JSignPdf Quick Start Guide

Digital signatures for your PDF documents
version 1.1.0

http://jsignpdf.sourceforge.net/

Table of Contents

1 JSignPdf Introduction

JSignPdf is an open source application which adds digital signatures to PDF documents. It's written in Java programming language and it can be launched on the most of current OS (MS Windows, Linux, Mac OS X, ...). User can control the application using simple Swing GUI or command line arguments. Main features:

- supports visible signatures
- can set certification level
- supports PDF encryption with setting rights
- timestamp support
- certificate revocation checking (CRL and/or OCSP)

1.1 Benefits of digital signatures (source: Wikipedia)

Below are some common reasons for applying a digital signature to communications:

1.1.1 Authentication

Although messages may often include information about the entity sending a message, that information may not be accurate. Digital signatures can be used to authenticate the source of messages. When ownership of a digital signature secret key is bound to a specific user, a valid signature shows that the message was sent by that user. The importance of high confidence in sender authenticity is especially obvious in a financial context. For example, suppose a bank's branch office sends instructions to the central office requesting a change in the balance of an account. If the central office is not convinced that such a message is truly sent from an authorized source, acting on such a request could be a grave mistake.

1.1.2 Integrity

In many scenarios, the sender and receiver of a message may have a need for confidence that the message has not been altered during transmission. Although encryption hides the contents of a message, it may be possible to change an encrypted message without understanding it. (Some encryption algorithms, known as nonmalleable ones, prevent this, but others do not.) However, if a message is digitally signed, any change in the message will invalidate the signature. Furthermore, there is no efficient way to modify a message and its signature to produce a new message with a valid signature, because this is still considered to be computationally infeasible by most cryptographic hash functions

1.2 License

JSignPdf is released under LGPL and/or MPL license. It means, it can be freely used for personal use, for academic use, by government institutions, by non-profit making organizations and by commercial organizations. JSignPdf, or parts of it, can also be freely incorporated into commercial products. For more details look directly to license files.

1.3 History

Project started on the beginning of 2008 as a response for a request for PDF-signing Add-On to OpenOffice.org which came up from Czech OpenOffice.org users community.

1.4 Author

Author of the JSignPdf is a Czech developer Josef Cacek. He works in Java from 2000. He is a member of OpenOffice.org developers community and he made first Windows releases of OpenOffice.org in Czech language.

Josef Cacek likes (besides programming) also reading books and his second open source project EBookME is aimed on creating e-books for mobile devices with Java ME support. This application is hosted on the SourceForge.net too. http://ebookme.sourceforge.net/

Email: <josef.cacek@gmail.com>

1.5 Support JSignPdf

JSignPdf project has a constant need for several types of support from the user community. If you find JSignPdf useful, you are strongly encouraged to find a way to contribute. If none of the suggestions below are right for you, feel free to propose an alternative by sending e-mail to josef.cacek@gmail.com

- Help to internationalize. We look for people which will provide translations to new languages or corrections for the current. The main parts of translations are application itself, this guide, project web pages.
- Donate your time and skills. Programmers who enjoy writing Java applications are naturally always welcome.
- Tell people about the project and why to sign their documents.

2 Prerequisites

2.1 Java

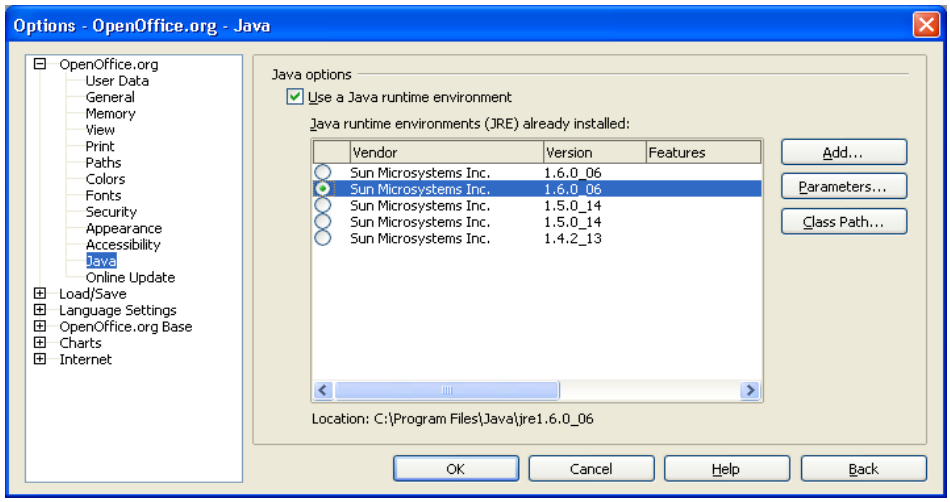
If you want to use JSignPdf, you will need Java Runtime Environment (JRE) version 5 or newer on your computer. If you don't have it, you can download it freely from Sun web pages:

http://java.sun.com/javase/downloads/index.jsp

To direct support of MS Windows certificates is required Java version 6. When you install JSignPdf from windows installer, you can select Java Runtime as one of the components to install, and then you don't need to download JRE by yourself.

If you use JSignPdf OpenOffice.org Add-On, you have to allow Java in OpenOffice.org preferences.

Start OpenOffice.org, go to menu *Tools* → *Options*... Select *OpenOffice.org* → *Java* in tree menu and choose Java installation from the list



2.2 Keystore

To sign PDF documents you need keystore with your private key. The most common keystores supported by Java are:

- PKCS#12 – keys stored in .p12 and .pfx files
- JKS (Java Key Store)
- WINDOWS-MY – supported only on MS Windows with Java 6 and newer. You can use directly your certificates imported in your system.

2.2.1 Exporting PKCS12 certificates from Internet Explorer

Guide of National Bank of Belgium:

http://www.nbb.be/doc/dq/E_pdf_dq/certificates_management_v.1.0_EN.pdf

2.2.2 Java Key and Certificate Management Tool

Keytool can handle JKS keystore files. It is a part of Java Runtime installation.

Documentation for keytool from Java 6 is here:

http://java.sun.com/javase/6/docs/technotes/tools/windows/keytool.html

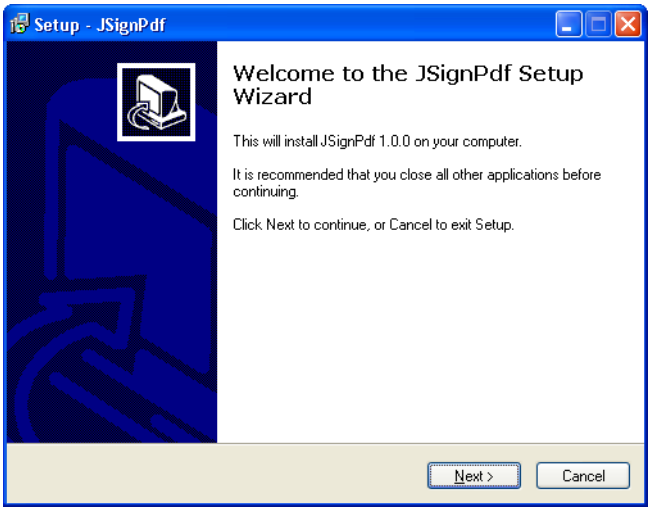
3 Installation

This chapter describes how to install JSignPdf using Windows installer, zip package and how to enable JSignPdf as OpenOffice.org Add On.

