User Manual

3-Heights®
PDF Security Service

PDF/A conform

Version 6.27.2

pdftools

Contents

1 Introduction

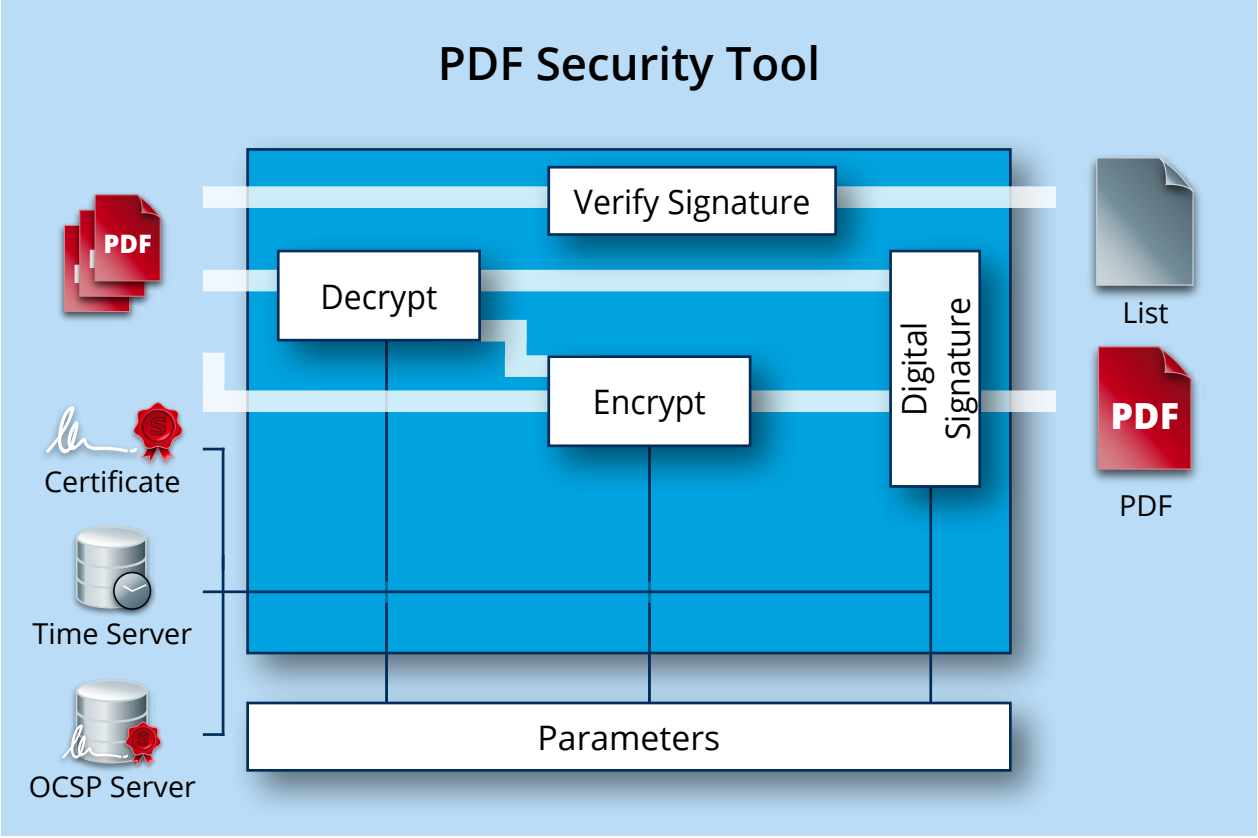
The 3-Heights® PDF Security Service is a ready-to-use product that allows to install a Windows NT service process to automatically deal with the security of PDF documents from watched folders.

1.1 Description

The 3-Heights® PDF Security Service enables the application of digital signatures to PDF documents and their subsequent protection through setting passwords and user authorizations.

Both standard signatures and qualified signatures that use signature cards (“smartcards”, “USB tokens”, “HSM”) can be used.

PDF documents used in professional circumstances contain important information that needs to be protected against misuse and unintentional alteration. This is achieved by protecting PDF documents through encryption and user authorization rights.



When exchanging electronic documents, the ability to ascertain that a document is authentic and has not been manipulated on its way from sender to recipient is of particular importance. This is only achievable through the use of electronic signatures.

1.2 Functions

The 3-Heights® PDF Security Service enables users to encrypt and—if the passwords are known—decrypt PDF documents. The tool can set and cancel all known PDF user authorizations. For instance, it can set an owner password so that only authorized users can edit and change the document. A user password ensures that only authorized

users have access to the document's content. The tool's signature module allows the user to apply, read, and verify both classic digital signatures and MDP (modification detection and prevention) signatures. The visibility and visual appearance of digital signatures can be adapted to suit requirements. The tool also supports customized signature handlers and types.

1.2.1 Features

- Apply simple, advanced, and qualified electronic signatures
  - PDF/A conforming signatures
  - Support European signature standards
  - Signature types
    - Document signatures to "digitally sign" documents
    - Modification detection & prevention (MDP) signatures to "certify" documents
    - Document timestamp signatures to "timestamp" documents
  - Apply PAdES-B-LTA (long-term availability and integrity of validation material) and PAdES-LTV (Long-Term Validation) signatures
    - Embedded trust chain, timestamp, and revocation information (OCSP, CRL)
    - Extend the longevity of existing signatures
    - Add signature validation material to the document security store (DSS)
  - Add an optional visual appearance of the signature (page, size, color, position, text, background image, etc.)
  - Cache OCSP, CRL, and other data for mass-signing
  - Various types of cryptographic providers
    - Windows certificate store
    - Hardware such as hardware security module (HSM), smartcards, and USB tokens
    - Online signature services
      - myBica Digital Signing Service
      - Swisscom All-in Signing Service
      - GlobalSign Digital Signing Service
      - QuoVadis sealsign
  - Multiple signatures
- Encrypt and decrypt PDF documents
  - Set document restrictions, including:
    - Print document
    - Modify document content
    - Extract or copy content
    - Add comments
    - Fill in form fields
    - Extract content for accessibility
    - Assemble documents
    - Print in high resolution
  - Set crypt and stream filters
  - Set encryption strength
  - Set owner and user password
- Set document metadata
- Optimize for the web (linearize) (not for PDF 2.0)

1.2.2 Formats

Input formats

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

Output formats

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

1.2.3 Conformance

Standards:

- ISO 32000-1 (PDF 1.7)
- ISO 32000-2 (PDF 2.0)
- ISO 19005-1 (PDF/A-1)
- ISO 19005-2 (PDF/A-2)
- ISO 19005-3 (PDF/A-3)
- PAdES (ETSI EN 319 142) signature levels B-B, B-T, B-LT, B-LTA, CMS
- Legacy PAdES baseline signature (ETSI TS 103 172) B-Level, T-Level, LT-Level, and LTA-Level
- Legacy PAdES (ETSI TS 102 778) Part 2 (PAdES Basic), Part 3 (PAdES-BES), and Part 4 (PAdES-LTV, Long-Term Validation)
- Long-term signature profiles for PAdES (ISO 14533-3)
- Cryptographic Suites (ETSI TS 119 312)

1.3 Operating systems

The 3-Heights® PDF Security Service is available for the following operating systems:

- Windows Client 7+ | x86 and x64
- Windows Server 2008, 2008 R2, 2012, 2012 R2, 2016, 2019, 2022 | x86 and x64

'+' indicates the minimum supported version.

1.4 Digital signatures

1.4.1 Overview

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

1.4.2 Terminology

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

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

Abbreviations

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

1.4.3 Why digitally signing?

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

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

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

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

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

This part of the signature is entirely created by the 3-Heights® PDF Security Service.

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

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

The 3-Heights® PDF Security Service supports the following types of digital signatures:

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

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

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

1.4.4 What is an electronic signature?

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

Three types of electronic signatures are distinguished:

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

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

Simple electronic signature

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

Example:

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

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

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

Advanced electronic signature

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

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

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

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

This brings the following advantages over a simple electronic signature:

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

Qualified electronic signature

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

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

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

This brings the following advantages over an advanced electronic signature:

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

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

1.4.5 Creating electronic signatures

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

Preparation steps

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

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

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

Application of the signature

Apply the signature by providing the following information:

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

Example: Steps to add an electronic document signature

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

To add an electronic document signature with the 3-Heights® PDF Security Service, the following steps need to be done:

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

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

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

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

2 Installation

2.1 Overview

The PDF Security Service is configured by the file `PdfSecureSvr.ini`, which needs to be located in the same directory as the executable `pdfsecuresvr.exe`. Before starting the service, the configuration file needs to be adjusted. The procedure to perform this step is described in PdfSecureSvr.ini configuration file.

Once configured, the service can be created, started, paused, continued, stopped, and deleted via the command line. To use the create and delete functions, administrator permissions are required. To start and stop the service, operator permissions are required.

When the service is running, it processes PDF documents that are copied or moved into watched folders.

They are then renamed and moved to the folder Jobs. The renaming gives the PDF a 16 character long timestamp to create unique job tickets. This ensures there are no conflicts with documents that have the same name.

2.2 Windows

The 3-Heights® PDF Security Service comes as an MSI installer.

To install the software, proceed as follows:

1. You need administrator rights to install this software.
2. Log in to your download account at `https://www.pdf-tools.com`. Select the product "PDF Security Service". If you have no active downloads available or cannot log in, please contact `pdfsales@pdf-tools.com` for assistance.
   You can find different versions of the product available. Download the version that is selected by default. You can select a different version.
   The product comes as an MSI (Microsoft Installer) package that provides an installation routine for installing and uninstalling the 3-Heights® PDF Security Service.
   The package installs the 64-bit version, which runs on 64-bit platforms only.
3. Start the MSI package and follow the steps in the installation routine.
4. Ensure the cache directory exists as described in Special directories.
5. If you want to sign documents, set up your cryptographic provider as described in Cryptographic provider.

2.3 Uninstall

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

2.4 Note about the evaluation license

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

2.5 Special directories

2.5.1 Directory for temporary files

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

The directory is determined as follows. The product checks for the existence of environment variables in the following order and uses the first path found:

Windows

1. The path specified by the %TMP% environment variable
2. The path specified by the %TEMP% environment variable
3. The path specified by the %USERPROFILE% environment variable
4. The Windows directory

2.5.2 Cache directory

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

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

Windows

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

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

2.5.3 Font directories

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

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

Windows

1. `%SystemRoot%\Fonts`
2. User fonts listed in the registry key `\HKEY_CURRENT_USER\Software\Microsoft\Windows NT\CurrentVersion\Fonts`. This includes user specific fonts from `C:\Users\<user>\AppData\Local\Microsoft\Windows\Fonts` and app specific fonts from `C:\Program Files\WindowsApps`
3. `Fonts` directory, which must be a direct subdirectory of where `pdfsecuresvr.exe` resides.

