pdf output.pdf
pdf2pdf -cl pdfa-3b -ai "fx1min;invoice.xml;Supplement" input.pdf output.pdf
```

7.1.5 -az Add a Factur-X or ZUGFeRD XML invoice file

[Deprecated] Add a Factur-X or ZUGFeRD XML invoice file `-az ‹file›`

This property was deprecated in version 5.5, use option `-ai` instead.

Equivalent command for ZUGFeRD invoices:

```
pdf2pdf -cl pdfa-3b -ai "zf;zugferd-invoice.xml" input.pdf output.pdf
```

Equivalent command for Factur-X invoices:

```
pdf2pdf -cl pdfa-3b -ai "fx;factur-x.xml" input.pdf output.pdf
```

7.1.6 -ef Add embedded file

Add embedded file `-ef ‹file›`

`-ef "file"` is the same as `-af ";;;;file"`.

7.1.7 -ax Add XMP metadata

Add XMP metadata `-ax ‹file›`

Add XMP metadata from a file. Providing a path that does not exist or an invalid XMP file results in return code 3.

```
pdf2pdf -ax metadata.xml input.pdf output.pdf
```

The following metadata properties may be modified by the 3-Heights® PDF to PDF/A Converter Shell:

- `pdf:Producer`, `xmp:ModifyDate`, and `xmp:MetadataDate`
- Properties from the PDF/A Identification (pdfaid) schema and `pdf:PDFVersion`
- Keys set using option `-id` override the corresponding values in the XMP metadata stream

7.1.8 -ma Analyze the input file

Analyze the input file -ma

Analyze the input file and verify if it meets a certain conformance level. To get a report, either the option -rs or -rd can be used in combination. Otherwise, only the return code it set.

```
pdf2pdf -ma -rs input.pdf output.pdf
The document contains fonts without embedded font programs or encoding
information (CMAPs).
The documents metadata is either missing or inconsistent
or corrupt.
```

An output file name must be provided, since the output name also specifies the name of the log file that is generated. However, no output PDF document is created. The analysis is similar to the analysis using the 3-Heights® PDF Validator. However, the 3-Heights® PDF to PDF/A Converter Shell is more strict in certain issues, especially concerning those corner cases of the PDF/A ISO Standard, in which a conversion is strongly advised.

7.1.9 -cff Embed Type1 fonts as CFF

Embed Type1 fonts as CFF -cff

Convert Type1 (PostScript) fonts to Compact font format before embedding. This reduces the file size. This affects the embedding of fonts only, existing Type1 fonts of the input document will not be converted.

7.1.10 -mc Force conversion even if there are analysis errors

Force conversion even if there are analysis errors -mc

Setting this option forces the conversion even if the input file already conforms to the requested standard.

7.1.11 -q Image quality

Image quality -q ‹n›

Set or get the image quality index for images that use a prohibited lossy compression type and must be re-compressed.

Supported values are 1 to 100. A higher value means better visual quality at the cost of a larger file size. Recommended values range from 70 to 90. The default value is 80.

Example:

JPX (JPEG2000) is not allowed in PDF/A-1. If a PDF contains a JPX compressed image, its compression type must be altered. Thus the 3-Heights® PDF to PDF/A Converter Shell converts it to an image with regular JPEG compression and the image quality as defined by this switch.

7.1.12 -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.

```
pdf2pdf -lk X-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX ...
```

This is required in an OEM scenario only.

7.1.13 -cem Mask conversion errors

Mask conversion errors `-cem ‹mask›`

The conversion error mask defines which operations and conditions are not allowed and consequently, cause a conversion error (return value 5).

Values for ‹mask›

Value	Description
0	None (never return a conversion error)
4	(default) Visual differences in output file.
8	Resolve name collisions of colorants (PDF/A-2 and PDF/A-3 only).
16	(default) Remove optional content groups (layers) (PDF/A-1 only).
32	(default) Remove transparency (PDF/A-1 only).
64	(default) Remove embedded files.
128	(default) Remove non convertible XMP metadata.
512	Conversion of signed document forced removal of signatures.
4096	(default) The input document is corrupt and should be repaired. The errors encountered are printed to the log file. Some errors can be repaired, but it is crucial to review the output file and perform the post-analysis.
16384	Font substituted.
32768	Remove interactive elements such as actions or annotations.
65536	Remove logical structure information.

Add up all operations are conditions to define the conversion mask. The default is 4340. In order to accept the removal of XMP metadata, set the mask to 4212.

