

User Manual



# 3-Heights® PDF Validator Shell

Version 6.18.1



# Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
1.1	Description	4
1.2	Functions	4
1.2.1	Features	5
1.2.2	Formats	5
1.2.3	Conformance	5
1.3	Operating Systems	5
<b>2</b>	<b>Installation</b>	<b>6</b>
2.1	Windows	6
2.1.1	How to set the Environment Variable "Path"	6
2.2	Linux and macOS	7
2.2.1	Linux	7
2.3	Uninstall	8
<b>3</b>	<b>License Management</b>	<b>9</b>
3.1	License Features	9
<b>4</b>	<b>Getting Started</b>	<b>10</b>
4.1	Usage	10
4.2	Validate a Document	10
4.2.1	Validate a Single Document	10
4.2.2	Validate all Documents in a Directory	11
4.2.3	Validate without Report	11
4.3	What is PDF/A?	12
4.3.1	PDF/A-1	12
4.3.2	PDF/A-2	12
4.3.3	PDF/A-3	12
4.4	Custom Validation Profiles	13
4.4.1	[File] INI-File Section	13
	FileSize1	13
	FileSize2	13
	MaxPdfVersion	14
	MinPdfVersion	14
	Encryption	14
	Linearization	15
	NonFilters, NonFilter<i>(Non-Approved Filters)</i>	15
4.4.2	[Document] INI-File Section	16
	NonCreators, NonCreator<i>(Non-Approved PDF Creators)</i>	16
	NonProducers, NonProducer<i>(Non-Approved PDF Producers)</i>	16
	EmbeddedFiles, EmbeddedFile<i>(Allowed Embedded File Types)</i>	16
	ProhibitEmbeddedFiles	17
4.4.3	[Pages] INI-File Section	17
	PageSizes, PageSize<i>(Approved Page Sizes)</i>	17
	SizeTolerance (Tolerance for Page Size Comparison)	18
	EmptyPage	18
	MaxPageSize	18
	RequirePageResources	19
4.4.4	[Graphics] INI-File Section	19

	ImageMaxDPI (Maximum Resolution of Images) .....	19
	ImageMinDPI (Minimum Resolution of Images) .....	20
	ScanMaxDPI (Maximum Resolution of Scanned Images) .....	20
	ScanMinDPI (Minimum Resolution of Scanned Images) .....	20
	ScanColor (Color for Scanned Images) .....	21
	OCRText .....	21
	ProhibitColor .....	21
	ProhibitTransparency .....	22
	Layers .....	22
	HiddenLayers .....	22
4.4.5	[ Fonts ] INI-File Section .....	23
	Fonts, Font<i> (Approved Fonts) .....	23
	NonFonts, NonFont<i> (Non-Approved Fonts) .....	23
	Subsetting .....	24
	NonStdEmbedded .....	24
	Embedding, EmbeddingExcFonts, EmbeddingExcFont<i> (Embedding of Fonts) ...	25
4.4.6	[ Interactive Features ] INI-File Section .....	25
	Annotations, Annotation<i> (Approved Annotations) .....	25
	NonActions, NonAction<i> (Non-Approved Actions) .....	25
4.4.7	[ Digital Signatures ] INI-File Section .....	26
	Provider .....	26
	ValidateNewest (Validate Newest Signature) .....	27
	Criteria, Criterion<i> (Signature Validation Criteria) .....	27
<b>5</b>	<b>Interface Reference .....</b>	<b>29</b>
5.1	Switches .....	29
5.1.1	-c1 Set the Conformance Level .....	29
5.1.2	-e Stop on Error .....	29
5.1.3	-pw Read an Encrypted PDF File .....	30
5.1.4	-rd Report Conformance Violations in Detail .....	30
5.1.5	-r1 Reporting Level .....	31
5.1.6	-rs Report Conformance Violations Summary .....	31
5.1.7	-cc1 Claimed Conformance and Level .....	32
5.1.8	-p Set custom validation profile .....	32
5.1.9	-lk Set License Key .....	32
5.1.10	-v Verbose Mode .....	33
5.2	Return Codes .....	33
<b>6</b>	<b>Coverage .....</b>	<b>34</b>
6.1	All PDF Versions .....	34
6.1.1	Lexical Checks .....	34
6.1.2	Syntactic Checks .....	34
6.1.3	Semantic Checks .....	34
6.2	Checks Specific for PDF/A .....	34
6.2.1	Lexical Checks .....	34
6.2.2	Semantic Checks .....	35
6.3	Supported PDF Versions .....	35