3 License management

The 3-Heights® PDF Security Service requires a valid license in order to run correctly. If no license key is set or the license is not valid, then an error message is printed to the service log.

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

3.1 License features

The functionality of the 3-Heights® PDF Security Service contains one area to which the following license feature is assigned:

`Signature` Create and enhance signatures.

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 Configuration

Before starting the PDF Security Service for the first time, the file `PdfSecureSvr.ini` needs to be modified. Editing this file while the PDF Security Service is running has no impact. The service first needs to be stopped and restarted after the modification. When opening this file with a text editor, it looks like this:

```
[PdfSecureSvr]
AutoDelete=True
Threads=3
Thread1=-w d:\wfEncryptAllowPrint -o owner -p pd
Thread2=-w d:\wfEncryptAllowChange -o owner2 -p mc
Thread3=-w d:\wfEncryptUserPwd -pw password -o owner3 -user user3
```

The meaning of these keys and values in this example is as follows:

`AutoDelete=True` This option automatically deletes a PDF file after it is processed successfully. When set to `False`, the processed file is copied to the sub directory `Succeeded`.

`Threads=` The given value stands for the total number of concurrent threads. Each thread can have its own assigned settings. One thread corresponds to one watched folder.

`Threads1=` Sets the options such as name of watched folder and settings etc. for thread 1.

`-w d:\wfEncryptAllowPrint` Creates a watched folder with the given name for this thread. The path must be an absolute path. Network mapped drive letters or relative paths or driver letters mapped via the `subst` command are not recognized, because the service process per default runs under the "LocalSystem" account. (The user can be changed as described in Managing the service.)

`-o owner` Sets the owner password for the output document to "`owner`".

`-p ...` Sets the permission flags for the output document.

> **Note:** Any string, such as a file name, that contains spaces must be enclosed in quotation marks. For example, if the watched folder contains spaces in its path, the entire path needs to be quoted: `-w "C:\A path\with spaces"`.

4.1.1 Retrieve information about available options and settings

A quick overview of all configuration options and service control commands that the 3-Heights® PDF Security Service supports can be output in the form of a usage message in the command line.

To display this information, open a Windows command line (`cmd.exe`) and then type:

```
pdfsecuresvr
```

(See also Service control commands)

A short overview of all the options that can be configured in the `PdfSecureSvr.ini` is displayed when typing the following in a Windows command line:

```
pdfsecuresvr -i
```

(See also Encryption, Digital signatures, and General switches)

4.2 Managing the service

Once the configuration is done, the service can be started and controlled by executing `pdfsecuresvr.exe` on the command line. The path can be omitted if the `pdfsecuresvr.exe` is included in the %PATH% environment variable.

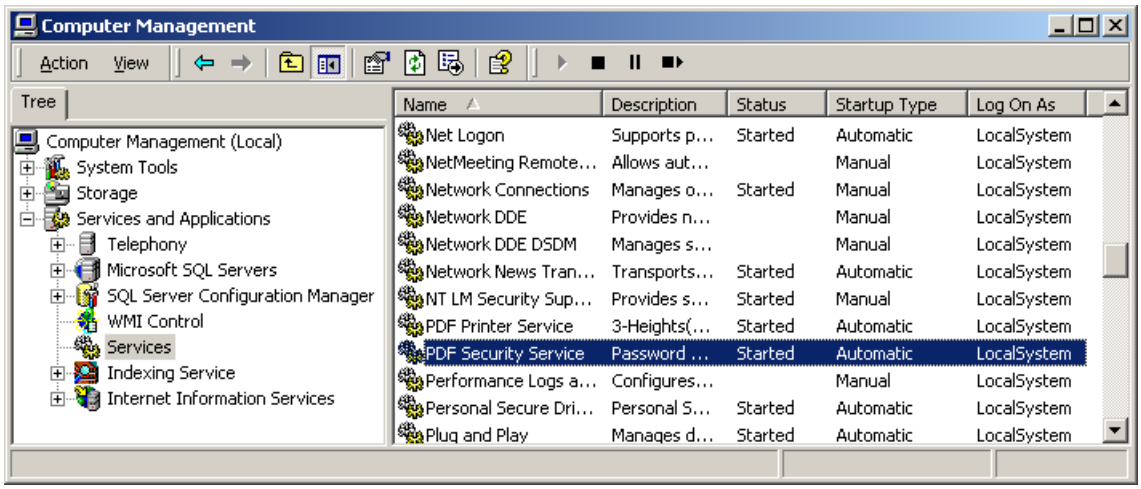
> **Note:** It is essential that the executable `pdfsecuresvr.exe` and the configuration file `PdfSecureSvr.ini` be on a non-mapped drive.

> **Note:** To create or delete the service, administrator permissions are required.

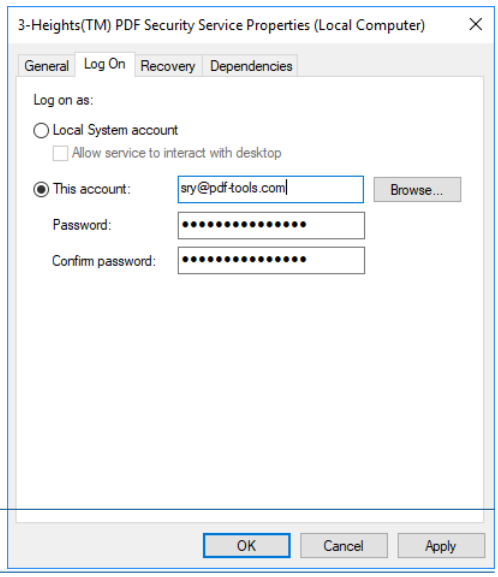
1. To create the service, use the `-c` option.

   ```
   pdfsecuresvr -c
   ```

   After executing this command, the service is created. It is now visible in the "Computer Management" window under "Services". To open the "Computer Management" window, go to Start → Control Panel → Administrative Tools → Computer Management or simply right-click the icon "My Computer" on the desktop and select "manage". If the services is created correctly, it appears as "3-Heights® PDF Security Service" as shown in the image below.



2. By default, the 3-Heights® PDF Security Service runs in the "LocalSystem" account. After the service has been created, the user can be changed.
   This is required in a situation where a network share is used as a watched folder and the process needs to run under a user with the appropriate access permission rights, since the account "LocalSystem" does not have any permissions on remote systems.
   If digital signatures are to be applied, the certificates may be bound to a user or even an interactive session. Check your Cryptographic provider for requirements regarding the user.
   To change the user, right-click the service in the Services window and select "Properties". Then change the user in the tab "Log On".
3. Once created, the service can be started with the option `-s`.




```
pdfsecuresvr -s
```

4. Now the 3-Heights® PDF Security Service is up and running, and files can be moved, copied or drag-and-dropped into the watched folder.
5. To stop the service, use the option `-t`.

```
pdfsecuresvr -t
```

   To restart, use `-s` again.
6. To delete the service, use the option `-d`.

```
pdfsecuresvr -d
```

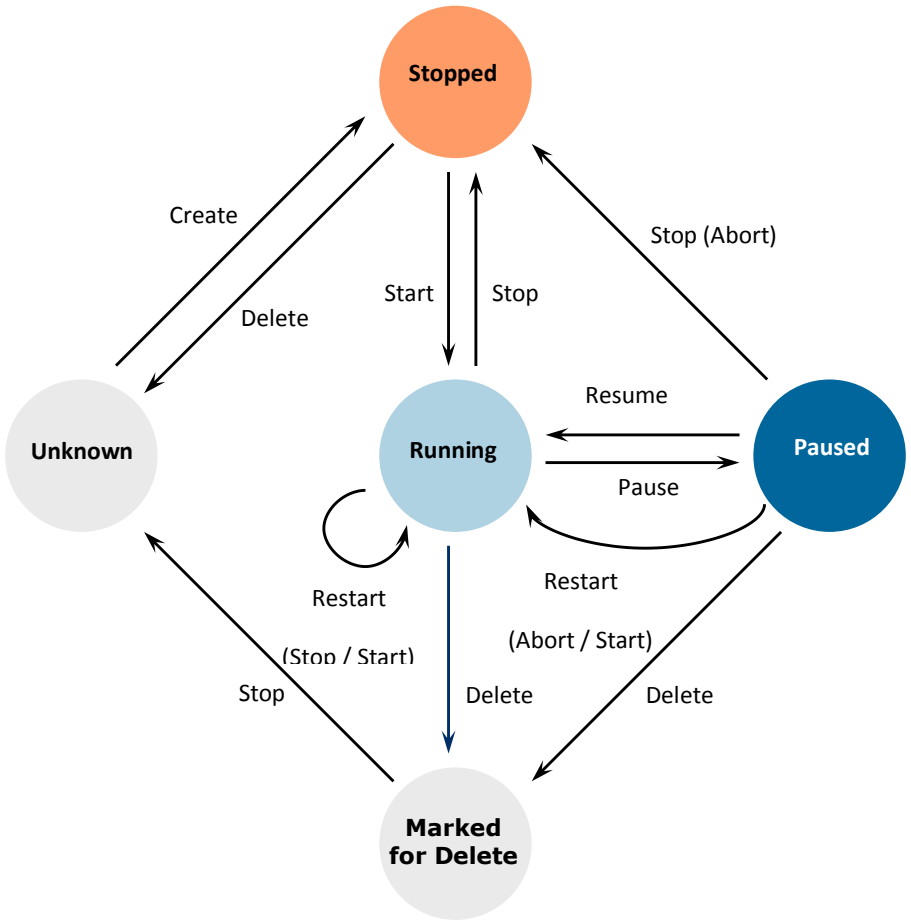
Due to the nature of a security tool, the setup of the service is required to follow some security guide lines. Most important is that the configuration file `PdfSecureSvr.ini` is only accessible by authorized users, i.e. it is not on a publicly shared folder.

4.2.1 Service state diagram

The 3-Heights® PDF Security Service behaves as described in the state diagram below.

If "Stop" is called when the service is in the "Paused" state, the current job is aborted. This means the current page is finished processing, then the job is terminated.

If "Stop" is called when the service is the "Running" state, the current job (all pages) is finished. Then the service is stopped.