```
pdf2pdf -cem 4212 input.pdf output.pdf
```

See chapter Conversion errors for more information on conversion errors and how they can be handled.

7.1.14 -cef Try to convert embedded PDF documents (PDF/A-3 only)

Try to convert embedded PDF documents (PDF/A-3 only) `-cef`

By default, embedded files are copied as-is during conversion to PDF/A-3. If this option is used, the 3-Heights® PDF to PDF/A Converter Shell tries to convert embedded PDF documents to PDF/A. The converted document is embedded only, if the conversion was successful.

This option is relevant for PDF/A-3 only. During conversion to PDF/A-1, all embedded files are removed, whereas conversion to PDF/A-2 converts all embedded files to PDF/A.

7.1.15 -ow Optimize for the web

Optimize for the web `-ow`

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

Linearize the PDF output file, i.e. optimize file for fast web access.

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

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

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

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

> **Note:** To use a linearized PDF file, the PDF must reside as a "file" on the web server. It must not be streamed.

The 3-Heights® PDF to PDF/A Converter Shell cannot linearize signed files. So this option should not be used if a digital signature is applied. Linearizing signed files is possible with the 3-Heights® PDF Security product.

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

7.1.16 -p Read an encrypted PDF file

Read an encrypted PDF file `-p ‹password›`

When the input PDF file is encrypted and has a user password set, the password to open the PDF can be provided with the option `-p`. For example, if the user password were `"userpwd"`, then the command would look like this:

```
pdf2pdf -p userpwd input.pdf output.pdf
```

When a PDF is encrypted and the user password is not provided or is incorrect, `pdf2pdf` cannot decrypt and read the file. Instead it generates the following error message:

```
Cannot open file input.pdf.
```

7.1.17 `-rd` Report conformance violations in detail

Report conformance violations in detail `-rd`

This option lists all conformance violations per page. Each violation is listed with a page number (page 0 = document level), error number, a description, and a counter of how many times the error occurs. The option provides more detailed information than the summary (`-rs`), for both the pre- and the post-analysis.

Example:

```
pdf2pdf -ma -rd input.pdf output.pdf
- Opening file input.pdf.
- Analyzing input.pdf.
  "input.pdf", 0, 53, 0x0A09C882, "The property 'xmp:Format' is not defined in schema
'XMP Basic Schema'.", 1
  "input.pdf", 1, 4, 0x03418614, "A device-specific color space (DeviceGray) without an
appropriate output intent is used.", 1
  "input.pdf", 4, 10, 0x00418704, "The font Arial-BoldMT must be embedded.", 1
  "input.pdf", 4, 0, 0x83410612, "The document does not conform to the requested standard.",
1
```

7.1.18 `-rs` Report conformance violations summary

Report conformance violations summary `-rs`

This option gives a summary of all conformance violations. If any of the following violations is detected at least once, it is reported (once). This option provides less detailed information than the detailed list per page (`-rd`) for both the pre- and the post-analysis.

- The file format (header, trailer, objects, xref, streams) is corrupted.
- The document doesn't conform to the PDF reference (missing required entries, wrong value types, etc.).
- The file is encrypted and the password was not provided.
- The document contains device-specific color spaces.
- The document contains illegal rendering hints (unknown intents, interpolation, transfer and halftone functions).
- The document contains alternate information (images).
- The document contains embedded PostScript code.
- The document contains references to external content (reference XObjects, file attachments, OPI).
- The document contains fonts without embedded font programs or encoding information (CMAPs).
- The document contains fonts without appropriate character to Unicode mapping information (ToUnicode maps).
- The document contains transparency.
- The document contains unknown annotation types.

- The document contains multimedia annotations (sound, movies).
- The document contains hidden, invisible, non-viewable or non-printable annotations.
- The document contains annotations or form fields with ambiguous or without appropriate appearances.
- The document contains actions types other than for navigation (launch, JavaScript, ResetForm, etc.).
- The document's meta data is either missing or inconsistent or corrupt.
- The document doesn't provide appropriate logical structure information.
- The document contains optional content (layers).

Example:

```
pdf2pdf -ma -rs input.pdf output.pdf
- Opening file input.pdf.
- Analyzing input.pdf.
  The document contains device-specific color spaces.
  The document contains fonts without embedded font programs or encoding information
(CMAPs).
  The document's metadata is either missing or inconsistent or corrupt.
```

7.1.19 -cl Set conformance

Set conformance `-cl ‹conf›`

Set the PDF/A conformance level.

Supported values for the parameters `‹conf›` are: `pdfa-1b`, `pdfa-1a`, `pdfa-2b`, `pdfa-2u`, `pdfa-2a`, `pdfa-3b`, `pdfa-3u`, `pdfa-3a`. The default value is `pdfa-2b`.

Some files cannot be converted to the conformance requested. The 3-Heights® PDF to PDF/A Converter Shell can detect this and upgrade or downgrade the conformance automatically. Use the options `-au` and `-ad` to allow automatic upgrades or downgrades.

The following example sets the conformance level to PDF/A-2u:

```
pdf2pdf -cl pdfa-2u input.pdf output.pdf
```

7.1.20 -fd Add font directory

Add font directory `-fd ‹dir›`

All fonts must be embedded in order to create a valid PDF/A. If the input file contains a font which is not embedded, the font folder is searched for a font with the same name. If such a font is found, the font is embedded.

This option can be used to add (multiple) font directories to the search path for fonts.

In addition to directories added with this option, the default font directories are always considered.

