

User Manual



3-Heights™ PDF Validator Shell

Version 4.5



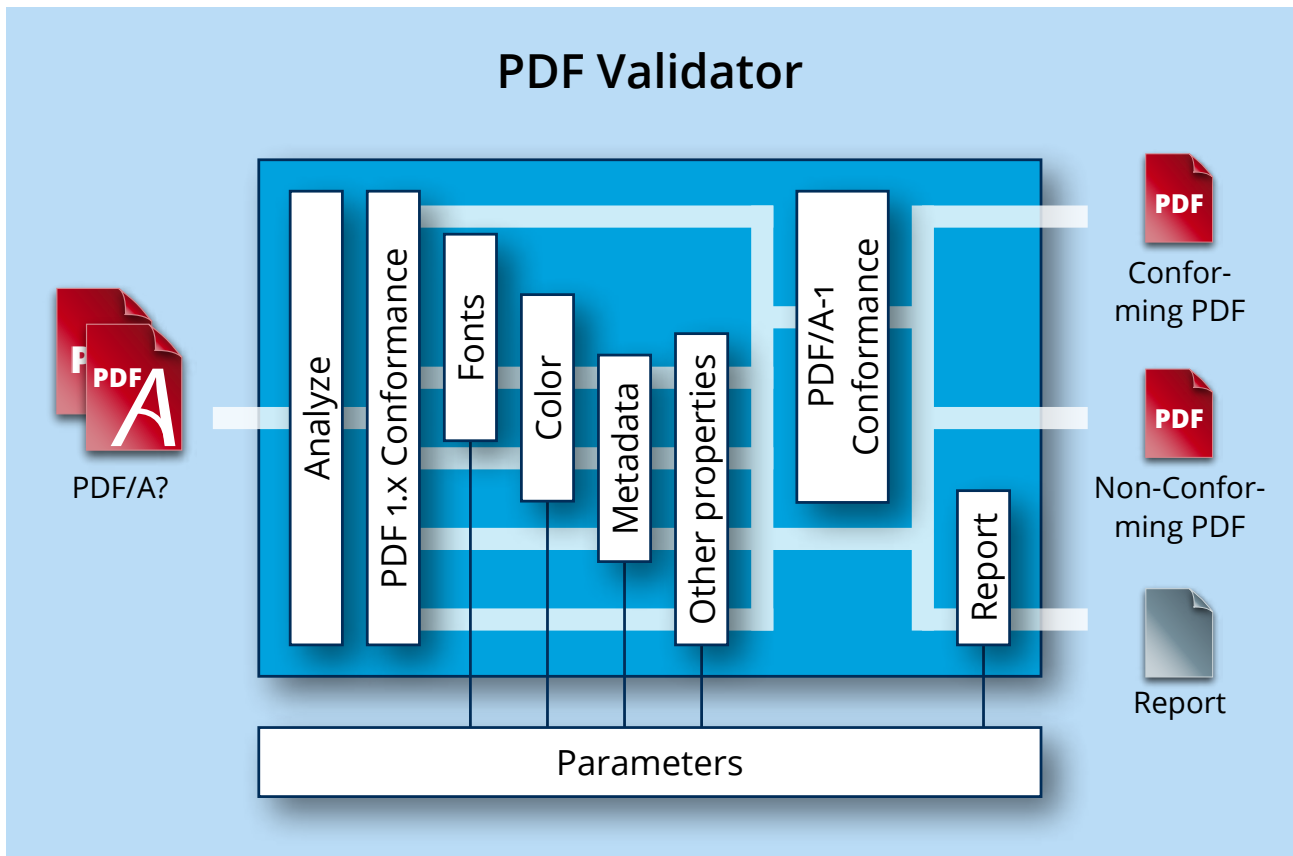
Contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 1 |
| 1.1 | Description | 1 |
| 1.2 | Functions | 2 |
| 1.3 | Operating Systems | 2 |
| 2 | Installation And Deployment | 3 |
| 2.1 | Windows | 3 |
| 2.2 | Unix | 3 |
| 3 | License Management | 4 |
| 3.1 | Graphical License Manager Tool | 4 |
| 3.2 | Command Line License Manager Tool | 5 |
| 3.3 | License Key Storage | 5 |
| 4 | Getting started | 6 |
| 4.1 | Usage | 6 |
| 4.2 | General Settings | 6 |
| | -lk: Set License Key | 6 |
| 4.3 | Validate a Document | 6 |
| 4.4 | What is PDF/A? | 8 |
| 4.5 | Custom Validation Profiles | 9 |
| | Section File | 9 |
| | Section Document | 10 |
| | Section Pages | 11 |
| | Section Graphics | 12 |
| | Section Fonts | 14 |
| | Section Interactive Features | 15 |
| | Section Digital Signatures | 15 |
| 5 | Reference Manual | 16 |
| 5.1 | Switches | 16 |
| | -cl: Set the Conformance Level | 16 |
| | -e: Stop on Error | 17 |
| | -pw: Read an Encrypted PDF File | 17 |
| | -rd: Report Conformance Violations in Detail | 17 |
| | -rl: Reporting Level | 17 |
| | -rs: Report Conformance Violations Summary | 18 |
| | -ccl: Claimed Conformance and Level | 18 |
| | -p: Set custom validation profile | 19 |
| | -lk: Set License Key | 19 |
| | -v: Verbose Mode | 19 |
| 5.2 | Return Codes | 19 |
| 6 | Coverage | 19 |
| 6.1 | All PDF Versions | 19 |
| 6.2 | Checks Specific for PDF/A | 20 |
| 6.3 | Supported PDF Versions | 21 |
| 7 | Licensing and Copyright | 21 |
| 8 | Contact | 21 |