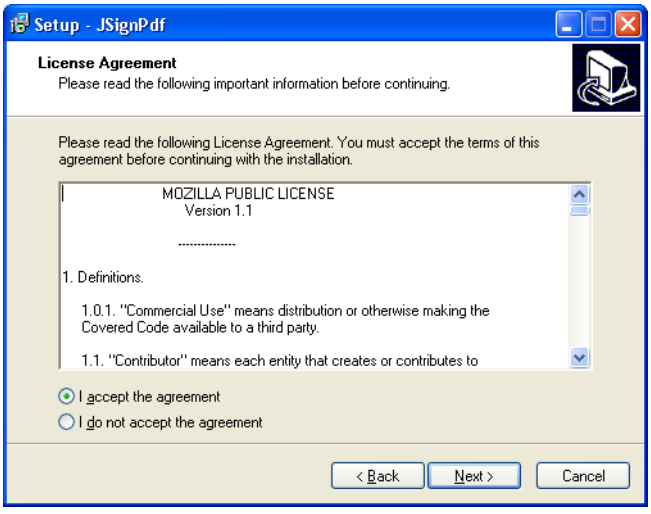
3.1 Windows installer

Windows installer contains ready to use version – the Java Runtime is also included in the package.

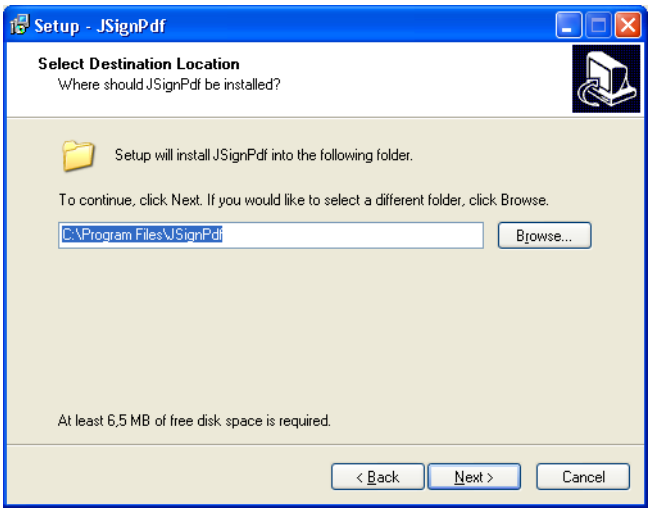
Download file `JSignPdf_setup_1.1.0_wjre.exe` and run it.



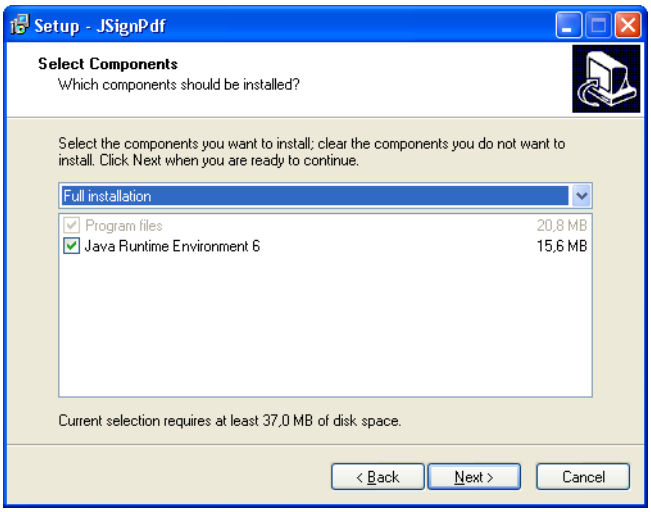
Accept the license agreement.



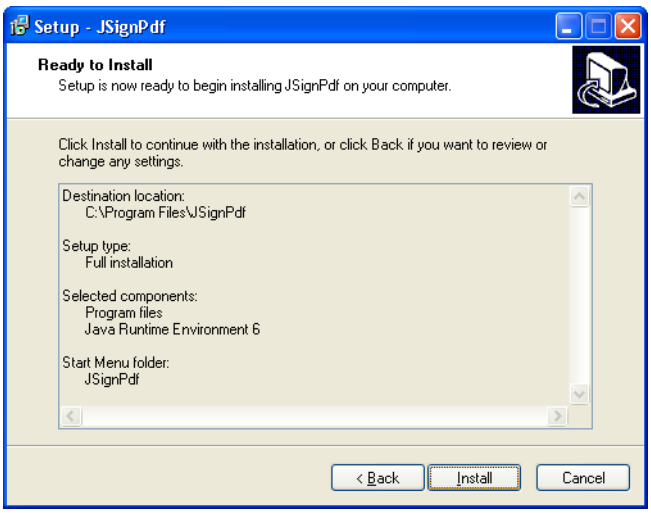
Choose the installation path.



Choose a Start menu Group name.



Verify your settings and let it Install.



3.2 Zip package

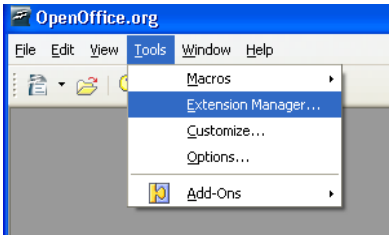
If you are advanced user or you have another OS than MS Windows, you can install JSignPdf from the zip archive file. Download file `JSignPdf-1.1.0.zip` and unpack it into your preferred directory with your preferred archiver or simply using the command line:

```
unzip JSignPdf-1.1.0.zip -d /my/preferred/path
```

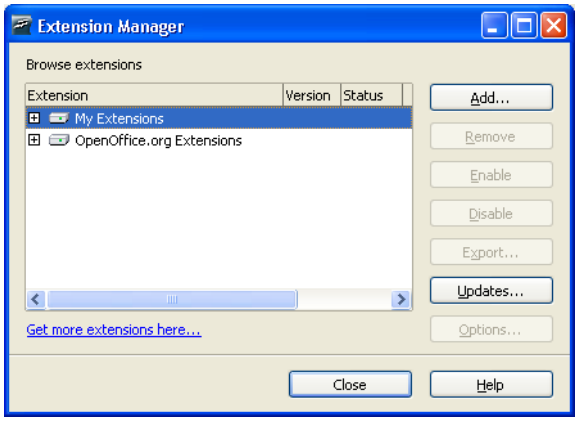
3.3 OpenOffice.org Add-On

Download file `JSignPdf-1.1.0.oxt` from JSignPdf download page.