```
pdf2pdf -fd C:\MyFonts input.pdf output.pdf
```

See chapter Fonts for more information on the font directories and font handling of the 3-Heights® PDF to PDF/A Converter Shell, in general.

7.1.21 -id Set value in the document information dictionary

Set value in the document information dictionary `-id ‹key› ‹value›`

Set the value of an document information dictionary entry `‹key›`. Popular entries specified in the PDF Reference 1.7 are `"Title"`, `"Author"`, `"Subject"`, `"Creator"` (sometimes referred to as Application), and `"Producer"` (sometimes referred to as PDF Creator). If the entry already exists then the previous entry is overwritten. If the key corresponds to a standard metadata key, then the XMP metadata is updated accordingly.

Example: Overwrite the default producer:

```
pdf2pdf -id Producer "MyProgram 1.2" input.pdf output.pdf
```

7.1.22 -uf Update the font Unicode characters

Update the font Unicode characters `-uf ‹file›`

Update the fonts' Unicode code points as specified by the `‹file›` parameter. The file must contain the mapping of character codes to Unicode characters for specific fonts.

7.1.23 -v Verbose mode

Verbose mode `-v`

This option turns on the verbose mode. In verbose mode, the log output is also written to the console.

7.2 Processing files in a directory

Wildcards return a list of existing files. If you wante to convert all files in a directory to individual PDF/A documents, you must use a variable to name the output files. Here is an example for the `for`-command of the Windows CMD-shell. It converts all PDF files to PDF/A documents with the same name and the extension `.pdf`, in the same folder:

```
for %i in (*.pdf) do pdf2pdf -v %i %~ni.pdf
```

Of course, you can adjust the paths, or use a different output name:

```
for %i in (.\input\*.pdf) do pdf2pdf %i.\output\new_%~ni.pdf
```

For additional help on the `for`-command, use the command:

```
for /?
```

Note: Variables used in a batch file require two leading `%` instead of one.

7.3 Color profiles

See the dedicated section Color spaces for more information on the topic.

7.3.1 -cs ICC profile for device-specific color spaces

ICC profile for device-specific color spaces `-cs ‹profile›`

This ICC profile represents the color profile of the scanner. It is required if a color space is used that is different from color ICC profile of the output intent. Initially, there is a default color profile for RGB (sRGB) and CMYK (`USWebCoatedSWOP.icc`) defined in the 3-Heights® PDF to PDF/A Converter. This switch can be used to set both, the RGB and the CMYK color profile. If an RGB color profile is passed as argument, it is set as new RGB color space. If a CMYK color is provided, it is set as new CMYK color space. If an invalid file is provided, it results in error code 3. To set the color profile for both color spaces, use the switch `-cs` twice.

The following command sets the standard sRGB color profile as color space:

```
pdf2pdf -cs "C:\Windows\system32\spool\drivers\color\sRGB Color Space
Profile.icm" input.pdf output.pdf
```

If a required color space profile is not available, a device-independent color space is created automatically.

7.3.2 -oi ICC profile for output intent

ICC profile for output intent `-oi ‹profile›`

The ICC profile for the output intent describes the color profile of the device (monitor or display). An output intent is required for PDF/A compatibility as described in PDF/A requirements.

Providing a path that does not exist or an invalid ICC color profile file results in return code 3.

It is not recommended to set an output intent using this option. Instead, the ICC profiles for device-specific color spaces should be set using the option `-cs`, which ensures an optimal result of the automatic color conversion algorithm (see PDF/A requirements).

```
pdf2pdf -oi "C:\Windows\system32\spool\drivers\color\sRGB Color Space
Profile.icm" input.pdf output.pdf
```

7.4 Digital signatures

For more information on digital signatures in general, see section Digital signatures. For more information on how to create digital signatures, see section Creating digital signatures.

7.4.1 -abg Signature background image

Signature background image `-abg ‹image›`

This is the background image that is added to the signature. The image is centered and scaled down proportionally to fit into the given rectangle. If the path is `Nothing`, or the image does not exist, the appearance's background is a filled rectangle using the colors fill color and stroke color.

To create a signature with the image only, set the signature text 1 and 2 to a space `" "`.

7.4.2 -af1 Signature font name 1

Signature font name 1 `-af1 ‹font name›`

This defines the font used in upper text, i.e. the text that is set by the property `-at1`. The font can either be specified as a path to the font file, e.g. `"C:\Windows\Fonts\arial.ttf"`, or as a font name, such as `"Times New Roman,Bold"`. When using a font name, the corresponding font must be present in one of the font directories described in Fonts.

7.4.3 -af2 Signature font name 2

Signature font name 2 `-af2 ‹font name›`

This is the font used in lower text, i.e. the text that is set by `-at2`. The option works analogously to `-af1`.

7.4.4 -afs1 Signature font size 1

Signature font size 1 `-afs1 ‹font size›`

This defines the font size in points used in upper text, i.e. the text that is set by the property `-at1`. If the font size is not specified, a default value of `16pt` is used.

7.4.5 -afs2 Signature font size 2

Signature font size 2 `-afs2 ‹font size›`

This is the font size in points used in lower text, i.e. the text that is set by `-at2`. The option works analogously to `-afs1`. If the font size is not specified, a default value of `8pt` is used.

7.4.6 -ap Signature page number

Signature page number `-ap ‹page›`

Set the page number of where the visual appearance of the digital signature should be placed. The numbers are counted starting from 1 for the first page. The default is the last page. The last page can also be set using `-1` as argument.

7.4.7 -ar Signature annotation rectangle

Signature annotation rectangle `-ar ‹x› ‹y› ‹w› ‹h›`

Set the position and size of the digital signature annotation. The default is an invisible signature (`-ar 0 0 0 0`).

The position is defined by the four values for the lower-left corner (x, y) and dimensions (w, h) of the rectangle. The units are PDF points (1 point = 1/72 inch, A4 = 595 x 842 points, Letter = 612 x 792 points) measured from the lower left corner of the page. If either the width or height is zero or negative, an invisible signature is created, i.e. no visible appearance is created for the signature.

Example: Create a 200 by 60 points rectangle in the upper left corner of an A4 page.