1 Introduction

1.1 Description

The 3-Heights™ PDF Validator Shell safeguards the quality of PDF documents. It checks PDF files for compliance with 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 compliance with 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

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.

General Functions

- Validate PDF documents on the basis of various PDF specifications (PDF 1.4, PDF/A-1, PDF/A-2, PDF/A-3)
- 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 compliance of document
- Validate compliance with corporate directives defined in custom profile

Validation Functions

See chapter [Coverage](#)

Formats

Input Formats

- PDF 1.x (e.g. PDF 1.4, PDF 1.5, etc.)
- PDF/A-1a, PDF/A-1b
- PDF/A-2a, PDF/A-2b, PDF/A-2u
- PDF/A-3a, PDF/A-3b, PDF/A-3u

Compliance

- Standards: ISO 19005-1 (PDF/A-1), ISO 19005-2 (PDF/A-2), ISO 19005-3 (PDF/A-3), ISO 32000 (PDF 1.7)
- Quality assurance: Isartor test suite
- Bavaria test suite (unofficial) 2

1.3 Operating Systems

- Windows XP, Vista, 7, 8, 8.1 - 32 and 64 bit
- Windows Server 2003, 2008, 2008 R2, 2012, 2012 R2 - 32 and 64 bit
- HP-UX 11 and later PA-RISC2.0 32 bit or HP-UX 11i and later ia64 (Itanium) 64 bit
- IBM AIX 5.1 and later (64 bit)
- Linux (32 and 64 bit)
- Mac OS X 10.4 and later (32 and 64 bit)
- Sun Solaris 2.8 and later, SPARC and Intel
- FreeBSD 4.7 and later 32 bit or FreeBSD 9.3 and later 64 bit (on request)

2 Installation And Deployment

2.1 Windows

The retail version of the 3-Heights™ PDF Validator Shell comes as a ZIP archive containing various files including runtime binary executable code, files required for the developer, documentation and license terms.

1. Download the ZIP archive of the product from your download account at <http://www.pdf-tools.com>.
2. Unzip the file using a tool like WinZip available from WinZip Computing, Inc. at <http://www.winzip.com> to a directory on your hard disk where your program files reside (e.g. *C:\Program Files\PDF Tools AG*).
3. Check the appropriate option to preserve file paths (folder names). The unzip process now creates the following subdirectories:

bin: Contains the runtime executable binary code.

doc: Contains documentation files.

There is the option to download the software as MSI file, which makes the installation easier.

4. To easily use the 3-Heights™ PDF Validator Shell from a shell, the directory needs to be included in the "Path" environment variable.
5. Optionally register your license key using the [License Manager](#).