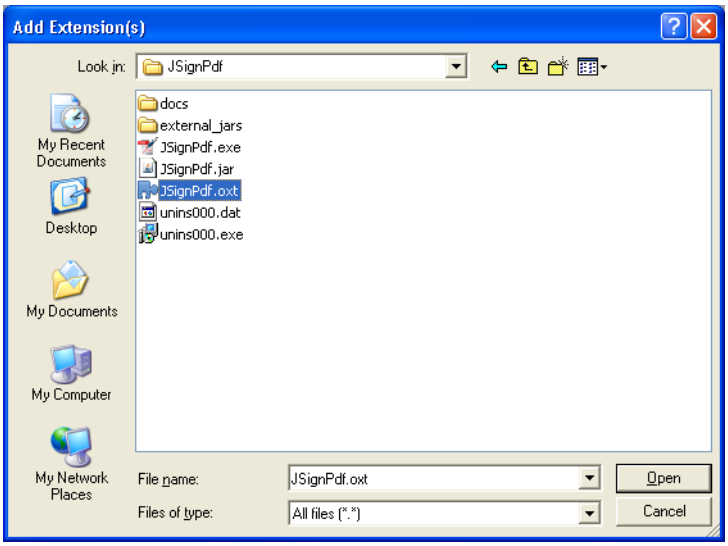
Run OpenOffice.org and from *Tools* menu choose item *Extension Manager...*



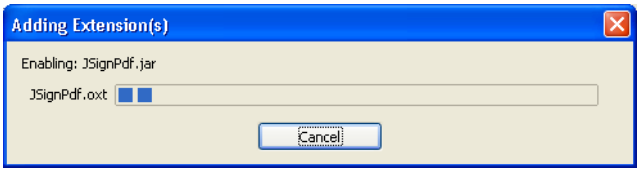
In extension manager window select *My Extensions* (or *OpenOffice.org Extensions* if you want to install it for all users) and press *Add...* button



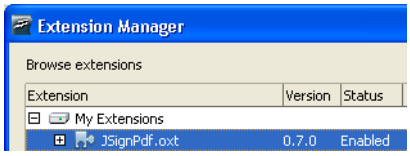
From standard open-file dialog select extension file `JSignPdf-1.1.0.oxt` and press *Open* button



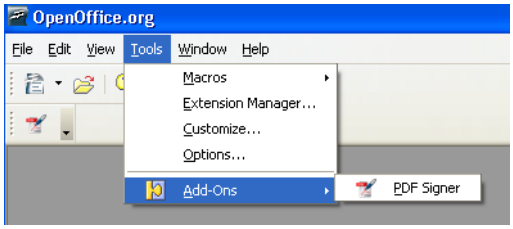
You should see installation progress progress dialog



and then JSignPdf in the extensions list



Restart your OpenOffice.org (quickstarter too) and check presence of the new *JSignPdf* toolbar icon and menu item *PDF Signer* in the menu *Tools → Add-Ons*



4 Launching

4.1 Windows Start menu

If you've installed JSignPdf from the windows installer package, there is a new Group in your system Start menu: *Start → Programs → JSignPdf → JSignPdf 1.1.0*

4.2 Without start menu

All platforms (with Java installed) should support launching of jar file `JSignPdf.jar`. Use following command in the directory, where the application is located.

```
$java -jar JSignPdf.jar
```

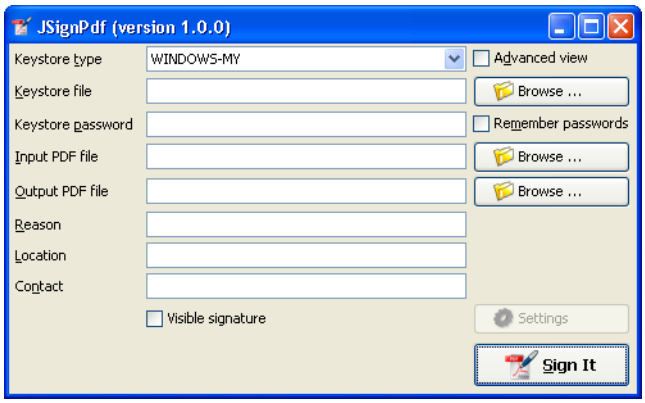
4.3 OpenOffice.org Add-On

Press JSignPdf icon in OpenOffice.org toolbar or choose in menu *Tools → Add-Ons → PDF Signer*

5 Using JSignPdf – signing PDF files

5.1 Simple version

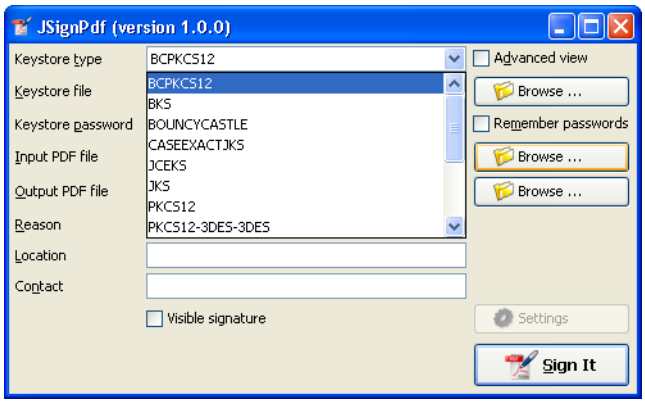
Fill text fields and press *Sign It* button.



5.2 More detailed version

5.2.1 Select Key Store Type

Keystore means location where the certificates (private or public keys) are located. Java 5 should support at least JKS and PKCS#12 keystores. Java 6 also supports native Windows certificate keystore "WINDOWS-MY".



By default, JSignPdf displays keystore types provided by Java (Sun Provider) and Bouncy Castle cryptographic provider.

More info: http://java.sun.com/javase/6/docs/technotes/guides/security/SunProviders.html

5.2.2 Keystore file and password

If you use JKS or PKCS#12 keystores, you have to select file where the keys are stored and provide password of this file. Path to the keystore file can be inserted directly by typing or you can use *Browse* button to navigate through the file system with Open File Dialog.

For Windows certificates (WINDOWS-MY keystore), you can leave these fields empty.

5.2.3 Input and Output PDF files

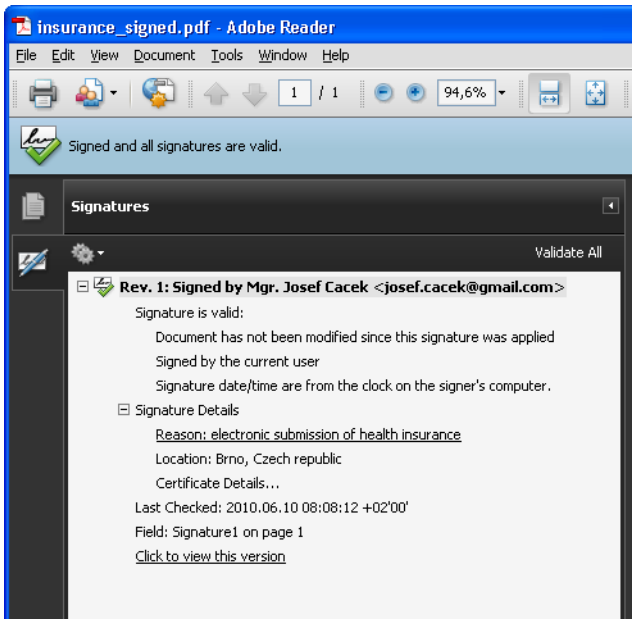
Input PDF file is an existing PDF file to which should be added digital signature.