```
pdf2pdf -cn "..." -ar 10 770 200 60 input.pdf output.pdf
```

7.4.8 `-at1` Signature text 1

Signature text 1 `-at1 ‹text›`

This is the upper text that is added to the signature.

If this property is not set, the signing certificate's name set with `-cn` is added to the upper text line of the visual signature.

See Creating a visual appearance of a signature for more information on customizing the appearance of digital signatures.

7.4.9 `-at2` Signature text 2

Signature text 2 `-at2 ‹text›`

This is the lower text that is added to the signature. The text can be multi-lined by using carriage returns.

If this property is not set, a three-line text is constructed that consists of:

- A statement who applied to signature
- The reason of the signature. This can be set using `-cr`.
- The date

See Creating a visual appearance of a signature for more information on customizing the appearance of digital signatures.

7.4.10 `-cci` Signer contact info

Signer contact info `-cci ‹info›`

Add a descriptive text as signer contact info, e.g. a phone number. This enables a recipient to contact the signer to verify the signature. This is not required in order to create a valid signature.

7.4.11 `-cfp` Certificate fingerprint

Certificate fingerprint `-cfp ‹fp›`
License feature: `Signature`

Set the hex string representation of the signer certificate's sha1 fingerprint. All characters outside the ranges `0-9`, `a-f` and `A-F` are ignored. In the Microsoft Management Console, the "Thumbprint" value can be used without conversion, if the "Thumbprint algorithm" is "sha1". E.g. "`b5 e4 5c 98 5a 7e 05 ff f4 c6 a3 45 13 48 0b c6 9d e4 5d f5`". This property can be used to select the signer certificate for signing (see Cryptographic provider).

7.4.12 `-ci` Certificate issuer

Certificate issuer `-ci ‹issuer›`
License feature: `Signature`

The issuer of the certificate. The "Certificate Issuer" corresponds to the common name (CN) of the issuer. In the Windows certificate store, this corresponds to "Issued by". This property can be used to select the signer certificate for signing (see Cryptographic provider).

7.4.13 `-cn` Certificate name (Subject)

Certificate name (Subject) `-cn ‹name›`
License feature: `Signature`

Set the name of the certificate used to sign the document (see Cryptographic provider). The name corresponds to the common name (CN) of the subject. In the Windows certificate store, this corresponds to "Issued to".

See Digital signatures to learn more about digital signatures in general and how to sign documents with the 3-Heights® PDF to PDF/A Converter Shell.

Example: Sign the document

```
pdf2pdf -cn "Philip Renggli" input.pdf output.pdf
```

The signature is added on the last page of the signed document.

7.4.14 `-cno` Certificate serial number

Certificate serial number `-cno ‹serialno›`
License feature: `Signature`

Set the serial number of the certificate. Specify a hex string as displayed by the "Serial number" field in the Microsoft Management Console (MMC), e.g. "`49 cf 7d d1 6c a9`". This property can be used to select the signer certificate for signing (see Cryptographic provider).

7.4.15 `-co` Do not embed revocation information

Do not embed revocation information `-co`

This switch inhibits the embedding of revocation information such as online certificate status response (OCSP - RFC 2560) and certificate revocation lists (CRL - RFC 3280). Revocation information is either an OCSP response or a CRL, which is provided by a validation service at the time of signing and acts as proof that at the time of signing

the certificate is valid. This is useful because even when the certificates expires or is revoked at a later time, the signature in the signed document remains valid.

Embedding revocation information is optional but suggested when applying advanced or qualified electronic signatures.

This option is not supported by all cryptographic providers and never for document timestamp signatures.

Revocation information is embedded for the signing certificate and all certificates of its trust chain. This implies that both OCSP responses and CRLs can be present in the same message.

The downsides of embedding revocation information are the increase of the file size (normally by around 20 KB) and that it requires a connection to a validation service, which delays the process of signing. For mass signing, it is suggested to use the caching mechanism, see Caching of CRLs, OCSP, and timestamp reponses.

Embedding revocation information requires an online connection to the CA that issues them. The firewall must be configured accordingly. In case a web proxy is used, it must be ensured the following MIME types are supported when using OCSP (not required for CRL):

`application/ocsp-request`

`application/ocsp-request`

7.4.16 `-cp` Cryptographic provider

Cryptographic provider `-cp ‹prov›`
License feature: `Signature`

This property specifies the cryptographic provider used to create and verify signatures.

For more information on the different providers available, see Cryptographic provider.

- When using the Windows Cryptographic Provider, the value of this property is set to a string with the following syntax:
  `"[ProviderType:]Provider[;PIN]"`
  If the name of the provider is omitted, the default provider is used.

  Examples: `"123456"` being the PIN code