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". If the environment variable "Path" does not exist, create it.

2.2 Unix

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

All Unix Platforms

1. Unpack the archive in an installation directory, e.g. */opt/pdf-tools.com/*
2. Copy or link the executable into one of the standard executable directories, e.g:

```
ln -s /opt/pdf-tools.com/bin/pdfvalidator /usr/bin
```
3. Verify that the GNU shared libraries required by the product are available on your system:
 - On Linux: `ldd pdfvalidator`
 - On AIX: `dump -H pdfvalidator`

In case you have not installed the GNU shared libraries yet, proceed as follows:

- (a) Go to <http://www.pdf-tools.com> and navigate to "Support" → "Resouces".
 - (b) Download the GNU shared libraries for your platform.
 - (c) Extract the archive and copy or link the libraries into your library directory, e.g */usr/lib* or */usr/lib64*.
 - (d) Verify that the GNU shared libraries required by the product are available on your system now.
4. Optionally register your license key using the [Command Line License Manager Tool](#).

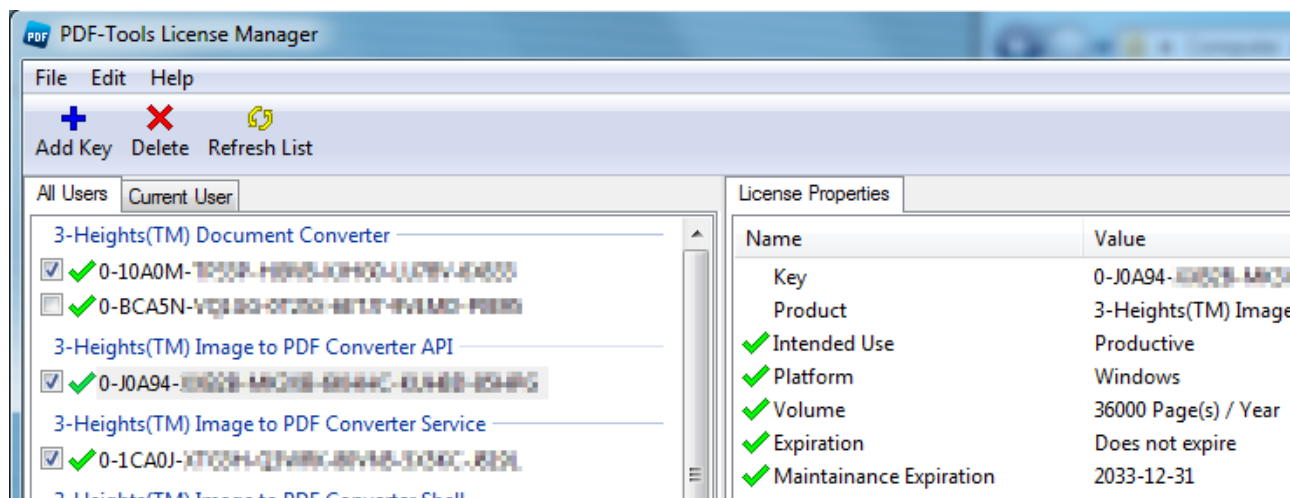
3 License Management

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

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

3.1 Graphical License Manager Tool

The GUI tool *LicenseManager.exe* is located in the *bin* directory of the product kit.



List all installed license keys

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

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

Add and delete license keys

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

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

Display the properties of a license

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

Select between different license keys for a single product

More than one license key can be installed for a specific product. The checkbox on the left side in the license list marks the currently active license key.

3.2 Command Line License Manager Tool

The command line license manager tool `licmgr` is available in the `bin` directory for all platforms except Windows. A complete description of all commands and options can be obtained by running the program without parameters:

```
licmgr
```

List all installed license keys

```
licmgr list
```

Add and delete license keys

Install new license key:

```
licmgr store X-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

Delete old license key:

