JSignPdf Quick Start Guide

Digital signatures for your PDF documents
version 1.6.4

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

Below are some common reasons for applying a digital signature to communications. (source Wikipedia)

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.

Active development stopped by the author on September 2012.

1.4 Author

Author of the JSignPdf is a Czech developer Josef Cacek. He works in Java from 2000. Some links to the Josef's projects:

- https://sourceforge.net/users/kwart/
- https://github.com/kwart/

1.5 Getting support

If you don't find the relevant information in this document or on JSignPdf web page (http://jsignpdf.sourceforge.net/) use JSignPdf Google Group to ask the community.

https://groups.google.com/d/jsignpdf/

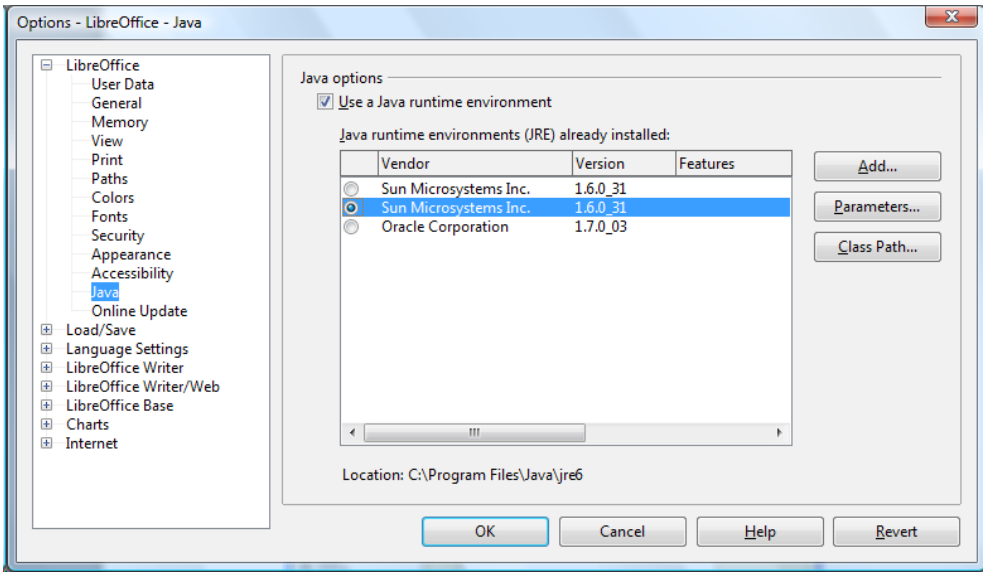
2 Prerequisites

2.1 Java

If you want to use JSignPdf, and you don't install it on Windows using the installation program, you will need Java Runtime Environment (JRE) version 5 or newer on your computer (recommended is the Java 6). If you don't have it, you can download it freely from web pages, for instance:

http://java.sun.com/

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



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

2.2 Keystore

To sign PDF documents you need a 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.

JSignPdf has been also extended to support external keystores like smart cards, or network HSMs. The first example is CloudFoxy (https://gitlab.com/cloudfoxy).

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

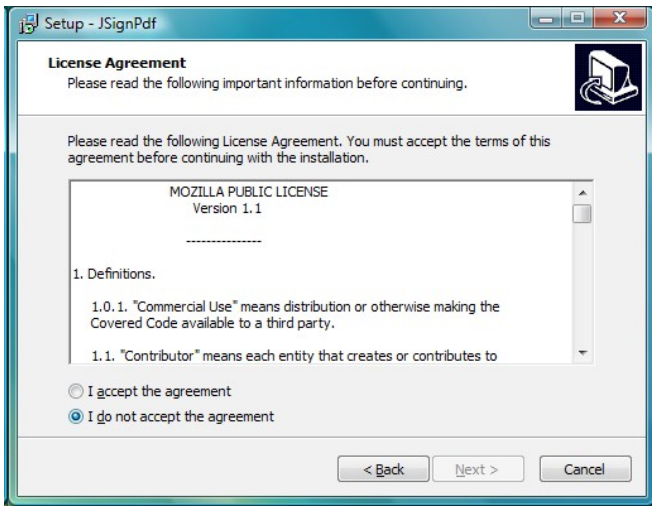
This chapter describes how to install JSignPdf using Windows installer, zip package and how to enable JSignPdf as OpenOffice.org/LibreOffice Add On.