  ```
  Provider = "Microsoft Base Cryptographic Provider v1.0;123456"
  ```

  ```
  Provider = ";123456"
  ```

- When using the PKCS#11 provider, the value of this property is set to a string with the following syntax:
  `"PathToDll;SlotId;Pin"`

  Example:

  ```
  Provider = "\WINDOWS\system32\siecap11.dll;4;123456"
  ```

- When using any of the service providers, such as the “Swisscom All-in signing service”, the value of this property is essentially the URL of the service endpoint:
  `"http[s]://server.servicedomain.com:8080/url"`

7.4.17 -cpf Cryptographic session property (file)

Cryptographic session property (file) `-cpf ‹name› ‹file›`

File data property for configuring cryptographic session. The supported names and values are specific to the cryptographic provider.

7.4.18 -cps Cryptographic session property (string)

Cryptographic session property (string) `-cps ‹name› ‹str›`

String property for configuring cryptographic session. The supported names and values are specific to the cryptographic provider.

7.4.19 -cr Signature reason

Signature reason `-cr ‹reason›`

Add a descriptive text about the reason why the document was signed.

Example: Sign the document an add a reason text.

```
pdf2pdf -cn "Philip Renggli" -cr "Review and approval" -ar 10 10 200 50 ^
input.pdf output.pdf
```

The signature of the resulting output looks as shown below:

7.4.20 -csl Certificate store location

Certificate store location `-csl ‹location›`

For the Windows Cryptographic Provider, this defines the location of the certificate store from where the signing certificate should be taken. Supported are:

0 Local Machine.

1 Current User (default).

For more information, see Windows Cryptographic Provider.

7.4.21 -csn Certificate store name

Certificate store name `-csn ‹store›`

For the Windows Cryptographic Provider, this defines the certificate store from where the signing certificate should be taken. This depends on the operating system. The default is `"MY"`. Other supported values are: `"CA"` or `"ROOT"`.

Example: Use the certificate store ROOT from the Local Machine account.

```
pdf2pdf -cn "..." -csn ROOT -csl 0 input.pdf output.pdf
```

7.4.22 -nc Disable cache for CRL and OCSP

Disable cache for CRL and OCSP `-nc`

Get or set whether to disable the cache for CRL and OCSP responses.

Using the cache is safe, since the responses are cached as long as they are valid only. The option affects both signature creation and validation.

See Caching of CRLs, OCSP, and timestamp reponses for more information on the caches.

7.4.23 -nd Disable the use of DSS when signing documents

Disable the use of DSS when signing documents `-nd`

Use this option to avoid embedding revocation information (OCSP, CRL, and trust chain) in the document security store (DSS) when signing documents. This is to work around issues with legacy software that does not support the DSS. The use of the DSS is recommended for long-term (LTV) signatures.

7.4.24 -st Set signature subfilter

Set signature subfilter `-st ‹subfilter›`

The `‹subfilter›` indicates the encoding of the signature. The following are common values for `‹subfilter›`:

`adbe.pkcs7.detached` (PDF 1.6) Legacy PAdES Basic (ETSI TS 102 778, Part 2) signature used for document signatures.

`ETSI.CAdES.detached` (PDF 2.0) PAdES signature as specified by European Norm ETSI EN 319 142. This type is used for document signatures. See Creating a PAdES signature for more information.

7.4.25 -tsc Timestamp credentials

Timestamp credentials `-tsc ‹cred›`

If a timestamp server requires authentication, use this switch to provide the credentials.

Example: Credentials commonly have the syntax `username:password`.

```
pdf2pdf -cn "..." -tsu http://mytimestamp.com -tsc username:password
input.pdf output.pdf
```

7.4.26 `-tsu` Timestamp URL

Timestamp URL `-tsu ‹url›`

The URL of the trusted timestamp server (TSA) from which a timestamp is acquired. This setting is only required when applying a Qualified Electronic Signature. Applying a timestamp requires an online connection to a timestamp server; the firewall must be configured accordingly. In case a web proxy is used, it must be ensured the following MIME types are supported:

`application/timestamp-query`

`application/timestamp-reply`

7.4.27 `-wpc` Web proxy server credentials

Web proxy server credentials `-wpc ‹cred›`

If a web proxy server is used, and it requires authentication, use this switch and the syntax `user:password`.

Example: Set a web proxy server URL and use authentication.

```
pdf2pdf -wpu "http://proxy.example.org" -wpc user:password input.pdf
output.pdf
```

7.4.28 `-wpu` Web proxy server URL

Web proxy server URL `-wpu ‹url›`

In an organization where a web proxy server is in use, it must be ensured this web proxy server is specified. The URL is something like `"http://proxy.example.org"` or an IP address. For more information, see Using a proxy.

7.5 OCR

In order to make use of OCR, an OCR engine must be installed. The OCR engine is provided as part of a separate product: The 3-Heights® OCR Enterprise Add-on.

The recommended options (besides `-ocr`, `-ocl` and `-ocp`) are:

- For scanned documents: `-oca -ocri -occs`
- For born-digital documents: none

7.5.1 -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 to PDF/A Converter 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:

```
pdf2pdf -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`.