4.3 Using the service

Once the service is created and started, the watched folders configured in `PdfSecureSvr.ini` are created automatically. In each watched folder, the following subfolders are created:

- `Jobs`
- `InProgress`
- `Succeeded`
- `Failed`
- `Secured`
- `Logs`
- `Temp`

When a file is moved, copied, or drag-and-dropped into the configured watched folder, the service performs these tasks:

1. Each file is moved to the `Jobs` subfolder. While moving, the file is renamed by adding a 16 character long job-number prefix. This ensures a well-defined processing order and unique file names.
2. A worker-thread takes the file from the `Jobs` folder and moves it to `InProgress`. The file is then processed.
3. Depending on the outcome of the processing, the following is done:

   The file was processed successfully
   - The input file is moved to the `Succeeded` folder or it is deleted, depending on whether `AutoDelete` or `AutoDeleteAll` is set to `true` or `false` in the configuration file `PdfSecureSvr.ini`.
   - The output document is stored in `Secured`.

   The file was not processed successfully

- The input file is moved to the `Failed` folder or it is deleted, depending on whether `AutoDeleteALL` is set to `true` or `false` in the configuration file `PdfSecureSvr.ini`.

4. In any case, an entry in the log file of this thread is created.

4.4 Log files

There are two types of log files.

The log file per thread Each thread (watched folder) has a log file. The log file resides in the same directory as the `pdfsecuresvr.exe` executable and the `PdfSecureSvr.ini` configuration file. It is named `PdfSecureSvr-log-‹n›.txt`, where the number of the log file `‹n›` is increased whenever the service is re-started. The log file is locked by the service as long as the service is running.

- The log file contains general messages (including a timestamp that is not shown here) such as:

```
- [1] Worker thread for directory C:\pdfsecuresvr\Folder started.
```

- Error messages such as:

```
* Error 0 while opening file C:\pdfsecuresvr\Folder\InProgress\Job-...
```

5 User guide

5.1 Encryption

5.1.1 Encryption and how it works in PDF

A PDF document can be encrypted to protect its contents from unauthorized access. The encryption process applies encryption to all streams (e.g. images) and strings, but not to other items in the PDF document. This means the structure of the PDF document is accessible, but the content of its pages is encrypted.

When encryption is used in PDF, a security handler must be selected. The 3-Heights® PDF Security Service always uses the standard security handler that, according to the PDF Specification, has to be supported by any software that can process encrypted PDF documents.

For more detailed information about PDF encryption in general, see PDF Reference, chapter 3.5.

5.1.2 Owner password and user password

The standard security handler allows access permissions and up to two passwords to be specified for a document: An owner password and a user password.

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

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

The following table shows the four possible combinations of passwords and how an application processing such a PDF document behaves.

Owner and user passwords

UserPwd	OwnerPwd	Behavior
none	none	Everyone can read. Everyone can change security settings. (No encryption)
none	set	Everyone can read. The user password is an empty string. Owner password required to change security settings.
set	none	User password required to read. The owner password is equal to the user password. User password required to change security settings.
set	set	User or owner password required to read. Owner password required to change security settings.

5.1.3 Permission flags

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

These access permission flags are:

- Modifying the content of the document
- Copying or extracting text and graphics from the document
- Adding or modifying text annotations and interactive form fields
- Printing the document (low or high quality)
- Filling in forms and digitally signing the document
- Assembling the document (inserting, rotating, deleting pages, etc.)

5.1.4 Encrypting a PDF document

If either of the passwords or permission flags is set, the document is encrypted.

If only a user password is set, but no owner password and no permission flags, the owner password is equal to the user password and all permissions are granted.

Example: Create a document where only low resolution printing is allowed

```
-o ownerpassword -p p
```

Example: Create a document where only low resolution printing is allowed and the user is prompted for a password upon opening the document. The user must therefore know either the user or the owner password.

```
-o ownerpassword -u userpassword -p p
```

5.1.5 Reading an encrypted PDF document

A PDF document that is not encrypted or protected with an owner password only can be read and decrypted by the 3-Heights® PDF Security Service without providing a password.

A PDF document that is protected by a user password can only be opened if either the user or the owner password is provided using the `-pw` option. Technically, it does not matter later on which of the two passwords was provided, because both grant full access to the document. However, it is up to the application programmer to distinguish between input documents that are password protected or not.

5.1.6 How secure is PDF encryption?

Any PDF application that is to process or display a PDF document must be able to read and decrypt the contents of the pages to be able to display them. Technically, it cannot display an encrypted text or image without first decrypting it. A PDF application program has therefore full access to any PDF document it can decrypt and display.

PDF application programs such as all products of the PDF Security Service family or Adobe Acrobat, can open and decrypt PDF documents that have an owner password but no user password, without knowing that password. Otherwise, they couldn't display the document. The application at that point has full access to the document. However, this does not imply the user of this application is given the same access rights. The user should only be given the

access permissions defined by the permission flags and the password provided. Any PDF application that behaves different from that can allow for changing the security settings or completely removing encryption from the document as long as the original document does not have a user password.

The user password protects the document, so that it only can be opened if the user or owner password is known. No PDF application program can open a user-password protected PDF document without providing the password. The security of such a document, however, strongly depends on the password itself. Like in most password-related situations, insecure passwords can easily be found programmatically. For example, a brute force attempt testing all passwords that either exist as word in a dictionary or have less than six characters only takes minutes.

5.1.7 Setting permission flags for Acrobat

In Acrobat 7, there are four different fields/check boxes that can be set. The corresponding setting is shown in brackets using the parameter `-p`.

1. Printing allowed
   - None (`""`)
   - Low resolution (`p`)
   - High resolution (`pd`)
2. Changes allowed:
   - None (`""`)
   - Inserting, deleting and rotating pages (`a`)
   - Filling in form fields and signing existing signature fields (`f`)
   - Commenting, filling in form fields, and signing existing signature fields (`fo`)
   - Any except extracting pages (`fom`)
3. Enable copying of text, images and other content (`sc`)
4. Enable text access for screen reader devices for the visually impaired (`s`)

These flags can be combined.

Example: To grant permission that are equal to Acrobat's 7 "Printing Allowed: High Resolution" and "Enable copying of text, images and other content", set the flags `pdsc`.

```
-o ownerpassword -p pdsc input.pdf output.pdf
```

5.2 Fonts

Some features of the 3-Heights® PDF Security Service require fonts to be installed, e.g. for the creation of the visual appearance of digital signatures.

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

5.2.1 Font cache

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

In order to achieve optimal performance, make sure that the cache directory is writable for the 3-Heights® PDF Security Service. Otherwise, the font cache cannot be updated and the font directories have to be scanned on each program startup.

The font cache is created in the subdirectory `<CacheDirectory>/Installed Fonts` of the Cache directory.

5.3 Cryptographic provider

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

The 3-Heights® PDF Security Service can use various different cryptographic providers. The following list shows the provider that can be used for each type of signing certificate.

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

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

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

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

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

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

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

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

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

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

5.3.1 PKCS#11 provider

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

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

Configuration

Provider Option `-cp`

The provider configuration string has the following syntax:

`"‹PathToDll›;‹SlotId›;‹Pin›"`

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

- The CardOS API from Atos (Siemens) uses `siecap11.dll`
- The IBM 4758 cryptographic coprocessor uses `cryptoki.dll`
- Devices from Aladdin Ltd., use `etpkcs11.dll`
- For SafeNet Luna, HSM use `cryptoki.dll` an Windows or `libCryptoki2_64.so` on Linux/UNIX.
- For Securosys SA, Primus HSM or CloudsHSM, use `primusP11.dll`[1] on Windows and `libprimusP11.so`[1] on Linux.
- For Google Cloud HSM (Cloud KMS), use `libkmsp11.so`[2].

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

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

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

Example:

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

Interoperability support

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

- SafeNet Protect Server
- SafeNet Luna
- SafeNet Authentication Client
- IBM OpenCrypTokI
- CryptoVision
- Siemens CardOS
- Utimaco SafeGuard CryptoServer
- Securosys SA CloudsHSM[1]

Selecting a certificate for signing

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

1. **Certificate fingerprint**
   Option `-cfp`
   - SHA-1 fingerprint of the certificate. The fingerprint is 20 bytes long and can be specified in hexadecimal string representation, e.g. "b5 e4 5c 98 5a 7e 05 ff f4 c6 a3 45 13 48 0b c6 9d e4 5d f5". In Windows certificate store, this is called "Thumbprint", if "Thumbprint algorithm" is "sha1".
2. **Certificate issuer and serial number**
   Options `-ci` and `-cno`
   - Certificate issuer (e.g. "QV Schweiz CA"). In Windows certificate store, this is called "Issued By".

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

- Serial number of the certificate (hexadecimal string representation, e.g. "4c 05 58 fb"). This is a unique number assigned to the certificate by its issuer. In Windows certificate store, this is the field called "Serial number" in the certificate's "Details" tab.

3. **Certificate name and issuer (optional)**
   Options `-cn` and `-ci`
   - Common Name of the certificate (e.g. "PDF Tools AG"). In Windows certificate store, this is called "Issued To".
   - Optional: Certificate issuer (e.g. "QV Schweiz CA"). In Windows certificate store, this is called "Issued By".

Using PKCS#11 stores with missing issuer certificates

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

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

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

PKCS#11 devices that contain private keys only

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

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

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

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

5.3.2 Cryptographic suites

Message digest algorithm

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

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

SHA-256 (default)

SHA-384

SHA-512

RIPEMD-160

Signing algorithm

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

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

RSA_SSA_PSS This algorithm is sometimes also called RSA-PSS.

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

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

5.4 Windows Cryptographic Provider

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

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

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

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

CNG is available only if:

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

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

> **Certificate store:** It is important that the service has access to the certificate store that contains the signing certificate. For instance, with the default configuration, a service does not have access to the users' certificate store. See Store location for more information on the topic.

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

5.4.1 Configuration

Provider Option `-cp`

The provider configuration string has the following syntax:

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

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

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

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

Examples: Use the default provider with no PIN.

```
Provider = ""
```

Examples: "123456" being the PIN code.

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

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

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

Certificate store Option `-csn`

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

Store location Option `-csl`

Either of the following store locations:

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

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

To use a signing certificate from a user's certificate store, the service must be logged on as the user:

1. Open the "Services" window of the Control Panel.
2. Open the "Properties" dialog of the 3-Heights® PDF Security Service.
3. Go to the "Log On" tab and enter the credentials of the user.