Output PDF file is a name of result PDF file. If the value is not filled, automatically will be used the *Input PDF file* with additional suffix "`_signed`" (e.g. input `test.pdf` will result in `test_signed.pdf`)

The Input and Output files has to be different!

5.2.4 Reason, location, contact

Reason, location and contanct fields provide additional information about signature. Filled values will be stored in the result PDF.



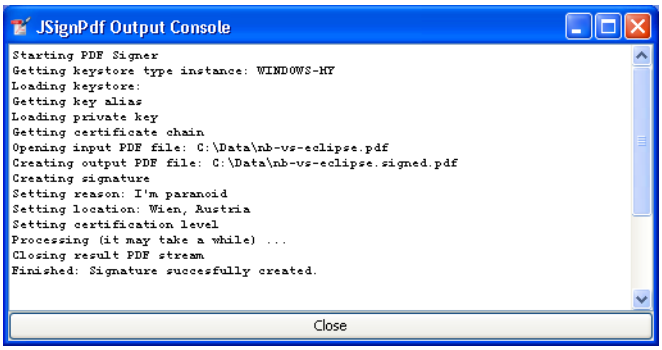
5.2.5 Remember passwords

JSignPdf stores filled information when you are exiting application, so it's present when you run it the next time. Passwords are not stored by default, but you can allow it by selecting checkbox *Remember passwords*.

Even if the password is stored in the encrypted form, we do not recommend to store passwords if your computer is used by more users!

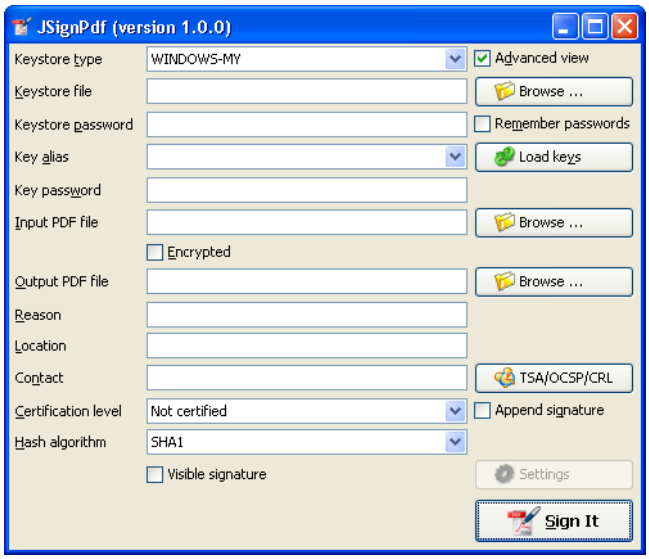
5.2.6 Sign It

Button *Sign It* starts the signing process. It displays console window and you can see what the program is doing.



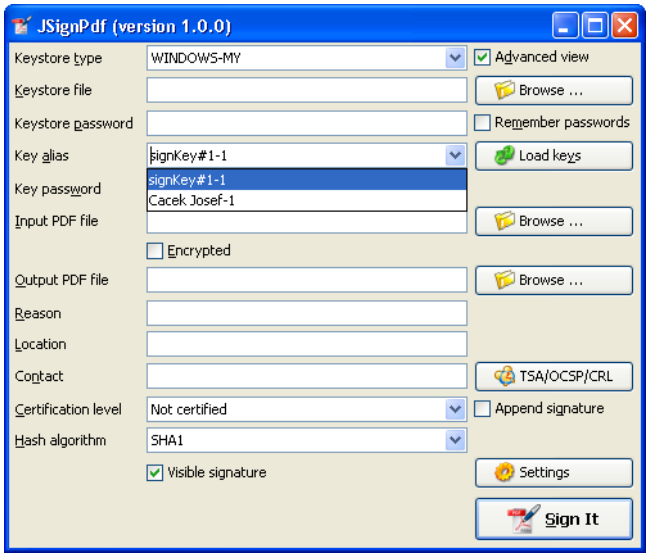
5.3 Advanced view

If you are more experienced user or you have to handle with encrypted PDFs or you have more keys stored in your keystore, you can use *Advanced view* checkbox to enable additional functionality.



5.3.1 Key alias

When you have more keys (certificates) stored in the keystore, you can select which one will be used to sign PDF file by filling *Key alias* field. Either you can type alias name directly (combo box is editable) or you can load all names by pressing *Load keys* button and then select one from drop-down list.



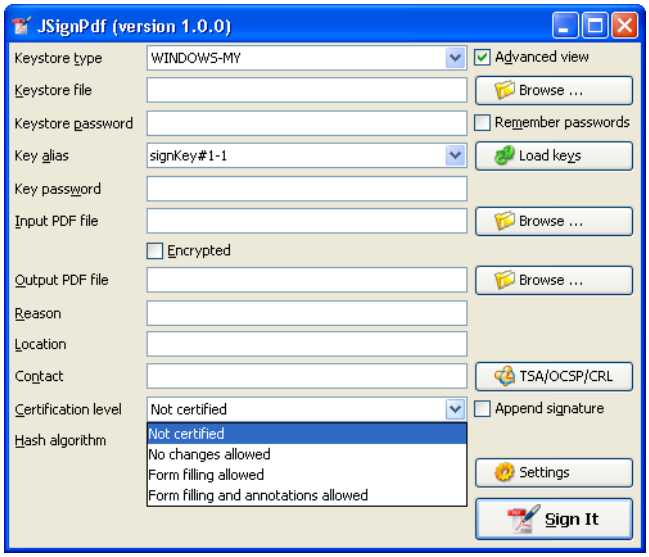
Only the certificates, which are valid (during the signing) are displayed in the list. If the certificate supports Key Usage extension, it will only be displayed if it is meant for signing.

5.3.2 Key password

Each key in keystore can be protected with its own password. If this password differs from the password of keystore, fill it to *Key password* input field.