7.5.2 -oca Rotate the image to the detected angle

Rotate the image to the detected angle `-oca`

The OCR engine may detect that an image needs to be rotated to have the text in an up-right position. If this is the case and this switch is used, then the original image is replaced by the rotated image.

7.5.3 -ocb Convert images to bitonal before OCR recognition

Convert images to bitonal before OCR recognition `-ocb`

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

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

7.5.4 -ocbc Embed barcodes

Embed barcodes `-ocbc`

Embed the recognized barcodes in the XMP metadata.

7.5.5 -occs Correct skew angle

Correct skew angle `-occs`

Correct the skew angle of images.

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

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

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

7.5.6 -ocd Resolution for OCR recognition

Resolution for OCR recognition `-ocd ‹n›`

Re-sample images to target resolution before they are sent to the OCR engine. If no value is set, images are re-sampled to 300 DPI for OCR, which is the preferred resolution for most OCR engines.

7.5.7 -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:

```
pdf2pdf -ocr abbyy11 -ocl "English, German" input.pdf output.pdf
```

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

7.5.8 -ocm OCR mode

OCR mode `-ocm ‹n›`

Specify behavior of the 3-Heights® PDF to PDF/A Converter Shell for files with existing OCR text. Available OCR modes are the following:

1 Only perform OCR for images without existing OCR text (default).

2 If OCR engine is active, remove old OCR text and perform OCR for all images. Hence, existing OCR text is not removed if OCR engine is not active.

3 Always remove old OCR text and, if OCR engine is active, perform OCR for all images. This can be used to strip existing OCR text, without adding new OCR text.

4 Only perform OCR if input file contains no text.

Example: Set OCR mode 2

```
pdf2pdf -ocr abbyy11 -ocm 2 input.pdf output.pdf
```

7.5.9 -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:

```
pdf2pdf -ocr abbyy11 -ocp "PredefinedProfile = DocumentArchiving_Accuracy"
input.pdf output.pdf
```

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

7.5.10 -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

```
pdf2pdf -ocr service input.pdf output.pdf
```

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

7.5.11 -ocri Reembed pre-processed image

Reembed pre-processed image `-ocri`

This option currently requires the `-occs` to be set.

The OCR engine deskews and denoises the input image before recognizing the characters. This option controls whether the 3-Heights® PDF to PDF/A Converter Shell should use the pre-processed image or keep the original image.

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

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

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

7.5.12 -oct Threshold resolution for OCR

Threshold resolution for OCR -oct ‹n›

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

Example: Re-sample all images with a resolution of more than 300 DPI to 300 DPI:

```
pdf2pdf -ocd 300 -oct 1 input.pdf output.pdf
```

Example: Re-sample all images with a resolution of 400 DPI or more to 300 DPI (default):

```
pdf2pdf -ocd 300 -oct 400 input.pdf output.pdf
```

Example: Do not re-sample:

```
pdf2pdf -oct -1 input.pdf output.pdf
```

Compatibility note: Initially, this switch was called -ocD and then renamed to -oct to avoid confusions with the switch -ocd.

7.5.13 -ocx Export recognized OCR text to file

Export recognized OCR text to file -ocx ‹file›

Export the retrieved OCR text to a file. This function can only be used in combination with an OCR engine (see -ocr). When an OCR engine is set, the OCR text is always embedded in the resulting PDF document. If this method is used, it is also extracted to a file.

The output format is a table, where rows are separated by a new line and columns are separated by a tabulator. The table contains the following columns:

Output column	Description
Page	Page number
Image	PDF object number which contains the image
FontSize	Font size in points

`FontName`	Font name, for any barcode font the name is `Barcode`. This value is only set if the font name is returned by the OCR engine.
`FontFamily`	**1** Serif **2** SansSerif **3** Monospaced This value is only set if provided by the OCR engine
`FontStyles`	**2** Bold **4** Italic **8** Underline **16** Strikeout This value is only set if provided by the OCR engine Example: 6 = 2 + 4 = Bold + Italic
`Baseline`	Baseline of the text
`Left, Top, Right, Bottom`	Bounding box of the text in PDF coordinates
`String`	Recognized text

Example: Write extracted text to the file `text.txt`.

```
pdf2pdf -ocr abbyy11 -ocx text.txt input.pdf output.pdf
```

7.6 Return codes

There are different return codes supported. All codes except return code `0` indicate an error or problem.