Some cryptographic hardware (such as smart cards or USB tokens) require an interactive environment. As a result, the private key might not be available, unless the 3-Heights® PDF Security Service is run interactively.

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

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

5.4.2 Selecting a certificate for signing

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

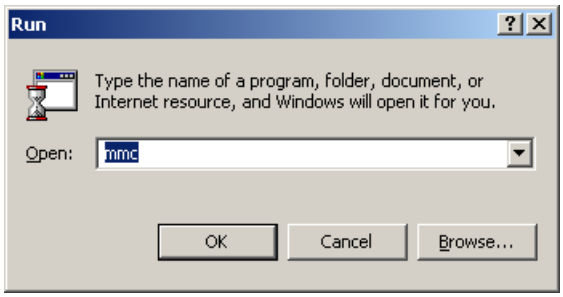
5.4.3 Certificates

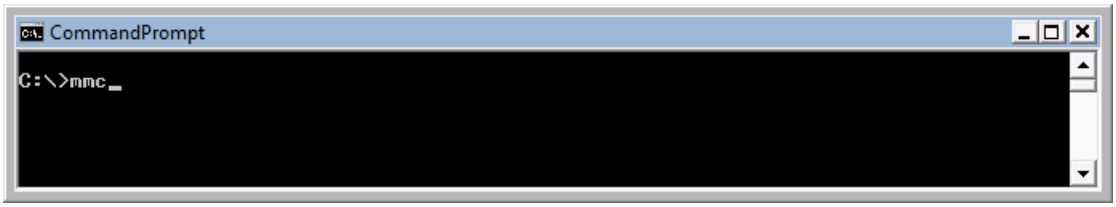
To sign a PDF document, a valid existing certificate name must be provided and its private key must be available.

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

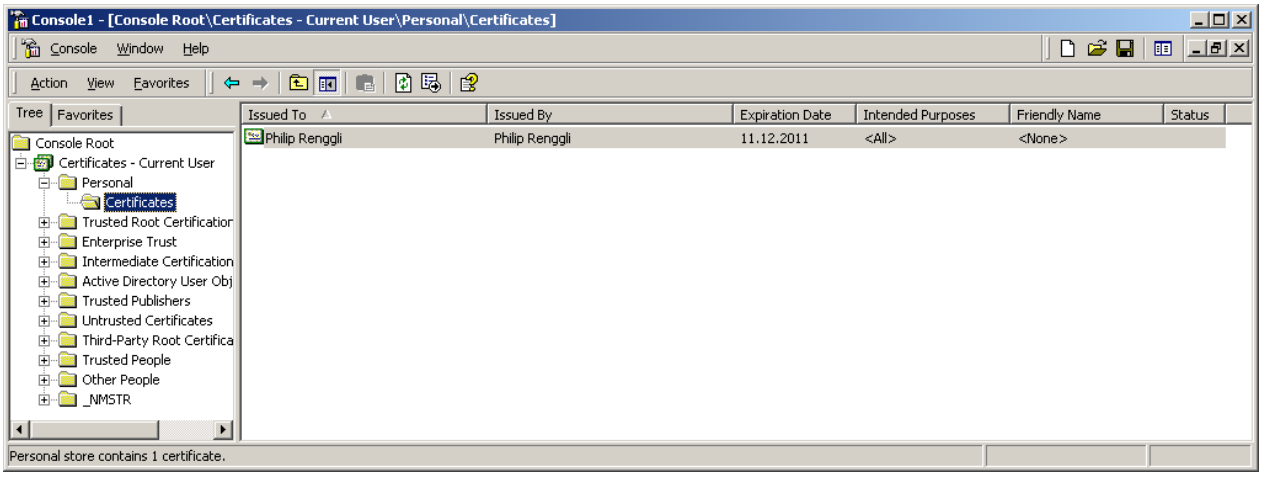
On the Windows operating system, certificates can be listed by the Microsoft Management Console (MMC), which is provided by Windows. To see the certificates available on the system, perform the following steps:

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

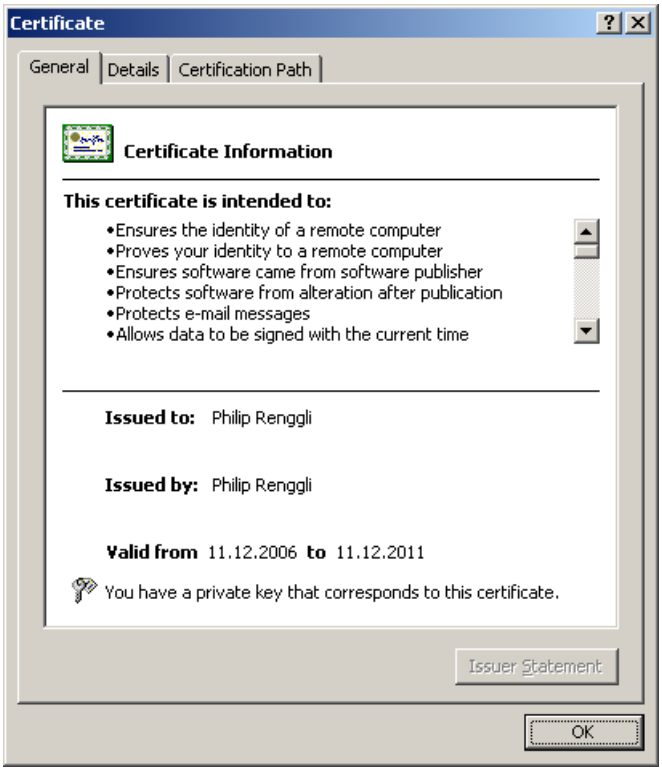




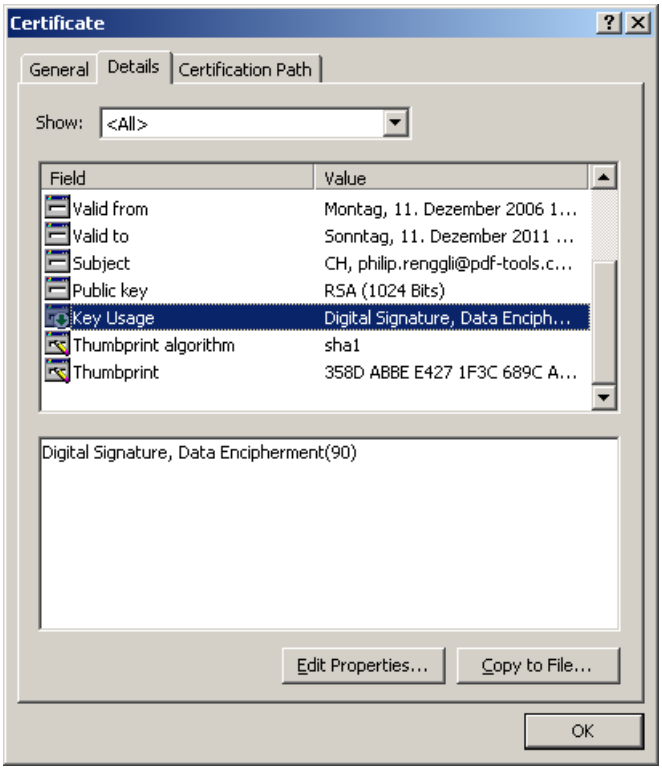
2. Under “File” →“Add/Remove Snap-in”.
3. Choose “Certificates” and click the “Add” button.
4. In the next window choose to manage certificates for “My user account”.
5. Click “Finish”.
6. The certificate must be listed under the root “Certificates - Current User”. For example, as shown in the screenshot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



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

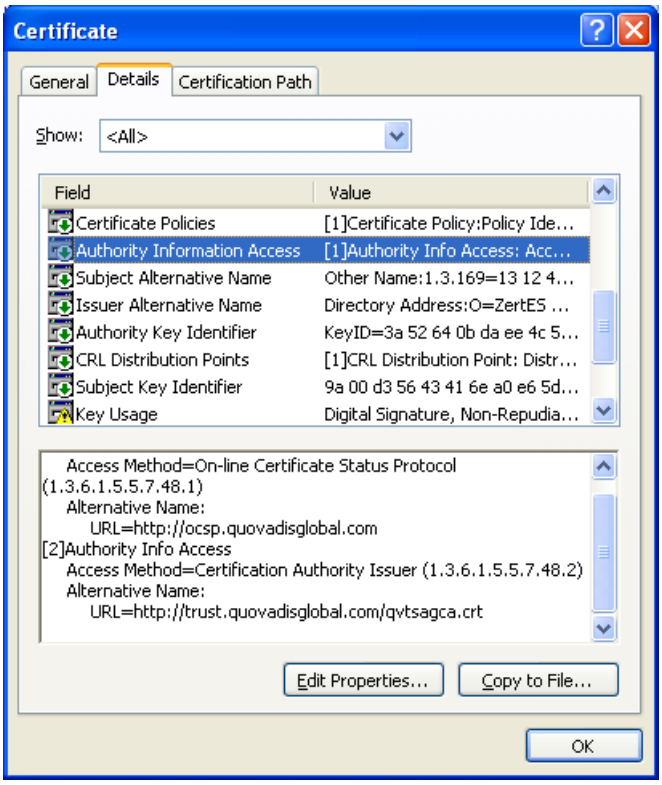


8. In the Details tab of the certificate, there is a field named “Key Usage”. This field must contain the value “Digital Signature”. Additional values are optional. See the figure below.
   You must have the private key that corresponds to this certificate.



5.4.4 Qualified certificates

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



5.4.5 Cryptographic suites

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

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

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

5.5 myBica Digital Signing Service

Provider Option `-cp`

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

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

There are two types of properties:

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

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

Signature configuration The signature can be customized using standard options of the 3-Heights® PDF Security Service.

Description	Required	Value	Setting
Common Name	Required	The name of the signer must be set[3].	Option `-cn`.
Timestamp	optional	Use the value `urn:ietf:rfc:3161` to embed a timestamp.	Option `-tsu`
Signature Format	Optional	To set the signature format	Option `-st`. Must be `adbe.pkcs7.detached`
Revocation Info	Recommended	To embed OCSP responses or CRL.	Option `-co`[4]

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

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

5.6 QuoVadis sealsign

Provider Option `-cp`

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

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

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

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

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

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

Signature configuration The signature can be customized using standard options.

Description	Required	Value	Setting
Common Name	Required	The name of the signer must be set[5].	Option `-cn`.
Timestamp	-	Not available.	
Revocation Info	Recommended	To embed OCSP responses or CRL.	Option `-co`[6]
Visual Appearance	Optional	See Creating a visual appearance of a signature.	