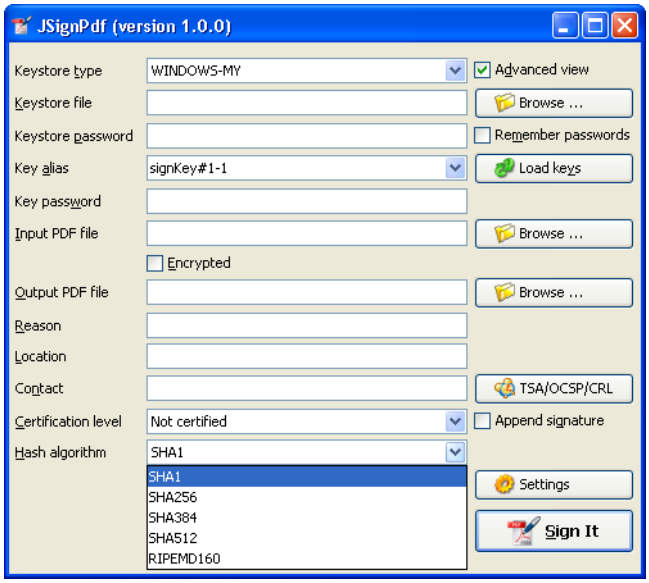
5.3.3 Certification level

The JSignPdf application is able to add certificate to the signed PDF. There are four levels of certification as you can see from the screenshot:



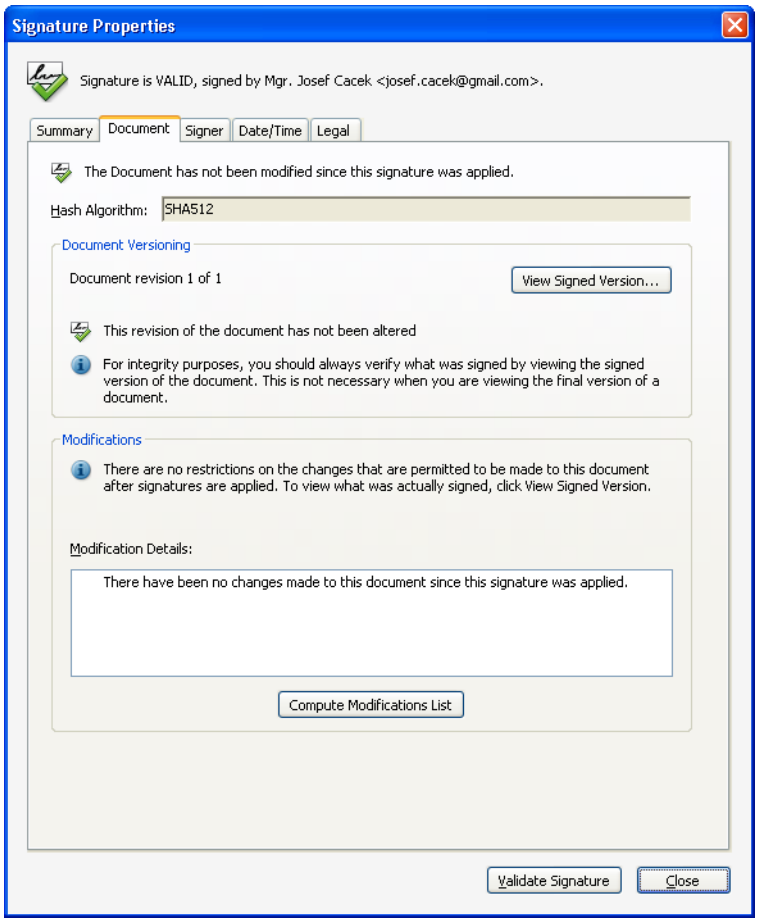
5.3.4 Hash algorithms

You can choose, which hash function will be used for signature. SHA-1 is the most common and should be supported by all keystore types. For better protection of signatures you can choose another one, but you can get then exception in console window during signing, if it's not supported



by chosen keystore provider. (Bouncy castle provider has better support than the Java one, so if you use for instance BCPKCS12 keystore type, all hash algorithms should work correctly.)

In Acrobat reader, you can display signature properties and on tab Document is the information about used algorithm.

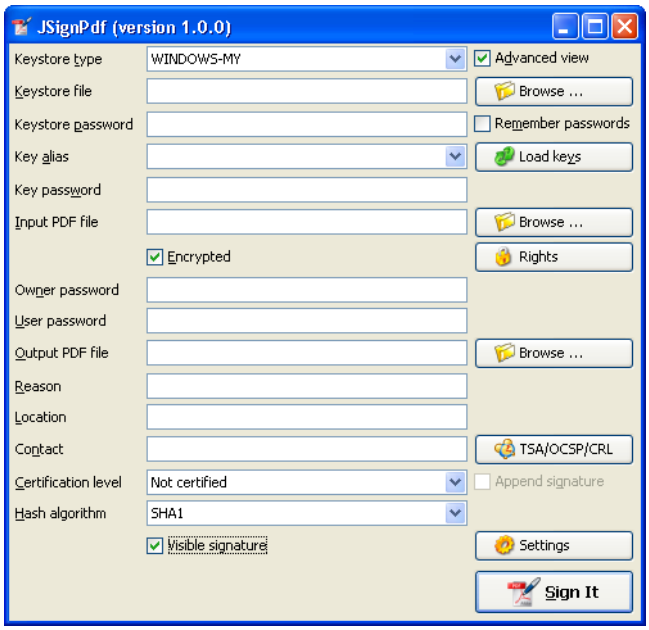


5.3.5 Append signature

JSignPdf can work in two signing modes. It replaces existing signatures with the new one by default. If you select *Append signature* checkbox, the new one will be appended and the old signatures will stay unchanged. ***This option is not possible in encrypted documents.***

5.4 Encryption

Encrypted checkbox enables additional fields for support of PDF security. By using this you can either sign secured PDFs (and change the rights and user password) or you can add encryption to unencrypted PDF during signing.



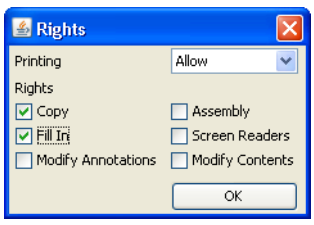
5.4.1 Owner and user passwords

Fill owner and user passwords to set it in secured result PDF. If the input PDF is encrypted, the *Owner password* field has to match to owner password of input PDF.

5.4.2 Rights

You can set allowed actions in encrypted result PDF by pressing *Rights* button. A new modal window will be displayed and you can set the possible options there.

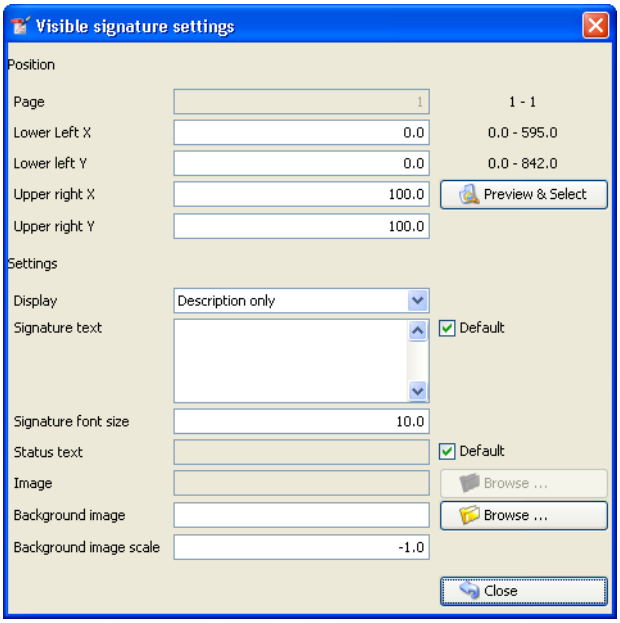
Normal rights are represented by checkboxes. Printing right has 3 levels, so the combobox is used for it.