Error Code	Description
`0`	Success Occurs when: ■ Analysis only and no errors ■ Conversion successful
1	Cannot open input file
2	Cannot create output file

3	Parameter error Examples: ■ Invalid option or parameter ■ Invalid XMP file provided ■ File is not a valid ICC color profile ■ ICC version does not conform ■ Other parameter error
4	Cannot convert input file due to conformance problems Examples: ■ File is corrupt and cannot be repaired ■ Aborted conversion because of an ocr error. ■ An error occurred applying the digital signature. ■ An error occurred during linearization. ■ Unable to convert file to PDF/A because a font that must be embedded is not available in the font directrories. See chapter Fonts for more information on resolving this issue. ■ Input file cannot be converted to meet required conformance level. See option `-ad`.
5	Critical conversion errors occurred during conversion Nonetheless the resulting document conforms to PDF/A. See Conversion errors for more information on conversion errors and how they can be handled.
6	Output file is not conformant (post-analysis) I.e. the output file does not conform to the requested standard. For more information, see Post-analysis.
8	Input file is not conformant (pre-analysis) Occurs when only pre-analysis is active and failed (see option `-ma`).
10	Invalid license No valid license for the 3-Heights® PDF to PDF/A Converter Shell could be found. See License management for more information.

If the input file cannot be read, the following error is returned (return code 1):

```
Cannot open file input.pdf.
```

If the output file cannot be created, an error message is returned (return code 2):

```
Failed to create output file "path\to\out.pdf": The system cannot find the path specified.
```

8 Log file

All steps in the diagram shown in the Process description can write to the log file. There are three types of messages in the log file: `Warnings/Information`, `Errors` and `Reports`.

8.1 Warnings and information

Describe the current process step. They do not inhibit the conversion. Prefix: -

Example:

```
- Opening file input.pdf
- Analyzing input.pdf
- Conformance level has been downgraded to level U.
- Performing post-analysis for output.pdf.
- Post-analysis for output.pdf has been successful.
- File input.pdf converted successfully.
```

8.2 Errors

Inhibit a successful conversion. Prefix: *

Example: The input file cannot be opoened

```
* Cannot open file input.jpg.
```

Example: Distinguish critical from non-critical conversion events

Critical conversion events use the prefix * and non-critical - (see chapter Conversion errors).

```
- Opening file input.pdf.
- Analyzing input.pdf.
- FontSubst: Substitute font 'Arial-BlackItalic' with multiple master font.
- Embed font 'Verdana'.
- Embed font 'Verdana-Bold'.
* Metadata: xmp:Format :: Value removed because it is not defined in the schema description.
(document metadata)
- Create calibrated color space substitute for DeviceGray. (content of page 1)
- Conversion events:
  * Parts of the XMP metadata could not be repaired and had to be removed.
  - Font substituted.
- Performing post-analysis for output.pdf.
- Post-analysis for output.pdf has been successful.
* Conversion errors in output.pdf.
```

8.3 Reports

Reports are only created if the corresponding option (Details or Summary) is selected. In detailed reports, each violation is listed with a page number (page 0 = document level), PDF object number, error code, a description, and a counter of how many times the error occurs. In a summary report, violations that are detected at least once are reported once. Prefix: none.

Example: Details

```
"input.pdf", 0, 53, 0x80410604, "The key Metadata is required but missing.", 1
"input.pdf", 2, 14, 0x00418704, "The font Verdana must be embedded.", 1
"input.pdf", 1, 4, 0x03418614, "A device-specific color space (DeviceGray) without an
appropriate output intent is used.", 1
```

Example: Summary

```
The document contains fonts without embedded font programs or encoding
information (CMAPs).
The document's metadata is either missing or inconsistent or corrupt.
The document doesn't provide appropriate logical structure information.
```

9 Version history

9.1 Changes in versions 6.19–6.27

- **Update** license agreement to version 2.9

9.2 Changes in versions 6.13–6.18

- **New** option `-nd`.

9.3 Changes in versions 6.1–6.12

- Digital Signatures
  - Swisscom All-in Signing Service
    - **New** support for accounts (`Identity`) based on Swisscom CA 4 Certificate Authorities.
    - **New** support to create PAdES signatures (format `ETSI.CAdES.detached`).
  - **Improved** embedding of revocation information (OCSP, CRL, and trust chain) to always use the document security store (DSS)[13].
  - **Changed** the creation of signatures of format `ETSI.CAdES.detached` to include revocation information if `-co` is not used and if supported by the cryptographic provider.
  - **Improved** support for new version of the GlobalSign Digital Signing Service. The service endpoint should be updated to `https://emea.api.dss.globalsign.com:8443/v2`.
- **Improved** repair of corrupt DCT stream data.
- **New** support for ZUGFeRD 2.1.1 electronic invoices (including XRechnung)
- **Improved** creation of annotation appearances.
- **New** support use of fonts that are installed only for the current Windows user.
- **Improved** search algorithm for installed fonts: User fonts under Windows are now also taken into account.
- **Changed** behavior of option `-az`: ZUGFeRD 2.1 is now used instead of the equivalent Factur-X 1.0. ZUGFeRD 2.1 is always preferred over 2.0 and thus profile "EN 16931" is now also supported and not considered ambiguous anymore.
- **Changed** behavior of option `-ai`: ZUGFeRD 2.1 is now preferred over 2.0 if `zf` was selected.