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

5.7 Swisscom All-in Signing Service

5.7.1 General properties

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

Description	Required	Value	Setting
Common Name	Required	Name of the signer[7].	Option `-cn`
Provider	Required	The service endpoint URL of the REST service. Example: `https://ais.swisscom.com/AIS-Server/rs/v1.0/sign`	Option `-cp`
Timestamp	optional	Use the value `urn:ietf:rfc:3161` to embed a timestamp.	Option `-tsu`
Signature Format	Optional	To set the signature format	Option `-st`. Supported values are `adbe.pkcs7.detached`, `ETSI.CAdES.detached`, `ETSI.RFC3161`[8].

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

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

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

5.7.2 Provider session properties

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

There are two types of properties:

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

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

[7] This parameter is not used for certificate selection, but for the signature appearance and description in the PDF only.
[8] `ETSI.RFC3161` is automatically set when signing with `-dts`

5.7.3 On-demand certificates

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

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

5.7.4 Step-up authorization using Mobile-ID

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

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

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

5.8 GlobalSign Digital Signing Service

Provider Option `-cp`

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

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

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

There are two types of properties:

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

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

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

Signature configuration The signature can be customized using standard options of the 3-Heights® PDF Security Service.

Description	Required	Value	Setting
Common Name	Required	The name of the signer must be set[9].	Option `-cn`.
Timestamp	recommended	Use the value `urn:ietf:rfc:3161` to embed a timestamp.	Option `-tsu`
Signature Format	Optional	To set the signature format	Option `-st`. Supported values are `adbe.pkcs7.detached`, `ETSI.CAdES.detached`, `ETSI.RFC3161`[8].
Revocation Info	Recommended	To embed OCSP responses or CRL.	Option `-co`[10]

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

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

Creating the SSL client certificate

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

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

Getting the SSL server certificate

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

1. Get the server's SSL certificate:

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

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

Advice on using the service

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

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

%

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

6 Creating digital signatures

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

6.1 Creating a PAdES signature

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

PAdES-B-B A digital signature.

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

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

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

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

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

The Compatibility of PAdES signature levels shows how the signature levels described above and as created by the 3-Heights® PDF Security Service conform with other standards.

Compatibility of PAdES signature levels

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

Requirements

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

Requirements

Level	Signing Certificate	Timestamp	Product
PAdES-B-B	any	no	3-Heights® PDF Security Service
PAdES-B-T	any	required	3-Heights® PDF Security Service
PAdES-B-LT	advanced or qualified certificate	required	3-Heights® PDF Security Service
PAdES-B-LTA	advanced or qualified certificate	required	3-Heights® PDF Security Service

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

Note on encryption and linearization: Because signature levels PAdES-B-LT and PAdES-B-LTA must be created in a two-step process, the files cannot be linearized and encryption parameters cannot be changed. When creating signature levels PAdES-B-B or PAdES-B-T that may later be augmented, linearization should not be used and all encryption parameters (user password, owner password, permission flags, and encryption algorithm) must be the same for both steps.

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

6.1.1 Create a PAdES-B-B signature

Input document Any PDF document.

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

```
-cp "myPKCS11.dll;0;pin" -cn "..." -st "ETSI.CAdES.detached" -co
```

6.1.2 Create a PAdES-B-T signature

Input document Any PDF document.

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

```
-cp "myPKCS11.dll;0;pin" -cn "..." -st "ETSI.CAdES.detached" -co

-tsu "http://server.mydomain.com/tsa"
```

6.1.3 Create a PAdES-B-LT signature

Input document A PDF document with a PAdES-B-T signature created using an advanced or qualified certificate.

Cryptographic provider Any cryptographic provider.

```
-cp "myPKCS11.dll;0;pin" -dss
```

6.1.4 Create a PAdES-B-LTA signature or extend longevity of a signature

Input document

- A PDF document with a PAdES-B-T signature created using an advanced or qualified certificate, or
- a PAdES-B-LTA signature whose longevity should be extended.

Cryptographic provider Any cryptographic provider whose trust store contains all certificates requried for `-dss`.

```
-cp "myPKCS11.dll;0;pin" -dss -dts -tsu "http://server.mydomain.com/tsa"
```

6.2 Applying multiple signatures

Multiple signatures can be applied to a PDF document. One signature must be applied at the time. Signing a signed file does not break existing signatures, because the 3-Heights® PDF Security Service uses an incremental update.

Signing a linearized file renders the linearization information unusable. Therefore, it is recommended to not linearize files that need to be signed multiple times.

6.3 Creating a timestamp signature

For a timestamp signature, no local signing certificate is required. Instead the timestamp signature requested from the timestamp authority (TSA) is embedded into the document. Nonetheless, a Cryptographic provider that supports timestamp signatures is required.

Example: Create a timestamp signature using the option `-dts`.

```
-tsu "http://server.mydomain.com/tsa" -dts
```

6.4 Creating a visual appearance of a signature

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

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

Different properties of the visual appearance can be specified.

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

Color See options `-acf` and `-acs`.

Line width The line width of the background rectangle, see option `-al`.

Text Two text fragments can be set using two different fonts, font sizes, and colors see options `-at1`, `-at2`, `-atc1`, `-atc2`, `-af1`, `-af2`, `-afs1`, and `-afs2`.

Background image See options `-abg`.

6.5 Miscellaneous

6.5.1 Caching of CRLs, OCSP, and timestamp reponses

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

The following caches are stored automatically by the 3-Heights® PDF Security Service at the indicated locations within the Cache directory:

Certificates	`<CacheDirectory>/Certificates/hash.cer`
CRL	`<CacheDirectory>/CLRs/server.der`
OCSP responses	`<CacheDirectory>/OCSP Responses/server-hash.der`
Service data	`<CacheDirectory>/Signature Sizes/hash.bin`
Timestamp responses[11]	`<CacheDirectory>/Time Stamps/server.der`

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

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

6.5.2 Using a proxy

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

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

Where:

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

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

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

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

6.5.3 Configuring a proxy server and firewall

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

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

OCSP

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

Timestamp

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

Signature services

- Signature service-specific MIME types.

6.5.4 Setting the signature build properties

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

7 Validating digital signatures

7.1 Validating a qualified electronic signature

There are basically three items that need to be validated:

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

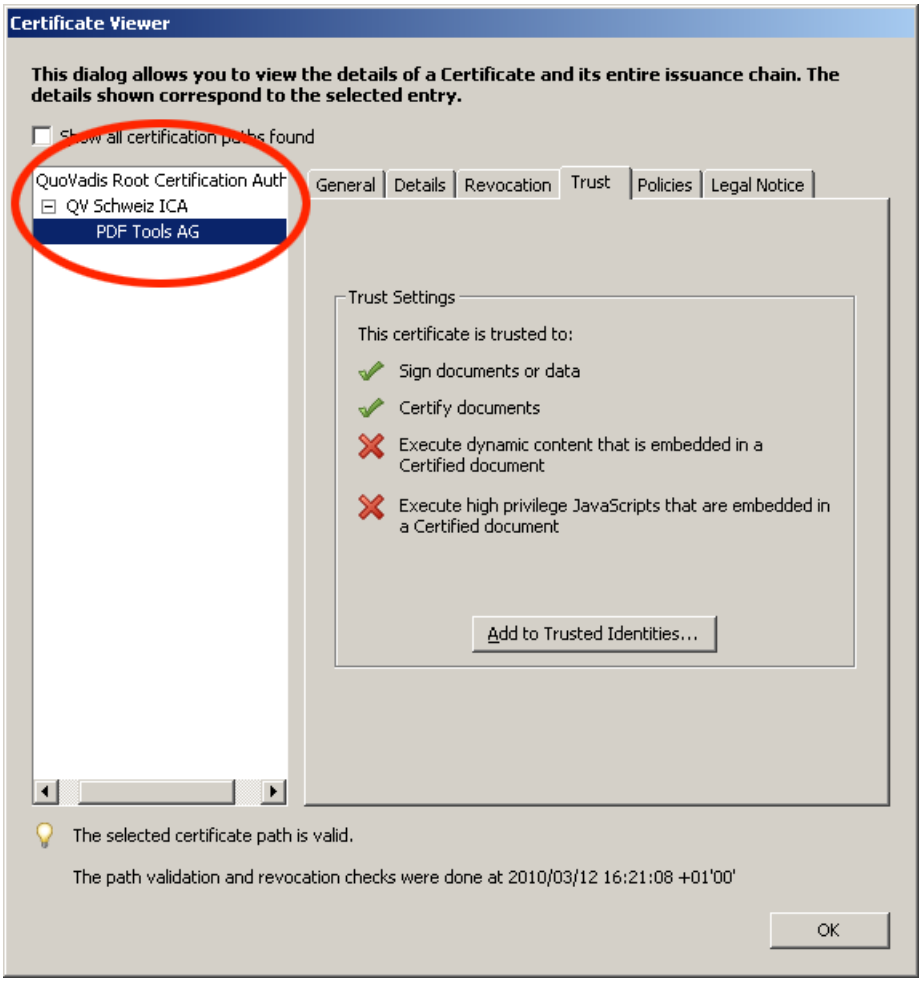
Validation can be done in different ways, e.g. Adobe Acrobat, from which the screenshots below are taken.