5.5 Visible signature

Checkbox *Visible signature* allows you to create visible field with signature directly in the signed PDF. If the chebox is checked, button *Settings* is enabled and you can configure parameters (position/texts/images) of visible signature.

Read ToolTip texts, which are assigned to some input fields. You will get information, how to fill them correctly.



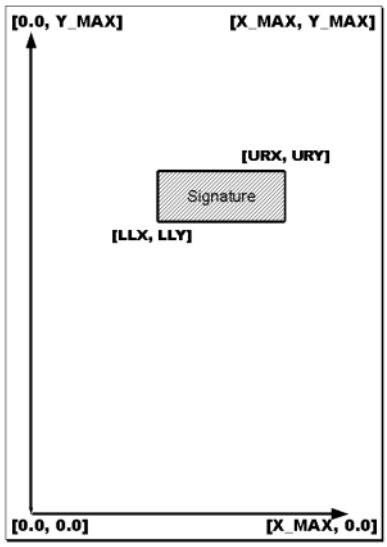
5.5.1 Page

Page number (counted from 1) to which the signature will be added.

5.5.2 Signature corners

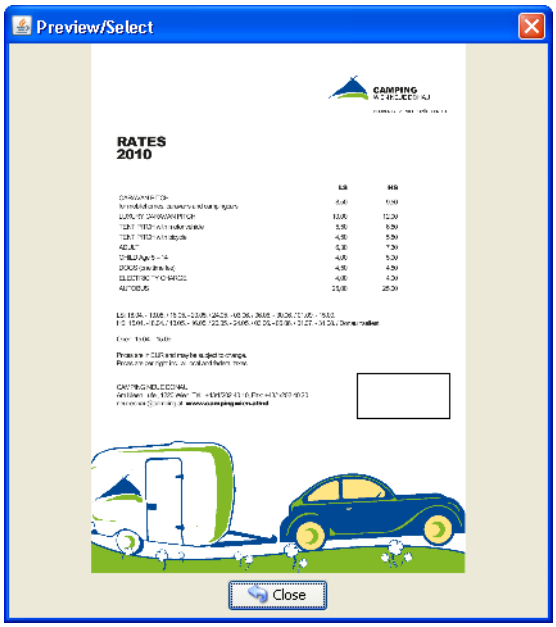
Next four inputs *Lower Left (X, Y)* and *Upper Right (X, Y)* defines position of signature on the page. You can fill in float numbers (with decimal places) as input. If you have already selected input PDF in the main window you will see possible range for X and Y values on the right side of *Lower Left (X, Y)* input fields.

Position of signature on page is bounded by lower left corner and upper right corner. Position of elements in PDF has base on the left bottom corner of page([0,0]).



5.5.3 Preview / Select button

PDF preview is supported from version 1.0.0. The borders of visible signature are displayed on the chosen page. You can select new position by pressing left mouse button at start corner, moving to end corner and releasing the mouse.



5.5.4 Display

In combobox *Display* you can set which fields will be generated to visible signature.

5.5.5 Texts and Images

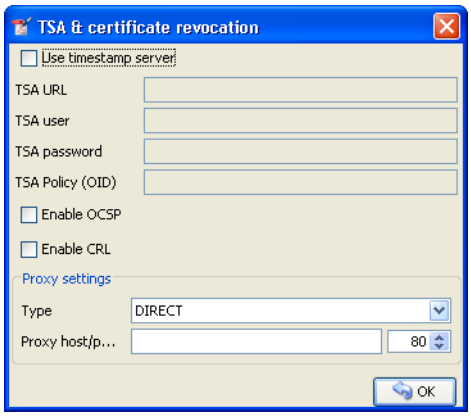
Signature Text, *Status Text*, *Image* and *Background Image* inputs define content of fields in visible signature. *Signature Font Size* is used for setting size of *Signature Text*, it should contain positive decimal number.

Background image scale defines size of background image. Any negative number means, the best-

fit algorithm will be used. Zero value means stretch, which fills whole field – it doesn't keep image ratio. Positive value means the multiplicator of original size.

5.6 TSA – timestamps

To add timestamp into signature you will need some timestamping authority (TSA) . Fill server address into *TSA URL* field and if the server requires authentication fill also *TSA User* and *TSA Password* fields. You can also set *TSA Policy OID*, which will be send to TSA server in the request, but probably you will not need to do so and the server use the right policy by itself.



You can try following URL for testing (doesn't require authentication):

http://dse200.ncipher.com/TSS/HttpTspServer

5.7 Certificate revocation checking

JSignPdf supports two standard ways of certificate revocation checking – CRL and OCSP. Most of the X.509 certificates supports CRL, but it has some disadvantages (for instance the size of list and possibly outdated information). The second – OCSP solves the mentioned issues, but not all Certification Authorities (CA) supports this protocol.

5.7.1 CRL

RFC 3280, Internet X.509 Public Key Infrastructure, Certificate and Certificate Revocation List (CRL) Profile.

Wikipedia says: In the operation of some cryptosystems, usually public key infrastructures (PKIs), a certificate revocation list (CRL) is a list of certificates (or more specifically, a list of serial numbers for certificates) that have been revoked or are no longer valid, and therefore should not be relied upon.

Such a list will be downloaded from CA and stored in PDF during signing process.

5.7.2 OCSP

RFC 2560, X.509 Internet PKI Online Certificate Status Protocol-OCSP.

Wikipedia says: The Online Certificate Status Protocol (OCSP) is an Internet protocol used for obtaining the revocation status of an X.509 digital certificate. It is described in RFC 2560 and is on the Internet standards track. It was created as an alternative to certificate revocation lists (CRL),

specifically addressing certain problems associated with using CRLs in a public key infrastructure (PKI). Messages communicated via OCSP are encoded in ASN.1 and are usually communicated over HTTP. The "request/response" nature of these messages leads to OCSP servers being termed OCSP responders.

If OCSP is enabled in JSignPdf and the protocol is supported by the certificate, the OCSP request will be created and response will be stored in signed PDF.

5.8 Proxy settings

If some “online” feature (TSA, CRL, OCSP) is enabled and JSignPdf runs behind a firewall, you can set the proxy, which will be used for all internet connections. Proxy type DIRECT means no proxy will be used.

6 InstallCert Tool

In some cases, when the JSignPdf connects to server through HTTPS protocol (e.g. to TSA server for timestamping), it can fail with console message “SSLHandshakeException”. It's caused because Java uses keystore (named “cacerts”) with preinstalled well-known certification authorities root certificates and if the HTTPS server doesn't have certificate signed by a such registered authority, the connection is refused.