**7      Version History ..... 37**

7.1    Patches in Version 6.18 ..... 37

7.2    Changes in Version 6 ..... 37

7.3    Changes in Version 5 ..... 37

7.4    Changes in Version 4.12 ..... 37

7.5    Changes in Version 4.11 ..... 37

7.6    Changes in Version 4.10 ..... 37

7.7    Changes in Version 4.9 ..... 38

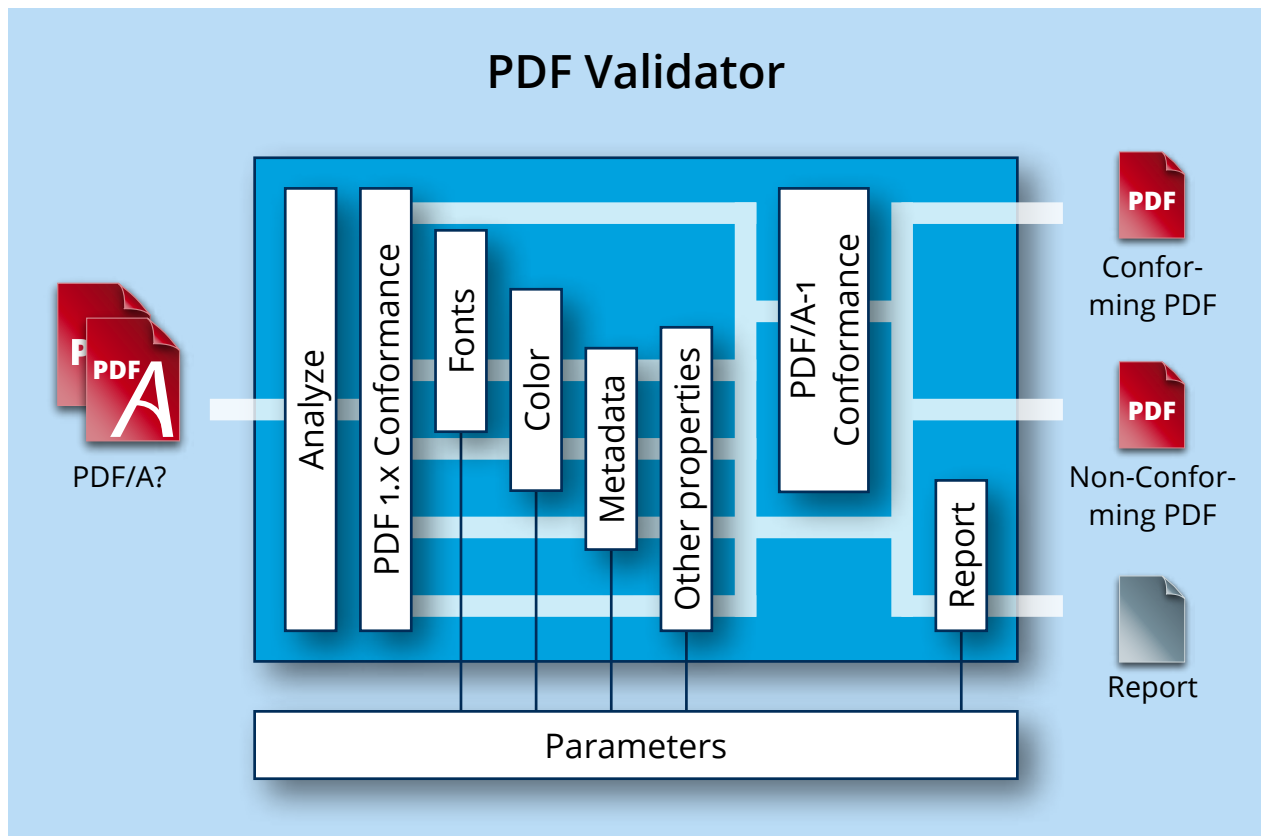
7.8    Changes in Version 4.8 ..... 38

**8      Licensing, Copyright, and Contact ..... 39**

# 1 Introduction

## 1.1 Description

The 3-Heights® PDF Validator Shell safeguards the quality of PDF documents. It checks PDF files for conformance to the ISO standards for PDF and PDF/A documents. Unfortunately, there are many PDF creation or manipulation tools in use that do not comply with the PDF or PDF/A standard. System and operational interruptions often occur as a result. Incoming documents should be verified before they flow into business processes to prevent interruptions of this nature and to avoid unexpected costs.



The 3-Heights® PDF Validator Shell checks whether PDF documents comply with the PDF or PDF/A standard. Additional verification tests, such as checking the version number of the PDF document, are also possible; the tool can also verify conformance to internal directives - use of the right color, for instance, or use of the right fonts and other specifications.

The 3-Heights® PDF Validator Shell is a command line tool. It is meant to be used in automated processes to validate high volumes of PDF files. It is a high performance tool made for developers and used in scripts; it does not provide any graphical user interface.

## 1.2 Functions

3-Heights® PDF Validator Shell verifies PDF documents in accordance with the ISO standard for PDF and also PDF/A for long-term archiving. The tool can check the conformity of individual documents and entire archives. The result output is needs-oriented, e.g. a detailed report for a manufacturer of PDF software or a summary of error reports for the user. The description includes every detail such as frequency, page number or PDF object number. Verification of internal specifications (e.g. standard image resolution) can occur at the same time.

## 1.2.1 Features

- Validate PDF documents on the basis of various PDF specifications (PDF 1.x, PDF 2.0, PDF/A-1, PDF/A-2, PDF/A-3)
- PDF-conforming dependent lexical, syntactic, and semantic checks (see [Coverage](#))
- Detailed or summarized reporting (log file)
- Detailed error description (number, type, description, PDF object, page number)
- Classification by error, warning and information
- Optional cancellation of validation on occurrence of the first error
- Read encrypted PDF files
- Determine claimed conformance of document
- Validate conformance to corporate directives defined in custom profile

## 1.2.2 Formats

Input Formats:

- PDF 1.x (PDF 1.3, ..., PDF 1.7)
- PDF 2.0
- PDF/A-1a, PDF/A-1b
- PDF/A-2a, PDF/A-2b, PDF/A-2u
- PDF/A-3a, PDF/A-3b, PDF/A-3u

## 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)
- Quality assurance: veraPDF test corpus and Isartor test suite

## 1.3 Operating Systems

The 3-Heights® PDF Validator Shell is available for the following operating systems:

- Windows Client 7+ | x86 and x64
- Windows Server 2008, 2008 R2, 2012, 2012 R2, 2016, 2019 | 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.

## 2 Installation

### 2.1 Windows

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

The installation of the software requires the following steps.

1. You need administrator rights to install this software.
2. Log in to your download account at <http://www.pdf-tools.com>. Select the product “PDF Validator Shell”. If you have no active downloads available or cannot log in, please contact [pdfsales@pdf-tools.com](mailto:pdfsales@pdf-tools.com) for assistance.

You will find different versions of the product available. We suggest to download the version, which is selected by default. A different version can be selected using the combo box.

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

This creates the following subdirectories:

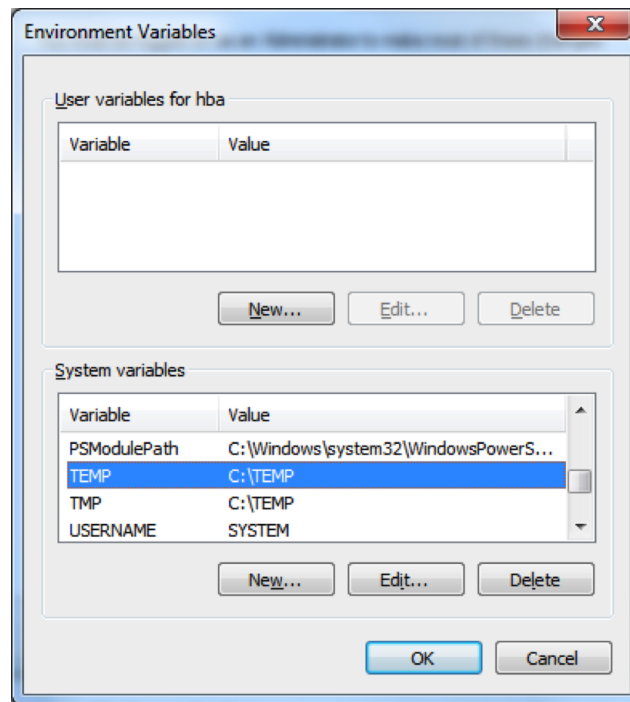
Subdirectory	Description
bin	Contains the runtime executable binaries.
doc	Contains documentation.

5. (Optional) To easily use the 3-Heights® PDF Validator 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](#).