```
licmgr delete X-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

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

- `g`: For all users
- `u`: Current user

Select between different license keys for a single product

```
licmgr select X-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

3.3 License Key Storage

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

Windows

The license keys are stored in the registry:

- `HKLM\Software\PDF Tools AG` (for all users)
- `HKCU\Software\PDF Tools AG` (for the current user)

Mac OS X

The license keys are stored in the file system:

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

Unix/Linux

The license keys are stored in the file system:

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

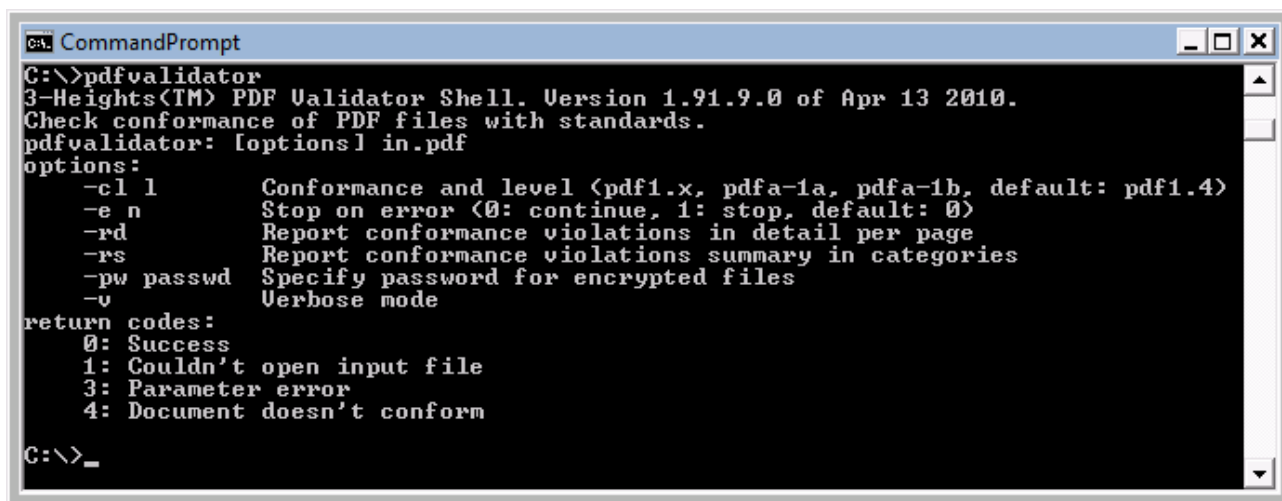
Note: The user, group and permissions of those directories are set explicitly by the license manager tool. It may be necessary to change permissions to make the licenses readable for all users. Example:

```
chmod -R go+rx /etc/opt/pdf-tools
```

4 Getting started

4.1 Usage

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



```
CommandPrompt
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 General Settings

-lk: Set License 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-XXXXX input.pdf output.pdf
```

This is only required in an OEM scenario.

4.3 Validate a Document

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:

- Reporting type (*-rs* or *-rd*)
- PDF file to validate

Optional Parameters are:

- Conformance level (*-cl*)
- 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 -rd 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.

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 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. If multiple documents are validated, the `-v` switch should always be set, otherwise it's unclear to which file the messages belong.

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

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 is compliant 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 compliant **
)
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 compliant **
Couldnt open PDF file Bbb.pdf.
Bbb.pdf : ** NOT compliant **
```

4.4 What is PDF/A?

PDF/A-1

PDF/A is an ISO Standard for using PDF format for the long-term archiving of electronic documents. 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-dependant 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

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)

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

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.

For additional information about PDF/A please visit: <http://www.pdf-tools.com/pdf/pdfa-longterm-archiving-iso-19005-pdf.aspx>.

4.5 Custom Validation Profiles

In addition to checking documents for compliance with the PDF Reference and PDF ISO standards, the 3-Heights™ PDF Validator Shell can ensure compliance with 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.