If you trust the server, which was refused, you can add its certificate (or some parent certificate in the certificate chain) to the Java cacerts keystore. JSignPdf comes with command line utility for it – InstallCert.

Usage:

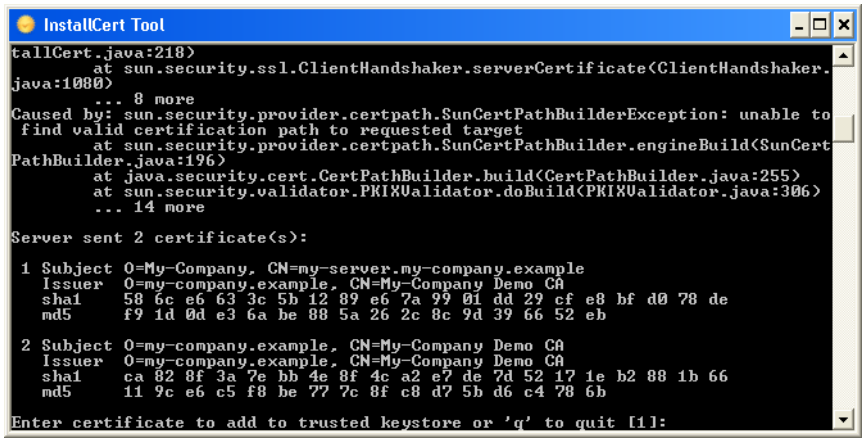
```
$java -jar InstallCert.jar
```

or

```
$java -jar InstallCert.jar hostname[:port] [cacertPwd]
```

If you don't provide a hostname argument, you will be asked for it.

The certificate chain will be displayed and you can choose which one will be imported.

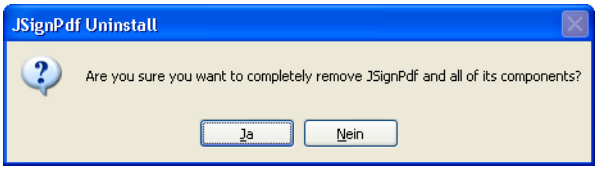


7 Uninstall

This chapter describes how to uninstall/remove JSignPdf from a computer.

7.1 Windows uninstaller

Choose *Programs* → *JSignPdf* → *Uninstall* and confirm the uninstallation.



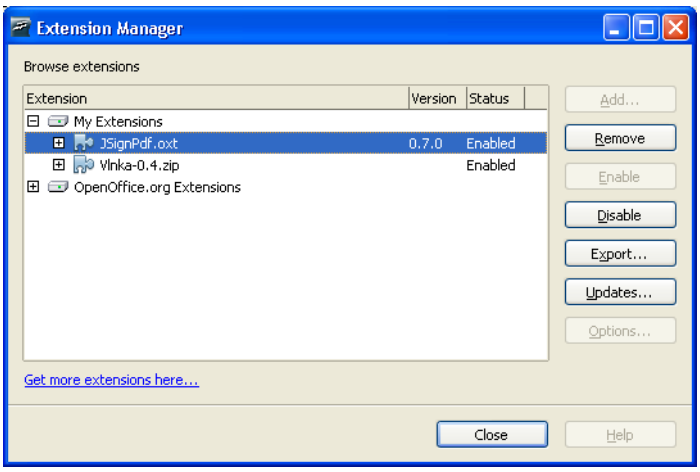
If you've installed OpenOffice.org Add-On during installation, you have to remove it manually, see chapter 7.3.

7.2 Zip package

Remove unpacked JSignPdf folder and you can also remove configuration file `.JSignPdf`, which is stored in user home directory.

7.3 OpenOffice.org Add-On

From main menu choose *Tools* → *Extension Manager* ... select JSignPdf entry and press *Remove* button.



Restart OpenOffice.org to complete uninstallation.

8 Solving problems

8.1 Out of memory error

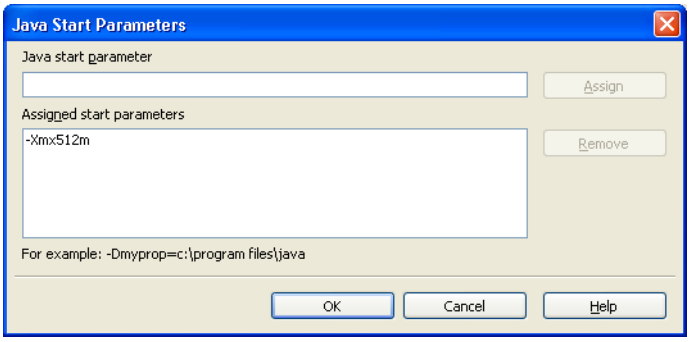
If you will see OutOfMemoryError in the program console, you need to allow java to use more memory.

Add -Xmx<size> switch to your java. Following example allows java to use 512MB (heap size).

```
$java -Xmx512m -jar JSignPdf.jar
```

8.1.1 OpenOffice.org Add-On

Configure this parameter in *Tools* → *Options...* → *OpenOffice.org* → *Java* by pressing *Parameters...* button.