#### 2.1.1 How to set the Environment Variable “Path”

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

Select “Path” and “Edit”, then add the directory where `pdfvalidator.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 Validator Shell:

### File Description

Name	Description
bin/x64/pdfvalidator	This is the main executable.
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 pdfvalidator
```

In case the above reports any missing libraries you have three 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 `libc++6`.
- c. Use GNU shared libraries provided by PDF Tools AG:
  1. Go to <http://www.pdf-tools.com> and navigate to "Support" → "Utilities".
  2. Download the GNU shared libraries for your platform.



3. Install the libraries manually according your system's documentation. This typically involves copying them to your library directory, e.g. `/usr/lib` or `/usr/lib64`, and running `ldconfig`.
4. Verify that the GNU shared libraries required by the product are available on your system now.
3. Create a link to the executable from one of the standard executable directories, e.g:

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

4. Optionally register your license key using the [license manager](#).

## 2.3 Uninstall

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

If you have used the ZIP file for the installation: In order to uninstall the product, undo all the steps done during installation.

## 3 License Management

The 3-Heights® PDF Validator 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 will be 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 Validator Shell contains one area to which the following license feature is assigned:

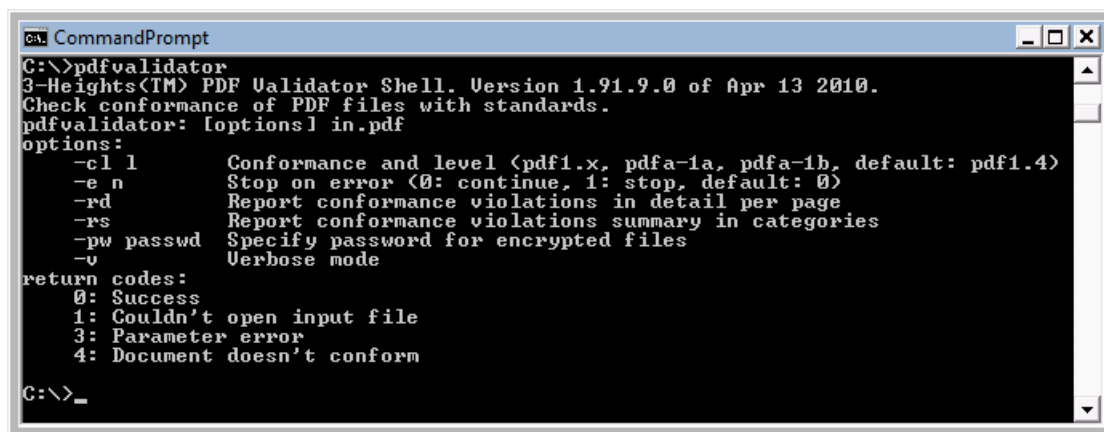
**Custom** Verify conformance to custom corporate directives.

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 Getting Started

### 4.1 Usage

By typing `pdfvalidator` without parameters, the usage, the version and a list of available options is returned.



```
Command Prompt
C:\>pdfvalidator
3-Heights(TM) PDF Validator Shell. Version 1.91.9.0 of Apr 13 2010.
Check conformance of PDF files with standards.
pdfvalidator: [options] in.pdf
options:
  -cl l      Conformance and level (pdf1.x, pdfa-1a, pdfa-1b, default: pdf1.4)
  -e n      Stop on error (0: continue, 1: stop, default: 0)
  -rd       Report conformance violations in detail per page
  -rs       Report conformance violations summary in categories
  -pw passwd Specify password for encrypted files
  -v        Verbose mode
return codes:
  0: Success
  1: Couldn't open input file
  3: Parameter error
  4: Document doesn't conform
C:\>_
```

### 4.2 Validate a Document

#### 4.2.1 Validate a Single Document

In order to validate a document and retrieve a report, two parameters are required, further parameters are optional.

The required parameters are:

- PDF file to validate

Optional Parameters are:

- Conformance level ([-cl](#))
- Reporting type ([-rs](#) or [-rd](#))
- Stop on error ([-e](#))
- Verbose Mode ([-v](#))

**Example:** Set the reporting type to "report summary" ([-rs](#)), set the conformance level to PDF/A-1b ([-cl pdfa-1b](#)), validate the PDF file `input.pdf`.

```
pdfvalidator -rs -cl pdfa-1b input.pdf
```

The result is written to standard out. No output means either no violations against the selected specification or no reporting type was set.

## 4.2.2 Validate all Documents in a Directory

Wildcards (\*) are supported by the tool.

**Example:** Validate all PDF files in the current directory against PDF/A-1b. Do not report any violations.

```
pdfvalidator -cl pdfa-1b *.pdf
```

Reporting messages is enabled using either of the switches [-rd](#) (report details) or [-rs](#) (report summary). If you are only interested in a general message (e.g. font not embedded), it is best to go by the summary. If you are a developer and like additional information what is interfering with the standard, use the option [-rd](#) to list a detailed report.

**Example:** Validate all PDF files in the current directory against PDF/A-1b. Report details ([-rd](#)). The switch [-v](#) lists the currently validated document.

```
pdfvalidator -cl -v pdfa-1b -rd *.pdf
Validating file aaa.pdf.
"aaa.pdf", 0, 10, 0x80410604, "The key Metadata is required but missing.", 1
"aaa.pdf", 1, 83, 0x00418704, "The font Helvetica-Bold must be embedded.", 1
"aaa.pdf", 1, 15, 0x00418608, "The dictionary must not contain the key 'D'.", 2
"aaa.pdf", 5, 0, 0x83410612, "The document does not conform to the requested standard.", 1
The document does not conform to the PDF/A-1b standard.
Validating file bbb.pdf
...
```

## 4.2.3 Validate without Report

If you are not interested in messages at all, and simply want a yes/no answer to the conformance test, then look at the return code. Any return code other than 0 indicates a problem.

**Example:** The following batch script (written for Windows) validates all PDF files in a directory and outputs whether the PDF file conforms to PDF/A-1b or not:

```
@ECHO OFF
FOR %%i in (*.pdf) DO (
    SET name=%%i
    CALL :_validate )
GOTO :EOF
:_validate
pdfvalidator -cl pdfa-1b -e 1 "%name%"
IF %ERRORLEVEL%==0 (
    @ECHO %name% : OK
) ELSE (
    @ECHO %name% : ** NOT conforming **
)
GOTO :EOF
```

If you want to use the batch file above, copy it into a text file and name it for example `validate.bat`. A possible output could look like this:

**Example:** Running the batch file `validate.bat` and its possible output:

```
C:\> validate
001.pdf : OK
002.pdf : OK
Aaa.pdf : ** NOT conforming **
Couldnt open PDF file Bbb.pdf.
Bbb.pdf : ** NOT conforming **
Ccc.pdf : OK
...
```

## 4.3 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 please visit: <http://www.pdf-tools.com/pdf20/en/resources/pdf-iso-standards/>.

### 4.3.1 PDF/A-1

PDF/A-1 (ISO 19005-1) is based on PDF 1.4 (Acrobat 5). On top of PDF 1.4, it has additional requirements to keep the document self-contained and suitable for long-term archival. The most important are:

- Encryption may not be used
- If device-dependent color space (e.g. DeviceRGB, DeviceCMYK, DeviceGray) are used, a corresponding color profile must be embedded
- Fonts used for visible text must be embedded
- Transparency may not be used

### 4.3.2 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 shall complement the first part and 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.

### 4.3.3 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.

## 4.4 Custom Validation Profiles

In addition to checking documents' conformance to the PDF Reference and PDF ISO standards, the 3-Heights® PDF Validator Shell can ensure conformance to custom corporate directives. Custom checks are defined in a configuration file and activated using the option `-p`.

The format of the configuration file follows the INI file syntax. By default, all custom checks are deactivated, so all custom checks must be enabled explicitly. All lines starting with a semicolon ";" are ignored.

### 4.4.1 [File] INI-File Section

#### FileSize1

**Key:** `FileSize1`  
Error code: `CHK_E_FILESIZE1`

Define the maximum allowed file size in megabytes.

**Example:** Set allowed file size to 100 MB.