9.4 Changes in version 5

- Digital Signatures
  - **New** support to get CRLs using HTTPS and via HTTP redirection.
- **New** support for ZUGFeRD 2.0 hybrid electronic invoices.
- **Improved** conversion of transparent objects to PDF/A-1.
- **Improved** log output for conversion events. Now all conversion events are written to the log file.
- **New** additional supported operating system: Windows Server 2019.
- **New** option `-ai` to add XML invoice files with additional options.
- **Deprecated** option `-az` (replaced by `-ai`).

[13] Use the option `-nd` to restore the previous behavior.

9.5 Changes in version 4.12

- **Introduced** license feature `Signature`.
- Digital Signatures
  - **New** support to sign OCSP requests, if required by the OCSP service.
  - **New** support for OCSP requests over HTTPS.
  - **Changed** acceptance criteria for OCSP responses that specify no validity interval (missing nextUpdate field, which is uncommon). Previously a validity interval of 24 hours has been used, now 5 minutes due to Adobe® Acrobat® compatibility.
- **New** support for Factur-X hybrid electronic invoices.
- **New** OCR plugin `"abbyy12"` for the ABBYY FineReader 12 engine.
- **Changed** behavior when adding additional font directories: The default directories are now always considered.
- **Improved** detection of corrupt embedded fonts, DCT streams and CMap encodings.
- **New** HTTP proxy setting in the GUI license manager.
- **Changed** option `-cef` to only process files whose MIME Type is PDF.

9.6 Changes in version 4.11

- **New** font substitution for CJK (Chinese, Japanese, Korean) fonts if an exact match is missing.
- **Improved** conversion of files with optional content (layers) to minimize visual differences while also preserving all content.
- **New** support for the creation of appearance streams for free text annotations that contain rich text content.
- Digital Signatures
  - **New** ability to sign documents that are larger than 2GB (64-bit version only).
- **New** support for reading PDF 2.0 documents.
- **New** support for the creation of output files larger than 10GB (not PDF/A-1).
- **Improved** search in installed font collection to also find fonts by other names than TrueType or PostScript names.
- **Improved** font subsetting of CFF and OpenType fonts.
- **Improved** repair of corrupt image streams.
- **New** treatment of the `DocumentID`. In contrast to the `InstanceID` the `DocumentID` of the output document is inherited from the input document.
- **New** option `-cef` to activate conversion of embedded PDF documents (PDF/A-3 only).

9.7 Changes in version 4.10

- **Improved** conversion of transparent objects to PDF/A-1. E.g. filled paths that are transparent are converted to outlines in order to not cover underlying content when the transparency attribute is removed.
- **Improved** conversion of numbers that are larger than the implementation limit of PDF/A-1.
- **Improved** conversion of logical structure information (PDF/A level A).
- **Changed** behavior: Lock OCGs (layers) that need to be added to user interface (PDF/A-2 and PDF/A-3).
- Digital signatures
  - **New** support for the new European PAdES norm (ETSI EN 319 142). See chapter "How to Create a PAdES Signature" in the user manual for more information.
  - **New** support for the GlobalSign Digital Signing Service as cryptographic provider to create signatures and timestamps.
  - **New** signature algorithm RSA with SSA-PSS (PKCS#1v2.1) can be chosen by setting the provider session property `SigAlgo`.
- **New** conversion event `65536` when logical structure information is removed.

- **New** support for writing PDF objects into object streams. Most objects that are contained in object streams in the input document are now also stored in object streams in the output document. When enabling linearization, however, no objects are stored in object streams.
- **Improved** robustness against corrupt input PDF documents.
- **Improved** annotation appearance generation for polyline, squiggly, and stamp annotations.
- **Removed** the font `ZapfDingbats.ttf` from the product kit as it is not required anymore.
- **Changed** option `-uf`: The file to update the ToUnicode font information now supports mappings from a character code to a sequence of Unicodes.
- **New** option `-st` to set signature format, e.g. for new European PAdES norm.
- **New** options `-afs1` and `-afs2` to set font size of the signature appearance.

9.8 Changes in version 4.9

- **New** conversion features, e.g. improved conversion of TrueType font programs, ICC color profiles, or creation of annotation appearances.
- **New** supports for bold font simulation if only non-bold font is available in installed font directories.
- **Changed** behavior: The pre-analysis is now more strict, especially in certain corner cases of the PDF/A ISO Standard for which a conversion is strongly advised.
- **Improved** conversion of .notdef character for PDF/A-2 and PDF/A-3.
- **New** generated conversion event 4:
  - When text in input file is ambiguous.
  - When XFA (XML Forms Architecture) form data is removed.
  - When visual appearance of annotation cannot be created.
- **Improved** support for and robustness against corrupt input PDF documents.
- **Improved** repair of embedded font programs that are corrupt.
- **New** support for OpenType font collections in installed font collection.
- **Improved** metadata generation for standard PDF properties.

9.9 Changes in version 4.8

- **New** conversion features, e.g. improved conversion of corrupt data such as fonts, text, or form XObjects.
- **New** conversion event 32768 when interactive elements such as actions or annotations are removed.
- **New** bold font simulation used when substituting bold with non-bold font.
- **Improved** creation of annotation appearances to use less memory and processing time.
- **Added** repair functionality for TrueType font programs whose glyphs are not ordered correctly.
- **Improved** error messages when using the verbose option `-v`.

10 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