7.1.1 Trust chain

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

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

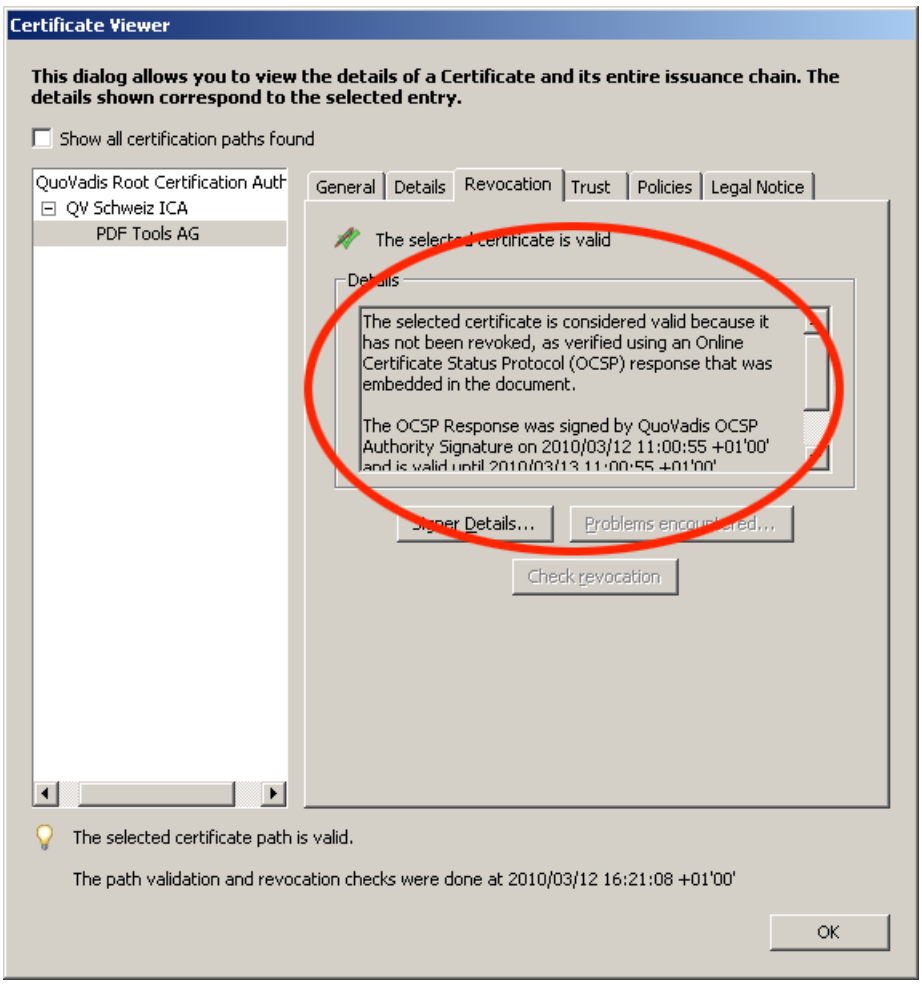
After that you can validate the signature, e.g. by open the PDF document in Adobe Acrobat, right-click the signature and select “Validate”, then select “Properties”, and select the tab “Trust”. There the certificate should be trusted to “sign documents or data”.



7.1.2 Revocation information

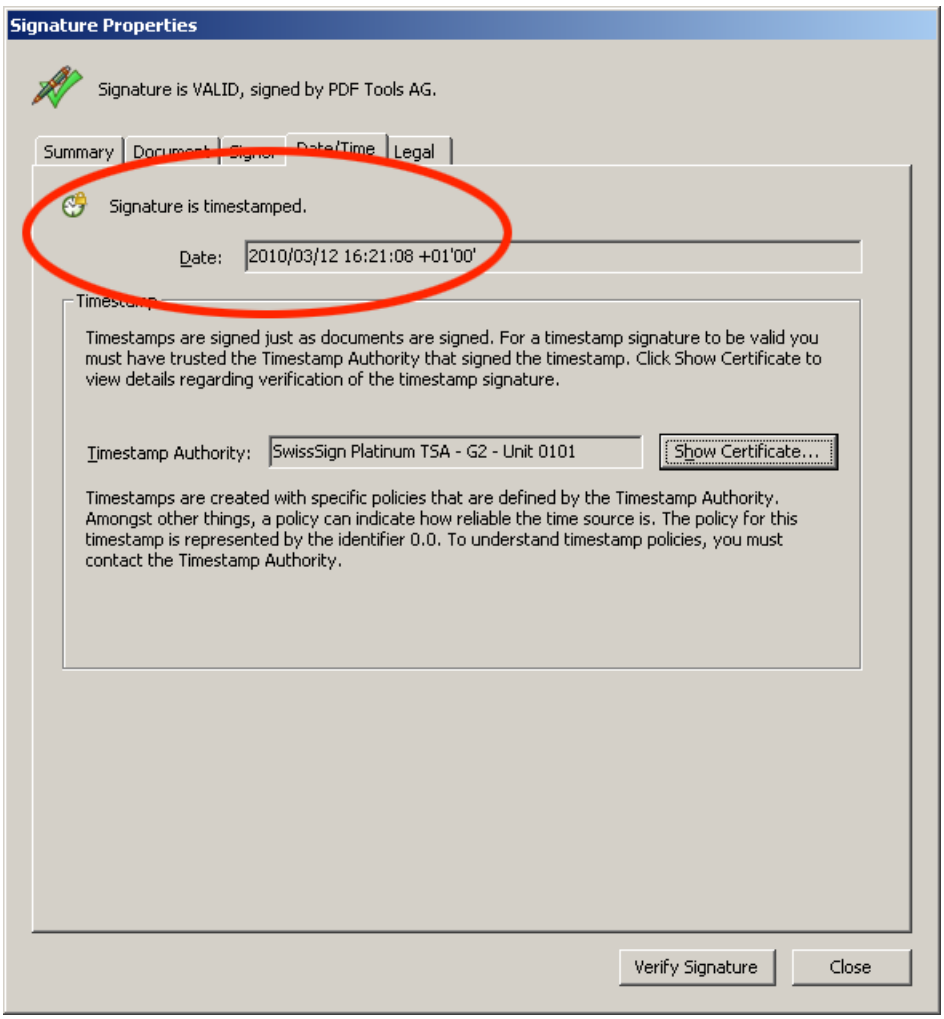
An OCSP response or CRL must be available. This is shown in the tab “Revocation”. The details should mention that “the certificate is considered *valid*”.

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



7.1.3 Timestamp

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



7.2 Validating a PAdES LTV signature

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

The following must be checked:

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

7.2.1 Trust chain

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

7.2.2 Revocation information

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

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

7.2.3 Timestamp

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

7.2.4 LTV expiration date

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

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

7.2.5 Other PAdES requirements

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

8 Interface reference

8.1 Service control commands

These options are used to control the service. The create and delete functions require administrator rights. The start and stop functions require operator rights.

8.1.1 -a Pause service

Pause service `-a`

This option pauses the service.

```
pdfsecuresvr -a
```

8.1.2 -c Create service

Create service `-c`

The 3-Heights® PDF Security Service is created using the `-c` option.

```
pdfsecuresvr -c
```

> **Important:** It is essential that `pdfsecuresvr.exe` is on a non-mapped drive.

8.1.3 -d Delete service

Delete service `-d`

The 3-Heights® PDF Security Service can be deleted with the `-d` option. It is best used after the service has already been stopped.

```
pdfsecuresvr -d
```

8.1.4 -i List the usage

List the usage `-i`

The `-i` option lists the current version and date of the service along with all available settings.

```
pdfsecuresvr -i
```

8.1.5 -o Continue service

Continue service `-o`

This option resumes the service.

```
pdfsecuresvr -o
```

8.1.6 -q Query current status of service

Query current status of service `-q`

This option returns the current status of the service.

```
pdfsecuresvr -q
The service starts automatically during system startup.
The service is stopped.
[pdfsecuresvr] QueryService: The operation completed successfully.
```

8.1.7 -s Start service

Start service `-s`

Once created, the 3-Heights® PDF Security Service can be started with the `-s` option.

```
pdfsecuresvr -s
```

8.1.8 -t Stop service

Stop service `-t`

To stop the service, use the `-t` option.

```
pdfsecuresvr -t
```

If “stop” is called while the service is “running”, the current job (all pages) will be finished, after that the service is stopped.

If the service was “paused” before calling “stop”, the current page will be finished processing. After that page, the job is aborted.

8.1.9 -x Run as executable

Run as executable `-x`

With this option, the PDF Security Service runs as an executable instead of as a Windows Service. It provides the same functionality as long as the executable is "running".

```
pdfsecuresvr -x
```

8.2 Configuration options

8.2.1 PdfSecureSvr.ini configuration file

The `PdfSecureSvr.ini` configuration file defines the setting for the watched folders. It is read on starting the service.

`[PdfSecureSvr]`	Required	
`AutoDelete=...`	Optional	`true` or `false`
`AutoDeleteAll=...`	Optional	`true` or `false`
`LogPath=...`	Optional	Either a path like `C:\mypath\log` or the keyword `EventLog`
`PollingInterval=...`	Optional	Value in milliseconds, default `1000`
`JobPrefix=...`	Optional	`true` or `false`
`LogLevel=...`	Optional	`0` or `1`
`Threads=n`	Required	The number of threads
`Thread1=-w ...`	Required	Options for the first thread
`Thread2=-w ...`		Options for the second thread
`Threadn=...`		There must be exactly as many threads as defined in `Threads=n`.

Autodelete of successfully processed files

When a print job succeeded, the PDF document will be moved from the `Jobs` folder to the `Succeeded` folder. To automatically delete the successfully printed files, the value `AutoDelete` can be set to `true` on the `PdfSecureSvr.ini` file. When set to `false`, the files remain in the folder `Succeeded`. Documents which fail to print are not deleted.

```
[PdfSecureSvr]
AutoDelete=true
```

To delete failed documents as well, use the following setting:

```
[PdfSecureSvr]
AutoDeleteAll=true
```

Job number prefix

Every time a document is copied from the watched folder to the `Jobs` subfolder, it is renamed by adding a 21 character prefix containing a timestamp of the form `Job-‹8 digits›-‹8 digits›_`. For example

`Job-01C61DD4-E72E1BCE_`

Note: When `JobPrefix=false` then the incoming documents must have unique file names.

The job number prefix ensures that several documents with the same name can correctly be processed. Adding the prefix can be prevented with the following line in the configuration file:

```
[PdfSecureSvr]
JobPrefix=false
```

Logpath

Log messages created by the service are by default written to the `log` subdirectory. To alter the directory, add a line similar as shown below to the configuration file:

```
[PdfSecureSvr]
LogPath=C:\path\log
```

Messages created by the service can be added to the system's application event log instead of written to a log file. This is achieved by adding the following line to the configuration file:

```
[PdfSecureSvr]
LogPath=EventLog
```

The system's application log event then logs messages similar as shown below:

- CreateService: The operation completed successfully.
- StartService: The operation completed successfully.

Note: The messages are only fully accessible while the service is created.

Otherwise, a message as shown below is displayed:

- The description for Event ID (1) in Source (`pdfsecuresvr`) cannot be found. The local computer may not have the necessary registry information or message DLL files to display messages from a remote computer. The following information is part of the event: DeleteService: The operation completed successfully.

Polling interval

The polling interval defines the time in milliseconds that the polling thread pauses between two polls. The time passing until the same watched folder is polled again (maximum pick-up time) is: The value of `PollingInterval` plus the actual time it takes to poll all watched folders. The higher the polling interval, the lower the network traffic, and the longer it takes until documents are picked up.