4.5.1 Section File

File Size Limit 1

```
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
```

File Size Limit 2

```
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
```

Maximum PDF Version

```
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
```

Minimum PDF Version

```
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.

Values:

- `true`: Raise error, if file is not encrypted.
- `false`: Raise error, if file is encrypted.

Example: Dis-allow encrypted files.

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

4.5.2 Section Document

Non-Approved PDF Creators

```
Key: NonCreators, NonCreatorX
Error Code: CHK_E_CREATOR
```

Non-approved PDF creators are defined by setting NonCreator='n', where 'n' is the count, i.e. a value larger than 0. Names of the creators are defined using NonCreator1=Name_1 to NonCreator'n'=Name_n.

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

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

Non-Approved PDF Producers

```
Key: NonProducers, 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
```

Allowed Embedded File Types

```
Key: EmbeddedFiles, 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
```

4.5.3 Section Pages

Approved Page Sizes

```
Key: PageSizes, PageSizeX
Error Code: CHK_E_PAGESIZE
```

Approved page sizes are specified by setting PageSizes='n', where 'n' is the count, i.e. a value larger than 0. Sizes are defined using PageSize1=Size_1 to

Values:

- **Letter:** US Letter page 8.5 x 11 in.
- **A'n':** 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, 1mm), unless the key SizeTolerance is specified.

Example

```
[Pages]
PageSizes=4
PageSize1=A0
PageSize2=A3
PageSize3=15.53 x 15.35 in
PageSize4=181 x 181 mm
```

Tolerance for Page Size Comparison

```
Key: SizeTolerance
Default: 3 (~1mm)
Error Code: n/a
```

Tolerance used for page size comparison.

Values:

- **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%
```

Empty Page

```
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.

Values:

- **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
```

4.5.4 Section Graphics

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.

Example: Set the maximum allowed resolution to 602DPI.

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

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
```

Color for Scanned Images

```
Key: ScanColor  
Error Code: CHK_E_SCANCLR
```

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

Values:

- **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
```

OCR Text

```
Key: OCRText  
Error Code: CHK_E_OCRTTEXT
```

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

Values:

- **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
```

Prohibit Color

```
Key: ProhibitColor  
Error Code: CHK_E_CLRUSED
```

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

Values:

- **true**: Raise error, if page contains color.
- **false**: Do not check for color.

Example

```
[Graphics]  
ProhibitColor=true
```

Layers

```
Key: Layers  
Error Code: CHK_E_LAYERS
```

Use the key Layers to disallow layers.

Values:

- **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
```

Hidden Layers

```
Key: HiddenLayers  
Error Code: CHK_E_HIDDENLAYERS
```

Use the key HiddenLayers to disallow hidden layers.

Values:

- **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
```

4.5.5 Section Fonts

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.

Approved Fonts

```
Key: Fonts , FontX
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 `Font1=Font_Name_1` to `Font'n=Font_Name_n`. 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
```

Non-Approved Fonts

```
Key: NonFonts , NonFontX
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.

Values:

- `true`: Raise error, if embedded font is not subsetting.
- `false`: Raise error, if embedded font is subsetting.

Example: Require all fonts to be subsetting.

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

Embedding of Non-Standard Fonts

```
Key: NonStdEmbedded
Error Code: CHK_E_FNTEMB
```

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

Values:

- **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
```

4.5.6 Section Interactive Features

Approved Annotations

```
Key: Annotations, AnnotationX
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
```

Non-Approved Actions

```
Key: NonActions, NonActionX
Error Code: CHK_E_ACTION
```

Set a list of non-approved actions

Example: Disallow URI-Actions.

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

4.5.7 Section Digital Signatures

Provider

```
Key: Provider
Error Code: n/a
```

Set the cryptographic provider used for signature validation.

Example: Use openCryptoki to validate signatures (note that openCryptoki must be installed):

```
[Digital Signatures]
Provider=libopencryptoki.so
```

Validate Newest Signature

```
Key: ValidateNewest  
Error Code: CHK_E_SIGVAL
```

Validate the newest signature of the document. Also see the keys Provider and Criteria.

Example: Validate the newest signature using openCryptoki.

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

Signature Validation Criteria