```
[File]
FileSize1=100
```

#### FileSize2

**Key:** `FileSize2`  
Error code: `CHK_E_FILESIZE2`

Define a second limit for the maximum allowed file size in megabytes. If `FileSize2` is specified, it must be larger than the value of `FileSize1`. If a file's size is larger than `FileSize2`, the error `CHK_E_FILESIZE2` is raised, else if the size is larger than `FileSize1`, `CHK_E_FILESIZE1` is raised.

**Example:** Set allowed file size to 200 MB.

```
[File]
FileSize2=200
```

## MaxPdfVersion

**Key:** MaxPdfVersion  
Error code: CHK\_E\_MAXPDFVERS

The highest PDF version a document may have is defined by the setting **MaxPdfVersion**. The argument is a period-separated value with a major version, a minor version and an optional extension level.

**Example:** Set maximum allowed PDF version to PDF 1.4 (Acrobat 5).

```
[File]  
MaxPdfVersion=1.4
```

**Example:** Set the maximum allowed PDF version to PDF 1.7, extension level 3 (Acrobat 9).

```
[File]  
MaxPdfVersion=1.7.3
```

## MinPdfVersion

**Key:** MinPdfVersion  
Error code: CHK\_E\_MINPDFVERS

The setting **MinPdfVersion** sets the minimum PDF version the document must have. The usage is equivalent to **MaxPdfVersion**.

**Example:** The following setting requires the document under test to be at least PDF 1.3 and no higher than PDF 1.6.

```
[File]  
MinPdfVersion=1.3  
MaxPdfVersion=1.6
```

## Encryption

**Key:** Encryption  
Error code: CHK\_E\_ENCRYPTION

Check whether or not the file is encrypted.

**true** Raise error if file is not encrypted.

**false** Raise error if file is encrypted.

**Example:** Dis-allow encrypted files.

```
[File]
Encryption=false
```

## Linearization

**Key:** **Linearization**  
Error code: **CHK\_E\_LINEARIZATIION**

Check whether or not the file is linearized.

**true** Raise error if file is not linearized.

**false** Raise error if file is linearized.

**Example:** Dis-allow linearized files.

```
[File]
Linearization=false
```

## NonFilters, NonFilter<i> (Non-Approved Filters)

**Key:** **NonFilters**  
**Key:** **NonFilter<i>**  
Error code: **CHK\_E\_FILTER**

Non-approved stream filters are defined by setting **NonFilters=<n>**, where **<n>** is the count of non-approved stream filters, i.e. a value larger than 0. The names of the filters are defined using **NonFilter<i>=<Name i>** where **<i>** is an index ranging from **1** to **<n>**. Possible values for **<Name i>** are the PDF filters:

- |                   |                  |
|-------------------|------------------|
| ■ ASCHexDecode    | ■ CCITTFaxDecode |
| ■ ASCII85Decode   | ■ JBIG2Decode    |
| ■ LZWDecode       | ■ DCTDecode      |
| ■ FlateDecode     | ■ JPXDecode      |
| ■ RunLengthDecode |                  |

**Example:** Disallow JPEG2000 compressed images:

```
[File]
NonFilters=1
NonFilter1=JPXDecode
```



## 4.4.2 [Document] INI-File Section

### NonCreators, NonCreator<i> (Non-Approved PDF Creators)

**Key:** NonCreators

**Key:** NonCreator<i>

Error code: CHK\_E\_CREATOR

Non-approved PDF creators are defined by setting **NonCreators**=<n>, where <n> is the count of non-approved creators, i.e. a value larger than 0. The names of the creators are defined using **NonCreator**<i>=<Name i>, where <i> is an index ranging from 1 to <n> and <Name i> is the name of the non-approved PDF creator.

**Example:** A list of non-approved PDF creators can be defined like this:

```
[Document]
NonCreators=2
NonCreator1=pdf fools
NonCreator2=badpdfcreator
```

### NonProducers, NonProducer<i> (Non-Approved PDF Producers)

**Key:** NonProducers

**Key:** NonProducerX

Error code: CHK\_E\_PRODUCER

Non-approved PDF producers are defined similar to non-approved PDF creators.

**Example:** A list of non-approved PDF producers can be defined like this:

```
[Document]
NonProducers=1
NonProducer1=pdf fools
```

### EmbeddedFiles, EmbeddedFile<i> (Allowed Embedded File Types)

**Key:** EmbeddedFiles

**Key:** EmbeddedFileX

Error code: CHK\_E\_EFTYPE

List of allowed embedded file types. Wild cards are supported at the beginning or the end of the string.

**Example:** Allow embedded PDF files and job options only.

```
[Document]
EmbeddedFiles=2
```

```
EmbeddedFile1=*.pdf
EmbeddedFile2=*.joboptions
```

## ProhibitEmbeddedFiles

**Key:** `ProhibitEmbeddedFiles`  
**Error code:** `CHK_E_EF`

Use the option `ProhibitEmbeddedFiles` to check for embedded files.

**true** Raise error if document contains embedded files.

**false** Do not check for embedded files.

**Example:** Disallow embedded files.