Suggested values for the polling intervals are 1000 to 10000 milliseconds.

```
[PdfSecureSvr]
```

```
PollingInterval=5000
```

8.2.2 -w Set the watched folder

Set the watched folder `-w`

Use the `-w` option to define the path of the watched folder. This path should not contain mapped drives, since other users (such as LocalSystem) do not recognize them. This parameter must always be the first parameter of a thread.

```
-w C:\output\watchedfolder
```

> **Note:** The service supports path lengths including file name of up to 258 characters. This includes the 21 characters of the job ticket. If a file name exceeds this value, its file name is truncated at the end of the file name and before the file extension. Therefore, watched folder names should be kept reasonably short.

8.2.3 -wd Set the drop in folder

Set the drop in folder `-wd`

By default, the drop in folder is equal to the folder defined as watched folder using the `-w` option. If the input files should be taken from a different folder, this can be configured using `-wd`. All folders created by service, including the output folder, are at the directory defined by `-w`.

```
-wd C:\SomePath\DropIn
```

8.2.4 -wfs Process only files with certain extensions

Process only files with certain extensions `-wfs ‹exts›`

By default, the service tries to process all files dropped into the drop directory, regardless of the extension. With this option, the processing can be restricted to a set of known file extensions.

Example: Restrict the processing to PDF and FDF files.

```
-wfs .pdf.fdf
```

8.2.5 -wfi Ignore files with certain extensions

Ignore files with certain extensions `-wfi ‹exts›`

By default, the service tries to process all files dropped into the drop-in folder, regardless of the extension. With this option, files with certain file extensions can be ignored.

Example: Ignore temporary files.

```
-wfi .temp.tmp
```

8.3 Encryption

8.3.1 -fe Force encryption

Force encryption `-fe`

File encryption is not allowed by the PDF/A standard. Therefore, 3-Heights® PDF Security Service aborts and returns an error when encryption is configured and an input file is PDF/A. Use this option to enable encryption of PDF/A conforming files. The conformance of the output file is downgraded to PDF.

8.3.2 -fm Set stream crypt filter

Set stream crypt filter `-fm ‹name›`

Set the stream crypt filter. Supported values for `‹name›` are the following strings: None, V2, RC4, AESV2, and AESV3. Certain PDF viewers require the stream crypt filter to be equal to the string crypt filter, e.g. both must be RC4 or AES. Other tools such as the 3-Heights® PDF Tools do not have this limitation. Setting an empty string selects the default filter.

Example: Set the stream crypt filter and the string crypt file to AESV2

```
-o owner -fm AESV2 -fr AESV2
```

8.3.3 -fr Set string crypt filter

Set string crypt filter `-fr ‹name›`

Set the string crypt filter. Supported values are the following strings: None, V2, RC4, AESV2, and AESV3. Setting an empty string selects the default filter.

Supported values for the crypt filter are described in the following table:

Description of supported values for setting the string crypt filter

Values	Description
None	The application does not decrypt data.
V2 or RC4	(PDF 1.2, default) The application asks the security handler for the encryption key and implicitly decrypts data using the RC4 algorithm.

Description of supported values for setting the string crypt filter

AESV2	(PDF 1.6) The application asks the security handler for the encryption key and implicitly decrypts data with using the AES-V2 128 bit algorithm.
AESV3	(PDF 1.7) The application asks the security handler for the encryption key and implicitly decrypts data with using the AES-V3 256 bit algorithm.

8.3.4 -k Set the length of the encryption key

Set the length of the encryption key `-k ‹key-length›`

The key length is a determining factor of the strength of the encrypting algorithm and the amount of time to break the cryptographic system. For RC4, the key length can be any value from 40 to 128 that is a multiple of 8. For AESV2, the key length is automatically set to 128; for AESV3, to 256.

Notes:

- Certain PDF viewers only support 40 and 128 bit encryption. Other tools such as the 3-Heights® tools also support other encryption key lengths.
- 256 bit encryption requires Acrobat 9 or later.
- If the selected permission flags require a minimum key length, the key length is automatically adjusted (e.g. to 128 bits).

8.3.5 -o Owner password

Owner password `-o ‹owner›`

The owner password is required to change the security settings of the document. To apply permission flags, an owner password must be set. Permission flags are set with the `-p` switch.

Example: Encrypt a document and set the owner password to `‹owner›`.

```
-o owner
```

8.3.6 -p Permission flags

Permission flags `-p ‹flags›`

This option sets the permission flags. It is only usable when producing encrypted documents. In other words, at least an owner password must be set with `-o`, and additionally a user password can be set with `-u`. When omitting the `-p` option, then all permissions are granted. The permissions that can be granted are listed below.

Permission flags

Flag	Description
p	Allow printing (low resolution)
m	Allow change of the document
c	Allow content copying or extraction
o	Allow commenting
f	Allow filling of form fields
s	Allow content extraction for accessibility
a	Allow document assembly
d	Allow high quality printing
i	Set the same permissions as in the input file
0	Allow nothing (no permissions are granted)

The actual ‹flags› given to this option is a string that contains one or several of the permission flags above.

> **Note:** The i and 0 values cannot be combined with any other permission flags.

Example: The following command sets the owner password to “owner” and the permission flags to “allow printing in low resolution” and “allow form filling”.

```
-o owner -p pf
```

Example: “High quality printing” requires the standard printing flag to be set too.

```
-o owner -p pd
```

Example: Create a document with the same permission settings as present in the input document.

```
-o owner -p i
```

For further information about the permission flags, see PDF Reference 1.7 Section 3.5.2.

8.3.7 -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:

```
-pw mypassword
```

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

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

8.3.8 -u User password

User password `-u ‹user›`

Set the user password of the document. If a document that has a user password is opened for any purpose (such as viewing, printing, editing), either the user or the owner password must be provided.

A user who knows the user password is able to open and read the document. A user who knows the owner password is able to open, read, and modify (e.g. change passwords) the document. A PDF document can have none, either, or both passwords.

Example: Encrypt a document with a user and an owner password

```
-u userpassword -o ownerpassword
```

8.4 Digital signatures

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

8.4.1 -abg Signature background image

Signature background image `-abg ‹image›`

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

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

8.4.2 -acf Signature fill color

Signature fill color `-acf ‹rgb›`

This is the color of the signature's background as in RGB value. The default is `16761024` (red = 192, green = 192, blue = 255). To not set a color, i.e. keep the rectangle transparent, set it to `-1`.

Color examples: Color values are

`color = ‹red› + ‹green›×256 + ‹blue›×256×256`,

where ‹red›, ‹green› and ‹blue› assume values from `0` to `255`.

	Red	`255,0,0`	`255`
	Green	`0,255,0`	`65'280`
	Blue	`0,0,255`	`16'711'680`
	Cyan	`0,255,255`	`16'776'960`
	Magenta	`255,0,255`	`16'711'935`
	Yellow	`255,255,0`	`65'535`
	Black	`0,0,0`	`0`
	Gray	`128,128,128`	`8'421'504`
	White	`255,255,255`	`16'777'215`

8.4.3 `-acs` Signature stroke color

Signature stroke color `-acs ‹rgb›`

This is the color of the signature's border line as RGB value. The default is `8405056` (red = 64, green = 64, blue = 128). To avoid setting a color, i.e. keep it transparent, set it to `-1`.

8.4.4 `-af1` Signature font name 1

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

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

8.4.5 `-af2` Signature font name 2

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

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

8.4.6 -afs1 Signature font size 1

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

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

8.4.7 -afs2 Signature font size 2

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

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

8.4.8 -al Signature line width

Signature line width `-al ‹width›`

This is the thickness of the line surrounding the visual signature in points.

8.4.9 -ap Signature page number

Signature page number `-ap ‹page›`

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

8.4.10 -ar Signature annotation rectangle

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

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

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

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

```
-cn "..." -ar 10 770 200 60
```

8.4.11 -at1 Signature text 1

Signature text 1 `-at1 ‹text›`

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

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

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

8.4.12 -at2 Signature text 2

Signature text 2 `-at2 ‹text›`

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

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

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

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

8.4.13 -atc1 Signature text color 1

Signature text color 1 `-atc1 ‹rgb›`

This option sets the color of the upper text, i.e. the text that is set by `-at1`.

The default is black: `0` (red = `0`, green = `0`, blue = `0`). See `-acf` for more examples of color values.

8.4.14 -atc2 Signature text color 2

Signature text color 2 `-atc2 ‹rgb›`

This option sets the color of the lower text, i.e. the text that is set by `-at2`.

The default is black: `0` (red = `0`, green = `0`, blue = `0`). See `-acf` for more examples of color values.

8.4.15 -cci Signer contact info

Signer contact info `-cci ‹info›`

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

8.4.16 -cfp Certificate fingerprint

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

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

8.4.17 -ci Certificate issuer

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

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

8.4.18 -cn Certificate name (Subject)

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

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

See Digital signatures to learn more about digital signatures in general and how to sign documents with the 3-Heights® PDF Security Service.

Example: Sign the document

```
-cn "Philip Renggli"
```

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

8.4.19 -cno Certificate serial number

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

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

8.4.20 -co Do not embed revocation information

Do not embed revocation information `-co`

This switch inhibits the embedding of revocation information such as online certificate status response (OCSP - RFC 2560) and certificate revocation lists (CRL - RFC 3280). Revocation information is either an OCSP response or a CRL, which is provided by a validation service at the time of signing and acts as proof that at the time of signing the certificate is valid. This is useful because even when the certificates expires or is revoked at a later time, the signature in the signed document remains valid.

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

This option is not supported by all cryptographic providers and never for document timestamp signatures. For these cases, `-dss` must be used.

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

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

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

`application/ocsp-request`

`application/ocsp-request`

8.4.21 -cp Cryptographic provider

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

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

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

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

  Examples: `"123456"` being the PIN code

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

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

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

Example:

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

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

8.4.22 -cpf Cryptographic session property (file)

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

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

8.4.23 -cps Cryptographic session property (string)

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

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

8.4.24 -cr Signature reason

Signature reason `-cr ‹reason›`

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

Example: Sign the document an add a reason text.

```
-cn "Philip Renggli" -cr "Review and approval" -ar 10 10 200 50
```

The signature of the resulting output looks as shown below:

8.4.25 -csl Certificate store location

Certificate store location `-csl ‹location›`

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

0 Local Machine.

1 Current User (default).

For more information, see Windows Cryptographic Provider.