```
Key: Criteria, CriterionX  
Error Code: n/a
```

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)

5 Reference Manual

5.1 Switches

-cl: Set the Conformance Level

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

| | |
|-------------------------|---|
| PDF 1.4 | PDF Reference 1.4(Corresponds to Acrobat 5) |
| pdfa-1a | PDF/A 1a, ISO 19005-1, Level A compliance in Part 1 |
| pdfa-1b | PDF/A 1b, ISO 19005-1, Level B compliance in Part 1 |
| pdfa-2a | PDF/A 2a, ISO 19005-2, Level A compliance in Part 2 |
| pdfa-2b | PDF/A 2b, ISO 19005-2, Level B compliance in Part 2 (default) |
| pdfa-2u | PDF/A 2u, ISO 19005-2, Level U compliance in Part 2 |
| pdfa-3a | PDF/A 3a, ISO 19005-3, Level A compliance in Part 3 |
| pdfa-3b | PDF/A 3b, ISO 19005-3, Level B compliance in Part 3 |
| pdfa-3u | PDF/A 3u, ISO 19005-3, Level U compliance in Part 3 |
| ccl | Determine claimed compliance of document and use it for validation. If the switch -v is used, the claimed compliance is also printed to stdout. Note that the claimed compliance is not limited to PDF/A. |

-e: Stop on Error

This option defines whether the validation process should be stop after the first error is found. The validation process will not stop on warnings, but only on errors. This option has two supported arguments:

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

-pw: Read an Encrypted PDF File

When the input PDF file is encrypted and has a user password set, (the password to open the PDF) the password can be provided as parameter of the switch `-pw`.

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 wasnt correct.
```

-rd: Report Conformance Violations in Detail

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 (switch `-rs`).

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

```
pdfvalidator -cl pdfa-1a -rd input.pdf
0, 0x80410604, "The key Metadata is required but missing.", 1
0, 0x80410604, "The key MarkInfo is required but missing.", 1
1, 0x00418704, "The font Arial-BoldMT must be embedded.", 1
1, 0x00418704, "The font TimesNewRomanPS-BoldMT must be embedded.", 1
1, 0x00418704, "The font Arial-BlackItalic must be embedded.", 1
1, 0x83410612, "The document does not conform to the requested standard.", 1
```

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

-rl: Reporting Level

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 -rl 2 input.pdf
```

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

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

-rs: Report Conformance Violations Summary

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, write a summary report.

```
pdfvalidator -cl pdfa-1a -rs input.pdf
The document contains fonts without embedded font programs or encoding in
The documents meta data is either missing or inconsistent or corrupt.
The document doesnt 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
```

-ccl: Claimed Conformance and Level

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 -ccl input.pdf
Conformance: pdfa-2a
```

-p: Set custom validation profile

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

-lk: Set License Key

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

-v: Verbose Mode

This option turns on the verbose mode.

In the verbose mode, the steps performed by the Validator are written to the shell. In particular it writes "Validating file {file name}" to standard out. See also chapter [Validate all Documents in a Directory](#).

5.2 Return Codes

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

| Table: Return Codes | |
|---------------------|--|
| Value | Description |
| 0 | Success |
| 1 | Couldn't open input file |
| 3 | Error with switch or too many parameters |
| 4 | PDF Input file is encrypted and password is missing or incorrect |

6 Coverage

6.1 All PDF Versions

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

Syntactic Checks

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

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

6.2 Checks Specific for PDF/A

Lexical Checks

- No header offset
- Presence of a “binary” marker

Semantic Checks

All Compliance Levels:

- Presence of a unique file identifier
- Presence of document meta data
- 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 filters
- Absence of Java scripts
- 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

Additional Checks for PDF/A-1a, PDF/A-2a, PDF/A-2u, PDF/A-3a, PDF/A-3u

- Presence of Unicode information where needed

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

- Presence of document structure information (tagging)

6.3 Supported PDF Versions

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

| | |
|----------|---|
| PDF 1.x | PDF Reference 1.1 - 1.6 |
| PDF 1.7 | PDF 1.7, ISO 32000-1 |
| PDF/A-1a | PDF/A 1a, ISO 19005-1, Level A compliance |
| PDF/A-1b | PDF/A 1b, ISO 19005-1, Level B compliance |
| PDF/A-2a | PDF/A 2a, ISO 19005-2, Level A compliance |
| PDF/A-2b | PDF/A 2b, ISO 19005-2, Level B compliance |
| PDF/A-2u | PDF/A 2u, ISO 19005-2, Level U compliance |
| PDF/A-3a | PDF/A 3a, ISO 19005-3, Level A compliance |
| PDF/A-3b | PDF/A 3b, ISO 19005-3, Level B compliance |
| PDF/A-3u | PDF/A 3u, ISO 19005-3, Level U compliance |

7 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.

8 Contact

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