```
[Document]
ProhibitEmbeddedFiles=true
```

## 4.4.3 [Pages] INI-File Section

### PageSizes, PageSize<i> (Approved Page Sizes)

**Key:** `PageSizes`  
**Key:** `PageSize<i>`  
**Error code:** `CHK_E_PAGESIZE`

Approved page sizes are specified by setting `PageSizes=<n>`, where `<n>` is the count of page sizes, i.e. a value larger than 0. Sizes are defined using `PageSize<i>=<Size i>`, where `<i>` is an index ranging from 1 to `<n>` and `<Size i>` is one of the following size specifications:

**Letter** US Letter page 8.5 x 11 in.

**A<k>** A series international paper size standard **A0** to **A10**.

**DL** DIN long paper size 99 x 210 mm.

**<w> x <h> <uu>** Arbitrary page size of width `<w>`, height `<h>` measured in units `<uu>`. Supported units are `in`, `pt`, `cm`, and `mm`.

The tolerance used for size comparison is 3 points (3/72 inch, approximately 1 mm), unless the key [SizeTolerance \(Tolerance for Page Size Comparison\)](#) is specified.

**Example:**

```
[Pages]
PageSizes=4
```

```
PageSize1=A0
PageSize2=A3
PageSize3=15.53 x 15.35 in
PageSize4=181 x 181 mm
```

## SizeTolerance (Tolerance for Page Size Comparison)

**Key:** SizeTolerance

Tolerance used for page size comparison.

**Percentage** Proportional difference, e.g. SizeTolerance=10%.

**Absolute Value** Absolute difference in points (1/72 inch), e.g. SizeTolerance=72 allows 1 inch.

The tolerance used for size comparison is 3 points (3/72 inch), unless the key SizeTolerance is specified.

**Example:** Allow a tolerance of 10%.

```
[ Pages ]
SizeTolerance=10%
```

## EmptyPage

**Key:** EmptyPage  
Error code: CHK\_E\_EMPTYPAGE

Use the key EmptyPage to disallow empty pages. A page is considered empty, if no graphic objects are drawn onto it.

**true** Raise error if page is not empty.

**false** Raise error if page is empty.

**Example:** Raise error CHK\_E\_EMPTYPAGE, if document contains an empty page.

```
[ Pages ]
EmptyPage=false
```

## MaxPageSize

**Key:** MaxPageSize  
Error code: CHK\_E\_MAXPAGESIZE

Use the key **MaxPageSize** to disallow pages exceeding the specified size in any dimension. The tolerance for size comparison is specified by the key **SizeTolerance**. Both portrait and landscape variants of **MaxPageSize** are allowed.

See description of **PageSize<i>** for a description of supported page size formats.

**Example:** Raise error **CHK\_E\_MAXPAGESIZE**, if document contains a page larger than A4.

```
[Pages]
MaxPageSize=A4
```

## RequirePageResources

**Key:** **RequirePageResources**  
Error code: **CHK\_E\_PAGERESOURCES**

Test if pages contain an explicitly associated resource dictionary.

**true** Raise error if page does not have resource dictionary.

Note that it is allowed for a page to not have an explicitly associated resource dictionary, if it is inherited from the pages tree. The 3-Heights® PDF Validator Shell always validates that all pages have a resource dictionary.

**Example:** Raise error **CHK\_E\_PAGERESOURCES**, if document contains a page without a resource dictionary.

```
[Pages]
RequirePageResources=false
```

## 4.4.4 [Graphics] INI-File Section

### ImageMaxDPI (Maximum Resolution of Images)

**Key:** **ImageMaxDPI**  
Error code: **CHK\_E\_IMGMAXDPI**

Use **ImageMaxDPI** to set maximum allowed resolution in DPI (dots per inch) for all images.

**Example:** Set the maximum allowed resolution to 602 DPI.

```
[Graphics]
ImageMaxDPI=602
```

## ImageMinDPI (Minimum Resolution of Images)

**Key:** ImageMinDPI  
Error code: CHK\_E\_IMGMINDPI

Use **ImageMinDPI** to set minimum allowed resolution in DPI (dots per inch) for all images.

**Example:** Embedded images must have a resolution from 148 to 152 DPI.

```
[Graphics]  
ImageMinDPI=148  
ImageMaxDPI=152
```

## ScanMaxDPI (Maximum Resolution of Scanned Images)

**Key:** ScanMaxDPI  
Error code: CHK\_E\_SCANMAXDPI

Use **ScanMaxDPI** to set maximum allowed resolution in DPI (dots per inch) for scanned images. All images that cover a majority of the page are classified as scanned images.

**Example:** Set the maximum allowed resolution to 602 DPI.

```
[Graphics]  
ScanMaxDPI=602
```

## ScanMinDPI (Minimum Resolution of Scanned Images)

**Key:** ScanMinDPI  
Error code: CHK\_E\_SCANMINDPI

Use **ScanMinDPI** to set minimum allowed resolution in DPI (dots per inch) for scanned images.

**Example:** Embedded images must have a resolution from 148 to 152 DPI.

```
[Graphics]  
ScanMinDPI=148  
ScanMaxDPI=152
```

## ScanColor (Color for Scanned Images)

**Key:** ScanColor

Error code: CHK\_E\_SCANCLR

If you do not want to allow color scans, use the option ScanColor.

**true** Raise error if scanned image does not contain color.

**false** Raise error if scanned image does contain color.

**Example:** If you want to dis-allow color scans.

```
[Graphics]  
ScanColor=false
```

## OCRText

**Key:** OCRText

Error code: CHK\_E\_OCRTTEXT

Test, if scanned images have OCR text, i.e. if the file is word searchable.

**true** Raise error if scanned image has no OCR text (i.e. file is not word searchable).

**false** Raise error if scanned image has OCR text (i.e. file is word searchable).

**Example:** Raise an error, if an image has no OCR text.

```
[Graphics]  
OCRText=true
```

## ProhibitColor

**Key:** ProhibitColor

Error code: CHK\_E\_CLRUSED

If you only want to allow black and white, use the option ProhibitColor.

**true** Raise error if page contains color.

**false** Do not check for color.

**Example:**

```
[Graphics]
```

```
ProhibitColor=true
```

## ProhibitTransparency

**Key:** ProhibitTransparency  
Error code: CHK\_E\_TRANSPARENCYUSED

Activate the option ProhibitTransparency in order to check for transparency. Method for determining transparency on a page is described in detail in Annex A of the ISO 19005-2:2011 standard (PDF/A-2).

**true** Raise error if page contains transparency.

**false** Do not check for transparency.

### Example:

```
[Graphics]  
ProhibitTransparency=true
```

## Layers

**Key:** Layers  
Error code: CHK\_E\_LAYERS

Use the key Layers to disallow layers.

**true** Raise error if document contains no layers.

**false** Raise error if document contains layers.

**Example:** Raise error CHK\_E\_LAYERS, if document contains layers.

```
[Graphics]  
Layers=false
```

## HiddenLayers

**Key:** HiddenLayers  
Error code: CHK\_E\_HIDDENLAYERS

Use the key HiddenLayers to disallow hidden layers.

**true** Raise error if document contains no hidden layers.

**false** Raise error if document contains hidden layers.

**Example:** Raise error CHK\_E\_HIDDENLAYERS, if document contains hidden layers.