8.4.26 -csn Certificate store name

Certificate store name `-csn ‹store›`

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

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

```
-cn "..." -csn ROOT -csl 0
```

8.4.27 -dap Document access permissions for DocMDP signature

Document access permissions for DocMDP signature `-dap ‹p›`

This option controls the type of permitted modifications to a certified document. Valid values are:

1 No changes to the document are permitted; any change to the document invalidates the signature (default).

2 Permitted changes are filling in forms, instantiating page templates, and signing; other changes invalidate the signature.

3 Permitted changes are the same as for 2, as well as annotation creation, deletion, and modification; other changes invalidate the signature.

8.4.28 -dss Add signature validation information to the document's DSS

Add signature validation information to the document's DSS `-dss`
License feature: `Signature`

Add signature validation information to the document security store (DSS). This information includes:

1. All certificates of the signing certificate's trust chain, unless they are already embedded into the signature.
2. Revocation data (OCSP or CRL) for all certificates that support revocation information.

Validation information for embedded timestamp tokens is added as well.

This requires a Cryptographic provider, which has been specified using `-cp`. All types of cryptographic providers support this method. However, this method fails when using a provider whose certificate store is missing a required certificate. Because providers of digital signature services do not have a certificate store, it is recommended that you use either the PKCS#11 or the Windows Cryptographic provider.

This method can be used to create signatures with long-term validation material or to enlarge the longevity of existing signatures. See section Creating a PAdES signature for more information.

Note: This method does not validate the signatures, but only downloads the information required.

Note: Adding validation information for expired certificates is not possible. Therefore, it is crucial to enlarge the longevity of signatures before they expire.

8.4.29 `-dts` Create a timestamp signature

Create a timestamp signature `-dts`
License feature: `Signature`

Add a document-level timestamp. No appearance is created. The following signature option must be set: `-tsu`. The following signature options may be set: `-cp`, `-tsc`, `-wpu`, `-wpc`.

8.4.30 `-fs` Force signature

Force signature `-fs`

Force signature allows DocMDP (PDF 1.6) and timestamp signatures (PDF 2.0) on PDF/A-1 documents. The output file's version is upgraded and PDF/A conformance is removed. So the output file contains the signature, but is not PDF/A-1 anymore.

Applying a DocMDP or timestamp signature breaks PDF/A-1 conformance. Therefore, the default behavior is to abort the operation with an error.

8.4.31 `-mdp` Create a DocMDP signature

Create a DocMDP signature `-mdp`
License feature: `Signature`

This option creates a DocMDP (document modification detection and prevention) signature instead of a document signature. The DocMDP signature is also referred to as "certify a document".

Note: This version can create visible DocMDP signatures. In order to create an invisible signature, set the signature's rectangle as follows: `-ar 0 0 0 0`.

8.4.32 `-nc` Disable cache for CRL and OCSP

Disable cache for CRL and OCSP `-nc`

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

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

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

8.4.33 -nd Disable the use of DSS when signing documents

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

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

8.4.34 -p2f Replace placeholder image with signature field

Replace placeholder image with signature field `-p2f`
License feature: `Signature`

This option enables the replacement of special placeholder images with signature fields that can later be signed (e.g. with Adobe Acrobat or the 3-Heights® PDF Security Service). This function is used to automatically place signature fields under the control of the creator program.

The following image must be used as placeholder: signature-placeholder.png.

8.4.35 -st Set signature subfilter

Set signature subfilter `-st ‹subfilter›`

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

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

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

8.4.36 -tsc Timestamp credentials

Timestamp credentials `-tsc ‹cred›`

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

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

```
-cn "..." -tsu http://mytimestamp.com -tsc username:password
```

8.4.37 -tsu Timestamp URL

Timestamp URL `-tsu ‹url›`

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

`application/timestamp-query`

`application/timestamp-reply`

8.4.38 `-wpc` Web proxy server credentials

Web proxy server credentials `-wpc ‹cred›`

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

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

```
-wpu "http://proxy.example.org" -wpc user:password
```

8.4.39 `-wpu` Web proxy server URL

Web proxy server URL `-wpu ‹url›`

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

8.5 General switches

8.5.1 `-id` Set value in the document information dictionary

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

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

Example: Overwrite the default producer:

```
-id Producer "MyProgram 1.2"
```

8.5.2 `-ow` Optimize for the web

Optimize for the web `-ow`

Note: This option has no effect when combined with -owa.

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

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

The 3-Heights® PDF Security Service does not support linearization of PDF 2.0 documents. For such documents, processing fails. In order to automatically disable linearization for PDF 2.0 use -owa.

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

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

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

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

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

8.5.3 -owa Optimize for the Web automatically

Optimize for the Web automatically -owa

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

Automatically decide whether to linearize the PDF output file for fast web access.

Applying linearization can lead to a large increase in file size for certain documents. Enabling this option lets the 3-Heights® PDF Security Service automatically apply linearization or refrain from doing so based on the estimated file size increase.

With this option enabled, PDF 2.0 documents are automatically excluded from linearization.

See also -ow for more information for linearized PDFs.

Note: When -owa is given, then the -ow option has no effect.

8.6 Frequent error source

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

8.7 Tracing

The 3-Heights® PDF Security Service contains tracing functionality that logs runtime information to a file. No confidential data, such as the content of processed files or passwords, are traced. The tracing functionality is designed to provide useful information to Pdftools's support team for support requests. Tracing is not active by default and can be activated by the customer under the guidance of the support team. Nonetheless, activating tracing and sharing the information is optional and there is no obligation to do so.

9 Version history

9.1 Changes in versions 6.19–6.27

- **New** support for Elliptic Curve DSA (ECDSA) signature algorithms in the PKCS#11 Provider and Windows Cryptographic Provider.
- **New** support for PKCS#11 devices that contain private keys only.
- **Update** license agreement to version 2.9
- **Improved** auto font size feature for text stamping feature.

9.2 Changes in versions 6.13–6.18

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

9.3 Changes in versions 6.1–6.12

- Digital Signatures
  - Swisscom All-in Signing Service
    - **New** support for accounts (`Identity`) based on Swisscom CA 4 Certificate Authorities.
    - **New** support to create PAdES signatures (format `ETSI.CAdES.detached`).
  - **Improved** embedding of revocation information (OCSP, CRL, and trust chain) to always use the document security store (DSS)[12].
  - **Changed** the creation of signatures of format `ETSI.CAdES.detached` to include revocation information if `-co` is not used and if supported by the cryptographic provider.
  - **Improved** support for new version of the GlobalSign Digital Signing Service. The service endpoint should be updated to `https://emea.api.dss.globalsign.com:8443/v2`.
- **Improved** search algorithm for installed fonts: User fonts under Windows are now also taken into account.

9.4 Changes in version 5

- Digital Signatures
  - **New** support to get CRLs using HTTPS and via HTTP redirection.
- **New** additional supported operating system: Windows Server 2019.
- **New** options `-atc1` and `-atc2` to set the color of the signature appearance's text.

9.5 Changes in version 4.12

- **Introduced** license features `Signature` and `Stamping`.
- Digital Signatures

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

    - **New** support to sign OCSP requests, if required by the OCSP service.
    - **New** support for OCSP requests over HTTPS.
    - **Changed** acceptance criteria for OCSP responses that specify no validity interval (missing nextUpdate field, which is uncommon). Previously a validity interval of 24 hours has been used, now 5 minutes due to Adobe® Acrobat® compatibility.
- **New** support for encryption according to PDF 2.0 (revision 6, replaces deprecated revision 5).
- **Improved** reading and recovery of corrupt TIFF images.
- **New** HTTP proxy setting in the GUI license manager.
- **New** option `-owa` to automatically choose whether to linearize the output document or not.

9.6 Changes in version 4.11

- Digital Signatures
    - **New** support to create Document TimeStamp signatures using Swisscom All-in Signing Service.
    - **New** ability to sign documents that are larger than 2GB (64-bit version only).
- Stamping
    - **New** default compression Flate for PNG images.
- **New** support for reading and writing PDF 2.0 documents.
- **New** support for the creation of output files larger than 10GB (not PDF/A-1).
- **Improved** search in installed font collection to also find fonts by other names than TrueType or PostScript names.
- **New** treatment of the `DocumentID`. In contrast to the `InstanceID` the `DocumentID` of the output document is inherited from the input document.
- **Changed** option `-p`: Added a new value `i` to adopt the encryption parameters from the input document.
- **New** directory `Logs` in which an error log file is generated for each failed operation.

9.7 Changes in version 4.10

- Digital signatures
    - **New** support for the new European PAdES norm (ETSI EN 319 142). See chapter "How to Create a PAdES Signature" in the user manual for more information.
    - **New** support for the GlobalSign Digital Signing Service as cryptographic provider to create signatures and timestamps.
    - **New** signature algorithm RSA with SSA-PSS (PKCS#1v2.1) can be chosen by setting the provider session property `SigAlgo`.
    - **New** ability to add multiple signatures to encrypted files.
- Stamping
    - **New** attribute `flags` of `<stamp>`, e.g. to create modifiable stamps or stamps that are only visible when printing.
    - **New** attribute `src` of `<image>` allows a HTTP URL or file path.
    - **New** ability to add or modify stamps of signed files that are also encrypted.
- **New** support for writing PDF objects into object streams. Most objects that are contained in object streams in the input document are now also stored in object streams in the output document. When enabling linearization, however, no objects are stored in object streams.
- **Improved** robustness against corrupt input PDF documents.
- **New** option `-dss`: Add signature validation information to the document. This option can be used to create signatures with long-term validation material or to enlarge the longevity of existing signatures.
- **New** option `-st` to set signature format, e.g. for new European PAdES norm.
- **New** options `-afs1` and `-afs2` to set font size of the signature appearance.

9.8 Changes in version 4.9

- **Improved** behavior: Before signing, missing appearance streams of form fields are created, because otherwise Adobe® Acrobat® cannot validate the signature.
- Stamping:
  - **New** tag `<link>` to add interactive web links.
  - **New** tag `<text>` allows to format spans in continuous text using nested `<span>` tags.
- **Improved** support for and robustness against corrupt input PDF documents.
- **Improved** repair of embedded font programs that are corrupt.
- **New** support for OpenType font collections in installed font collection.
- **Improved** metadata generation for standard PDF properties.

9.9 Changes in version 4.8

- **New** feature: Images used as signature appearance background or for stamping for PDF/A input files may now have any color space, even if it differs from the input file's output intent.
- **Improved** creation of annotation appearances to use less memory and processing time.
- **Added** repair functionality for TrueType font programs whose glyphs are not ordered correctly.

10 Licensing, copyright, and contact

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

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

Contact

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