8.2 Other problems – consult online FAQ

http://jsignpdf.sourceforge.net/en_US/faq.html

9 Command line (batch mode)

```
usage: java -jar JSignPdf.jar [file1.pdf [file2.pdf ...]] [-a] [--bg-path
       <file>] [--bg-scale <scale>] [-cl <level>] [--crl] [-d <path>]
       [--disable-assembly] [--disable-copy] [--disable-fill]
       [--disable-modify-annotations] [--disable-modify-content]
       [--disable-screen-readers] [-e] [-fs <size>] [-h] [-ha <algorithm>]
       [--img-path <file>] [-ka <alias>] [-ki <index>] [-kp <password>] [-ksf
       <file>] [-ksp <password>] [-kst <type>] [-l <location>] [--l2-text
       <text>] [--l4-text <text>] [-lk] [-lkt] [-llx <position>] [-lly
       <position>] [-lp] [-lpf <file>] [--ocsp] [-op <prefix>] [-opwd
       <password>] [-os <suffix>] [-pg <pageNumber>] [-pr <right>] [--proxy-host
       <hostname>] [--proxy-port <port>] [--proxy-type <type>] [-q] [-r
       <reason>] [--render-mode <mode>] [-ts <URL>] [--tsa-policy-oid
       <policyOID>] [-tsp <password>] [-tsu <username>] [-upwd <password>] [-urx
       <position>] [-ury <position>] [-V] [-v]
JSignpdf is an application designed to digitally sign PDF documents. If you
start the program without any command line argument, the GUI will be started,
otherwise you can use JSignPdf in command line batch mode.
 -a,--append                              add signature to existing ones. By
                                          default are existing signatures replaced
                                          by the new one.
    --bg-path <file>                      background image path for visible
                                          signatures
    --bg-scale <scale>                    background image scale for visible
                                          signatures. Insert positive value to
                                          multiply image size with the value.
                                          Insert zero value to fill whole
                                          background with it (stretch). Insert
                                          negative value to best fit resize.
 -cl,--certification-level <level>        level of certification. Default value is
                                          NOT_CERTIFIED. Available values are
                                          NOT_CERTIFIED,
                                          CERTIFIED_NO_CHANGES_ALLOWED,
                                          CERTIFIED_FORM_FILLING,
                                          CERTIFIED_FORM_FILLING_AND_ANNOTATIONS
    --crl                                 enable CRL certificate validation
 -d,--out-directory <path>                folder in which the signed documents will
                                          be stored. Default value is current
                                          folder.
    --disable-assembly                    deny assembly in encrypted documents
    --disable-copy                        deny copy in encrypted documents
    --disable-fill                        deny fill encrypted documents
    --disable-modify-annotations          deny modify annotations in encrypted
                                          documents
    --disable-modify-content              deny modify content in encrypted
                                          documents
    --disable-screen-readers              deny screen readers in encrypted
                                          documents
 -e,--encrypted                           enables pdf encryption and allows to set
                                          user rights. Use this option together
                                          with -opwd and -upwd.
 -fs,--font-size <size>                   font size for visible signature text,
                                          default value is 10.0
 -h,--help                                prints this help screen
 -ha,--hash-algorithm <algorithm>         hash algorithm used for signature.
                                          Default value is SHA1. Available values
```

```
                                  are SHA1, SHA256, SHA384, SHA512,
                                  RIPEMD160
   --img-path <file>              image path for visible signature
-ka,--key-alias <alias>           name (alias) of the key, which should be
                                  used for signing the document. If this
                                  option is not given, the first key in the
                                  keystore is used. (List the key aliases
                                  using -lk)
-ki,--key-index <index>           zero based index of the key, which should
                                  be used for signing the document. If
                                  neither this option nor alias is given,
                                  the first key (index=0) in the keystore
                                  is used. (List the key aliases using
                                  -lk). This option has lower priority than
                                  alias.
-kp,--key-password <password>     password of the key in keystore. In most
                                  cases you don't need to set this option -
                                  only keystore is protected by a password,
                                  but just in case :)
-ksf,--keystore-file <file>       sets KeyStore file - as the value use the
                                  path on which is file with private key(s)
                                  located (.p12, .pfx, .jks, ...). Some
                                  keystores haven't keys stored in a file
                                  (e.g. windows keystore - WINDOWS-MY),
                                  then don't use this option.
-ksp,--keystore-password <password>  password to KeyStore
-kst,--keystore-type <type>       sets KeyStore type (you can list possible
                                  values for this option -lkt argument)
-l,--location <location>          location of a signatue (e.g. Washington
                                  DC). Empty by default.
   --l2-text <text>               signature text
   --l4-text <text>               status text
-lk,--list-keys                   lists keys in choosen keystore
-lkt,--list-keystore-types        lists keystore types, which can be used
                                  as values -kst option
-llx <position>                   lower left corner postion on X-axe of a
                                  visible signature
-lly <position>                   lower left corner postion on Y-axe of a
                                  visible signature
-lp,--load-properties             Loads properties from a default file
                                  (created by GUI application).
-lpf,--load-properties-file <file>  Loads properties from the given file. The
                                  file can be create by copying the default
                                  property file .JSignPdf created by the
                                  GUI in the user home directory.
   --ocsp                         enable OCSP certificate validation
-op,--out-prefix <prefix>         prefix for signed file. Default value is
                                  empty prefix.
-opwd,--owner-password <password>  owner password for encrypted documents
                                  (used when -e option is given)
-os,--out-suffix <suffix>         suffix for signed filename. Default value
                                  is "_signed". (e.g. sign process on file
                                  mydocument.pdf will create new file
                                  mydocument_signed.pdf)
-pg,--page <pageNumber>           page with visible signature. Default
                                  value is 1 (first page)
-pr,--print-right <right>         printing rights. Used for encrypted
                                  documents. Default value is
```

```
                                    ALLOW_PRINTING. Available values are
                                    DISALLOW_PRINTING,
                                    ALLOW_DEGRADED_PRINTING, ALLOW_PRINTING
   --proxy-host <hostname>          hostname or IP address of proxy server
   --proxy-port <port>              port of proxy server, default value is 80
   --proxy-type <type>              proxy type for internet connections.
                                    Default value is DIRECT. Possible values
                                    are DIRECT, HTTP, SOCKS
-q,--quiet                          quiet mode - without info messages during
                                    process
-r,--reason <reason>                reason of signature. Empty by default.
   --render-mode <mode>             render mode for visible signatures.
                                    Default value is DESCRTIPTION_ONLY.
                                    Possible values are DESCRTIPTION_ONLY,
                                    GRAPHIC_AND_DESCRIPTION,
                                    SIGNAME_AND_DESCRIPTION
-ts,--tsa-server-url <URL>          address of timestamping server (TSA). If
                                    you use this argument, the timestamp will
                                    be included to signature. (For testing
                                    purposes you can try following URL
                                    http://dse200.ncipher.com/TSS/HttpTspServ
                                    er)
   --tsa-policy-oid <policyOID>     TSA policy OID which should be set to
                                    timestamp request.
-tsp,--tsa-password <password>      TSA user password. Use this switch if you
                                    use timestamping (-ts) and TSA server
                                    requires authentication.
-tsu,--tsa-user <username>          TSA user name. Use this switch if you use
                                    timestamping (-ts) and TSA server
                                    requires authentication.
-upwd,--user-password <password>    user password for encrypted documents
                                    (used when -e option is given)
-urx <position>                     upper right corner postion on X-axe of a
                                    visible signature
-ury <position>                     upper right corner postion on Y-axe of a
                                    visible signature
-V,--visible-signature              enables visible signature
-v,--version                        shows the application version
```

9.1 Examples

9.1.1 Simplest signature on windows

```
$ java -jar JSignPdf.jar -kst WINDOWS-MY mydocument.pdf
```

creates copy of mydocument.pdf with name mydocument_signed.pdf, which is digitally signed with the first certificate found in default windows certificate store

9.1.2 PKCS12 signature with encryption

```
$ java -jar JSignPdf.jar -kst PKCS12 -ksf my_certificate.pfx -ksp
    myPrivateKeystorePassword -ka cert23 -e -opwd xxx123 -upwd 123xxx -pr
    DISALLOW_PRINTING mydocument.pdf
```

creates signed and encrypted file mydocument_signed.pdf, printing of the new file is not allowed. For signature is used key with alias cert23 from the file my_certificate.pfx

9.1.3 Listing KeyStore types

```
$ java -jar JSignPdf.jar -lkt
```

lists keystore types

9.1.4 Listing key aliases in a KeyStore

```
$ java -jar JSignPdf.jar -kst PKCS12 -ksf my_certificate.pfx -ksp
    myVeryPrivatePassword -lk -q
```

list names (aliases) of keys stored in my_certificate.pfx file using the password for keystore. Quiet mode is enabled so no debug info is printed.