```
[Graphics]
HiddenLayers=false
```

## 4.4.5 [ Fonts ] INI-File Section

There are two ways of restricting the allowed fonts used in the validated document. Either every font that is approved is explicitly white-listed or every font that is not approved is black-listed. Most appropriately only one of the two settings is used at once.

### Fonts, Font<i> (Approved Fonts)

**Key:** Fonts  
**Key:** Font<i>  
Error code: CHK\_E\_FONT

Restrict the approved fonts to a defined set of fonts. The number of approved fonts is set by **Fonts=<n>**, where <n> is a number larger than 0. The names of the approved fonts are listed using **Font<i>=<fontname i>**, where <i> is an index ranging from 1 to <n> and <fontname i> is a font name. Wild cards are supported. Font styles are defined by adding a command and the style after the font family name.

**Example:** A list of approved fonts can be defined like this:

```
[Fonts]
Fonts=163
Font1=AdvC39b
Font2=AdvC39b
Font3=AdvHC39b
Font4=AdvHC39b
Font5=Arial
Font6=Arial,Bold
...
Font163=ZapfDingbats
```

### NonFonts, NonFont<i> (Non-Approved Fonts)

**Key:** NonFonts  
**Key:** NonFont<i>  
Error code: CHK\_E\_FONT



A list of non-approved fonts can be defined, wild cards are supported.

**Example:**

```
[ Fonts ]  
NonFonts=4  
NonFont1=MSTT*  
NonFont2=T1*  
NonFont3=T2*  
NonFont4=T3*
```

## Subsetting

**Key:** Subsetting  
Error code: CHK\_E\_FNTSUB

Subsetting a font means only those glyphs are embedded in the font program, which are actually used. Subsetting is mainly used to keep the file size small. The setting Subsetting can be used to test the subsetting of embedded fonts.

**true** Raise error if embedded font is not subset.

**false** Raise error if embedded font is subset.

**Example:** Require all fonts to be subset.

```
[ Fonts ]  
Subsetting=true
```

## NonStdEmbedded

**Key:** NonStdEmbedded  
Error code: CHK\_E\_FNTEMB

The setting NonStdEmbedded can be used to test the embedding of non-standard fonts.

**true** Raise error if non-standard font is not embedded.

**false** Raise error if non-standard font is embedded.

**Example:** Require all non-standard fonts to be embedded.

```
[ Fonts ]  
NonStdEmbedded=true
```

## Embedding, EmbeddingExcFonts, EmbeddingExcFont<i> (Embedding of Fonts)

**Key:** Embedding  
**Key:** EmbeddingExcFonts  
**Key:** EmbeddingExcFont<i>  
Error code: CHK\_E\_FNTEMB

The setting **Embedding** can be used to test the embedding of fonts that are used for rendering. The keys **EmbeddingExcFonts** and **EmbeddingExcFont<i>** define a list of fonts exempt from the test.

**true** Raise error if a font is neither embedded nor in the list of exceptions.

**false** Raise error if a font is embedded and not in the list of exceptions.

Note that this test works independently of **NonStdEmbedded**.

**Example:** Require all fonts except “Albertus” and “Courier” to be embedded.

```
[Fonts]
Embedding=true
EmbeddingExcFonts=2
EmbeddingExcFont1=Albertus*
EmbeddingExcFont2=Courier*
```

## 4.4.6 [Interactive Features] INI-File Section

### Annotations, Annotation<i> (Approved Annotations)

**Key:** Annotations  
**Key:** Annotation<i>  
Error code: CHK\_E\_ANNOTATION

Set a list of approved annotations.

**Example:** Allow form fields (Widget annotations) and links (Link annotations) only.

```
[Interactive Features]
Annotations=2
Annotation1=Widget
Annotation2=Link
```

### NonActions, NonAction<i> (Non-Approved Actions)

**Key:** NonActions  
**Key:** NonAction<i>  
Error code: CHK\_E\_ACTION

Set a list of non-approved actions.

**Example:** Disallow URI-Actions.

```
[Interactive Features]
NonActions=1
NonAction1=URI
```

## 4.4.7 [Digital Signatures] INI-File Section

### Provider

**Key:** `Provider`

In order to use the signature validation feature of the 3-Heights® PDF Validator Shell, a cryptographic provider is required. The cryptographic provider implements cryptographic algorithms. If signature validation is active but no valid cryptographic provider is configured the 3-Heights® PDF Validator Shell does not start validation and aborts with a return code 3.

The following cryptographic providers are supported:

#### PKCS#11 Provider

The provider configuration string has the following syntax:

`Provider=<PathToDll>;<SlotId>`

- `<PathToDll>` is the path to driver library filename, which is provided by the manufacturer of the HSM, UBS token or smart card. The bitness of the DLL and the 3-Heights® PDF Validator Shell must match. For more information and installation instructions see separate document [TechNotePKCS11.pdf](#).

#### Example:

- openCryptoki soft store on Linux uses `libopencryptoki.so`
- PKCS#11 soft-token on Solaris `libpkcs11.so`
- `<SlotId>` is optional, if it is not defined, it is searched for the first slot that contains a token.

#### 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). The latter is used if the operating system is at least Windows Vista or Windows Server 2008.

The provider configuration string has the following syntax:

`Provider=[<ProviderType>:]<Provider>`

The `<ProviderType>` is optional. An empty `<Provider>` uses the default provider. If CNG is available, `<ProviderType>` and `<Provider>` are both optional.

### Example:

- **Provider=**

The default provider is suitable for all systems where CNG is available.

- **Provider=Microsoft Base Cryptographic Provider v1.0**

- **Provider=PROV\_RSA\_AES:Microsoft Enhanced RSA and AES Cryptographic Provider**

The Microsoft CryptoAPI provider type **PROV\_RSA\_AES** supports the SHA-2 hash algorithms for signature validation. This provider type is recommended in order to validate signatures if neither a PKCS#11 device nor CNG are available.

**Example:** Use openCryptoki to validate signatures. Note that openCryptoki must be installed and the exact location of the PKCS#11 dll depends on your openCryptoki installation.

```
[Digital Signatures]
Provider=/usr/lib64/openssl/libopencryptoki.so
```

Validation of the following signature types is supported:

- adbe.pkcs7.sha1
- adbe.pkcs7.detached

## ValidateNewest (Validate Newest Signature)

**Key:** **ValidateNewest**  
Error code: **CHK\_E\_SIGVAL**

Validate the newest signature of the document. Also see the keys [Provider](#) and [Criteria, Criterion<i>\(Signature Validation Criteria\)</i>](#).

**Example:** Validate the newest signature using openCryptoki.

```
[Digital Signatures]
ValidateNewest=true
Provider=libopencryptoki.so
Criteria=1
Criterion1=Verification
```

## Criteria, Criterion<i> (Signature Validation Criteria)

**Key:** **Criteria**  
**Key:** **Criterion<i>**

List of signature validation criteria. Currently supported are:

**Verification** The signature can be verified, i.e. the cryptographic message syntax (CMS) is correct and the document has not been modified.

**EntireDoc** Require that the document has not been updated after the newest signature.

**Visible** Signature must be visible.

Example: see key [ValidateNewest](#) (Validate Newest Signature).

# 5 Interface Reference

## 5.1 Switches

### 5.1.1 -c1 Set the Conformance Level

**Set the Conformance Level** -c1 <compliance>

This option sets the conformance level against which the document is validated. Valid arguments are:

pdf1.3	PDF Reference 1.3
pdf1.4	PDF Reference 1.4 (Correspronds to Acrobat 5)
pdf1.5	PDF Reference 1.5
pdf1.6	PDF Reference 1.6 (corresponds to Acrobat 7)
pdf1.7	PDF Reference 1.7, ISO 32000-1
pdf2.0	PDF Reference 2.0, ISO 32000-2
pdfa-1a	PDF/A-1a, ISO 19005-1, Level A conformance in Part 1
pdfa-1b	PDF/A-1b, ISO 19005-1, Level B conformance in Part 1
pdfa-2a	PDF/A-2a, ISO 19005-2, Level A conformance in Part 2
pdfa-2b	PDF/A-2b, ISO 19005-2, Level B conformance in Part 2
pdfa-2u	PDF/A-2u, ISO 19005-2, Level U conformance in Part 2
pdfa-3a	PDF/A-3a, ISO 19005-3, Level A conformance in Part 3
pdfa-3b	PDF/A-3b, ISO 19005-3, Level B conformance in Part 3
pdfa-3u	PDF/A-3u, ISO 19005-3, Level U conformance in Part 3
cc1	Determine claimed conformance of document and use it for validation. (default)  If the switch -v is used, the claimed conformance is also printed to stdout. Note that the claimed conformance is not limited to PDF/A.

### 5.1.2 -e Stop on Error

**Stop on Error** -e <name=n>

If <n> is set to 1, then the validation will abort on the first validation error; i.e. the validation process will stop as soon as a problem is found that makes the file non-conforming. This speeds up the validation of non-conforming files.

## Parameter:

<name=n>

- 0 Continue on error (default)
- 1 Stop on validation error

### 5.1.3 -pw Read an Encrypted PDF File

**Read an Encrypted PDF File** -pw <password>

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

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

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

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

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

### 5.1.4 -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), pdf object number, error code, a description, and a counter of how many times the error occurs. The option provides more detailed information than the summary (switch **-rs**).

**Example:** Validate a PDF document against the PDF/A-1a specification, write a detailed report.

```
pdfvalidator -cl pdfa-1a -rd input.pdf

```

**Note:** If no reporting is selected (neither **-rd** nor **-rs**), no textual information is returned about whether the document conforms or not.

## 5.1.5 -r1 Reporting Level

### Reporting Level -r1 <n>

The reporting level describes which type of error messages should be written to standard error (stderr). This option can for example be used to see what replacement fonts are selected for non-embedded fonts. The available values are:

0	do not report	—
1	report errors	file cannot be opened, PDF is corrupted, etc.
2	report errors, warnings	non-embedded font is replaced
3	report errors, warnings, information	page number is about to be set

**Example:** The following command reports all errors and warnings.

```
pdfvalidator -r1 2 input.pdf
```

**Example:** The following command writes all error messages to the log file `error.log`.

```
pdfvalidator -r1 2 input.pdf 2> error.log
```

## 5.1.6 -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 (switch [-rd](#)).

1. The file format (header, trailer, objects, xref, streams) is corrupted.
2. The document doesn't conform to the PDF reference (missing required entries, wrong value types, etc.).
3. The file is encrypted.
4. The document contains device-specific color spaces.
5. The document contains illegal rendering hints (unknown intents, interpolation, transfer and halftone functions).
6. The document contains alternate information (images).
7. The document contains embedded PostScript code.
8. The document contains references to external content (reference XObjects, file attachments, OPI).
9. The document contains fonts without embedded font programs or encoding information (CMAPs).
10. The document contains fonts without appropriate character to Unicode mapping information (ToUnicode maps).
11. The document contains transparency.
12. The document contains unknown annotation types.
13. The document contains multimedia annotations (sound, movies).
14. The document contains hidden, invisible, non-viewable or non-printable annotations.
15. The document contains annotations or form fields with ambiguous or without appropriate appearances.
16. The document contains actions types other than for navigation (launch, JavaScript, ResetForm, etc.).
17. The document's metadata is either missing or inconsistent or corrupt.



- 18. The document doesn't provide appropriate logical structure information.
- 19. The document contains optional content (layers).

**Example:** Validate a PDF document `input.pdf` against the PDF/A-1a specification and write a summary report.

```
pdfvalidator -cl pdfa-1a -rs input.pdf
The document contains fonts without embedded font programs or encoding information (CMAPs).
The documents meta data is either missing or inconsistent or corrupt.
The document doesn't provide appropriate logical structure information.
```

The report is written to standard out. If you would like to write the report into a file, the pipe operator for standard out `>` can be used.

**Example:** Validate a PDF document `input.pdf` against the PDF/A-1a specification, write a summary report, and pipe it into the file `log.txt`.

```
pdfvalidator -cl pdfa-1a -rs input.pdf > log.txt
```

### 5.1.7 -cc1 Claimed Conformance and Level

**Claimed Conformance and Level** -cc1

This switch prints the document's claimed conformance and level to the output.

**Example:** List the claimed conformance level of the PDF document `input.pdf`.

```
pdfvalidator -cc1 input.pdf
Conformance: pdfa-2a
```

### 5.1.8 -p Set custom validation profile

**Set custom validation profile** -p <profile>  
License feature: Custom

Set custom profile to validate conformance to corporate directives. See chapter [Custom Validation Profiles](#) for more information on features and configuration file format.

### 5.1.9 -lk Set License Key

**Set License Key** -lk <key>

Pass a license key to the application at runtime instead of installing it on the system.

```
pdfvalidator -lk X-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX input.pdf output.pdf
```

This is only required in an OEM scenario.

### 5.1.10 -v Verbose Mode

#### Verbose Mode -v

This option turns on the verbose mode.

In the verbose mode, the steps performed by the 3-Heights® PDF Validator Shell are written to the shell. Specifically it writes the following to standard out:

- “Validating file <file name>.”
- “Conformance: <conformance>”, if `-cl ccl` is used
- “The document <does/does not> conform to the <conformance> standard.”

See also section [Validate all Documents in a Directory](#).

## 5.2 Return Codes

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

#### Return Codes

Value	Description
0	Success.
1	Couldn't open input file.
3	Error with given options, e.g. too many parameters.
4	PDF input file does not conform to the specified standard.
10	License error, e.g. invalid license key.

# 6 Coverage

## 6.1 All PDF Versions

### 6.1.1 Lexical Checks

- Structure of tokens such as keywords, names, numbers, strings etc.
- Structure of the cross reference table
- File positions in the trailer dictionary, cross reference table, etc.
- Whether a referenced object has the correct object and generation number
- Length attribute of stream objects

### 6.1.2 Syntactic Checks

- Structure of dictionaries, arrays, indirect objects, streams, etc.
- Compression errors, e.g. CCITT, JPEG, Flate, etc.
- Errors in embedded font programs
- Errors in ICC color profiles

### 6.1.3 Semantic Checks

- Required entries in dictionaries, e.g. Width entry in an image dictionary
- Inherited attributes
- Value of the parent entries in dictionaries, e.g. page objects
- Type of the dictionary entry's value, e.g. integer, string, name
- Whether the object must be indirect or direct, e.g. a page object must be an indirect object
- Order of operators in content streams
- Number of operands of the operators
- Type of operands of the operators
- Value ranges of the operands
- Unknown referenced resources
- Operand stack overflow and underflow
- Inconsistent information, e.g. if an image has a stencil mask and soft mask at the same time
- Conformance to implementation limits defined by the PDF Reference
- Absence of unrendered XFA forms

## 6.2 Checks Specific for PDF/A

### 6.2.1 Lexical Checks

- No header offset
- Presence of a "binary" marker

## 6.2.2 Semantic Checks

All Conformance Levels:

- Presence of a unique file identifier
- Presence of document metadata
- Presence of embedded font programs where needed
- Presence of character to glyph mapping (encoding) information for the fonts
- Presence of an output intent if needed
- Absence of encryption
- Absence of LZW and non-standard filters
- Absence of JavaScript
- Absence of un-allowed annotations
- Absence of un-allowed actions
- Absence of form fields that are generated on the fly
- Absence of embedded PostScript code
- Absence of invisible, hidden or non-printable annotations
- Absence of device specific color spaces
- Absence of unknown rendering intents
- Absence of image interpolation
- Absence of externally referenced information (external streams, reference XObjects, etc.)
- Absence of Open Print Interface (OPI) information
- Absence of alternate images
- Absence of color transfer and half-toning functions

Additional Checks for PDF/A-1

- Absence of JPX
- Absence of layers
- Absence of transparency
- Absence of embedded files
- Absence of XRef streams
- Conformity of metadata

Additional Checks for PDF/A-2

- PDF/A conformance of embedded files
- Consistency of spot colors

Additional Checks for Level A and U (PDF/A-1a, PDF/A-2a, PDF/A-2u, PDF/A-3a, PDF/A-3u)

- Presence of Unicode information of fonts where needed

Additional Checks for Level A (PDF/A-1a, PDF/A-2a, PDF/A-3a)

- Presence of logical structure information (tagging)
- Presence of alternate descriptions of content (replacement text) where needed

## 6.3 Supported PDF Versions

The 3-Heights® PDF Validator Shell currently validates the following versions of the PDF Reference and PDF/A ISO standard:

### Supported PDF Versions

PDF 1.x	PDF Reference 1.3 - 1.6
PDF 1.7	PDF 1.7, ISO 32000-1
PDF 2.0	PDF 2.0, ISO 32000-2
PDF/A-1a	PDF/A-1a, ISO 19005-1, Level A conformance
PDF/A-1b	PDF/A-1b, ISO 19005-1, Level B conformance
PDF/A-2a	PDF/A-2a, ISO 19005-2, Level A conformance
PDF/A-2b	PDF/A-2b, ISO 19005-2, Level B conformance
PDF/A-2u	PDF/A-2u, ISO 19005-2, Level U conformance
PDF/A-3a	PDF/A-3a, ISO 19005-3, Level A conformance
PDF/A-3b	PDF/A-3b, ISO 19005-3, Level B conformance
PDF/A-3u	PDF/A-3u, ISO 19005-3, Level U conformance

# 7 Version History

## 7.1 Patches in Version 6.18

### Patch 6.18.1

- **Improved** performance reading files with excessive number of compressed object streams.

## 7.2 Changes in Version 6

- **Improved** validation of corrupt DCT stream data.

## 7.3 Changes in Version 5

- Custom Validation Profiles
  - **New** key **Linearization** in section **File** to check whether files are linearized.
  - **New** keys **ImageMaxDPI** and **ImageMinDPI** in section **Graphics** to validate the resolution of images.
- **New** additional supported operating system: Windows Server 2019.

## 7.4 Changes in Version 4.12

- **Introduced** license feature **Custom**.
- Custom Validation Profiles
  - **New** key **MaxPageSize** in section **Pages** to disallow pages exceeding the specified size in any dimension.
  - **New** key **RequirePageResources** in section **Pages** to test if pages contain an explicitly associated resource dictionary.
  - **New** key **Embedding**, **EmbeddingExcFonts**, and **EmbeddingExcFont<i>** in section **Fonts** to test the embedding of fonts.
- **Changed** validation of certain numbers: Use lax validation according to the PDF Association's TechNote 0010 for certain numbers that have no effect on the visual appearance of the document.
- **Improved** validation performance, e.g. when reporting many errors or analyzing ICC profiles.
- **Improved** detection of corrupt DCT streams that might cause interoperability issues.
- **New** HTTP proxy setting in the GUI license manager.

## 7.5 Changes in Version 4.11

- **New** support for reading PDF 2.0 documents.

## 7.6 Changes in Version 4.10

- **Updated** validation according to the PDF Association's TechNote 0010, which describes some peer-reviewed resolutions to a variety of ambiguities of corner cases of the PDF/A specifications.
- **Improved** stricter validation of font files of embedded fonts.

- **Improved** stricter validation of logical structure information (PDF/A level A).
- Digital Signatures
  - **Improved** signature validation.
    - More signature formats supported, most notably the new European PAdES norm. The Windows cryptographic provider now supports the same formats as the PKCS#11 provider.
    - Support signature algorithm RSA with SSA-PSS (PKCS#1v2.1).
- **Improved** robustness against corrupt input PDF documents.
- **Changed** option -v to print validation result.

## 7.7 Changes in Version 4.9

- **Improved** support for and robustness against corrupt input PDF documents.
- **Improved** repair of embedded font programs that are corrupt.
- **New** support for OpenType font collections in installed font collection.

## 7.8 Changes in Version 4.8

No functional changes.

## 8 Licensing, Copyright, and Contact

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

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

**Licensing and Copyright** The 3-Heights® PDF Validator Shell is copyrighted. This user's manual is also copyright protected; It may be copied and given away provided that it remains unchanged including the copyright notice.

### Contact

PDF Tools AG  
Brown-Boveri-Strasse 5  
8050 Zürich  
Switzerland  
<http://www.pdf-tools.com>  
[pdfsales@pdf-tools.com](mailto:pdfsales@pdf-tools.com)