3.1 Windows installer

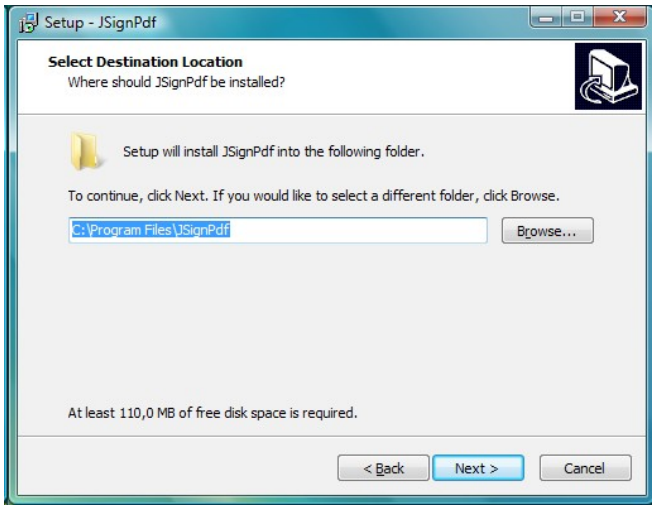
Windows installer contains ready to use version – the Java Runtime is also included in the package.

Download file `JSignPdf_setup_1.6.4_wjre.exe` and run it.

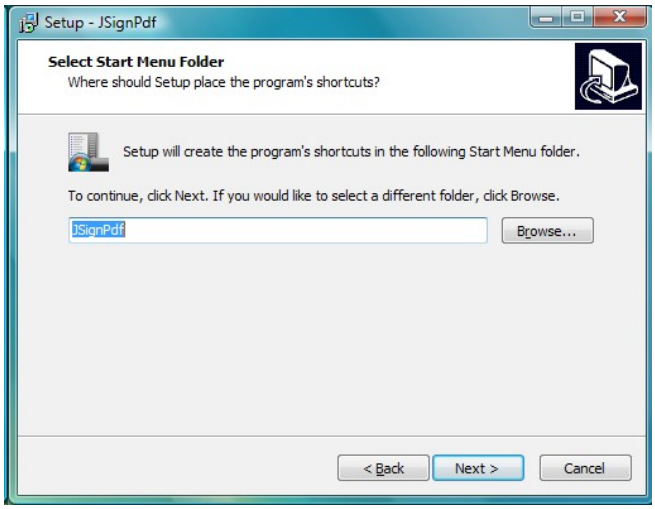
Accept the license agreement.



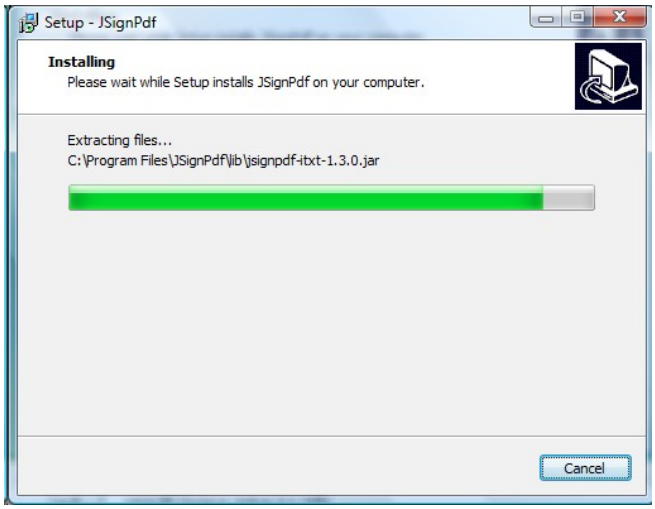
Choose the installation path.



Choose a Start menu Group name.



Verify your settings and let it Install.



3.2 Zip package

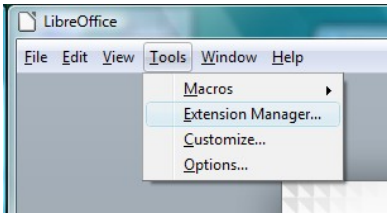
If you are advanced user or you have another OS than MS Windows, you can install JSignPdf from the zip archive file. Download file `JSignPdf-1.6.4.zip` and unpack it into your preferred directory with your preferred archiver or simply using the command line:

```
unzip JSignPdf-1.6.4.zip -d /my/preferred/path
```

3.3 OpenOffice.org/LibreOffice Add-On

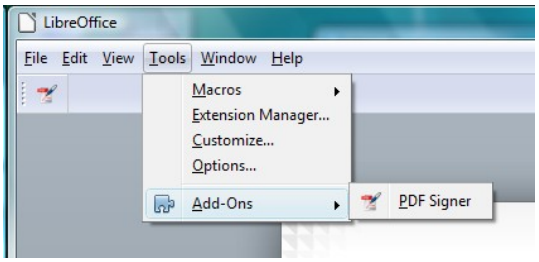
Download file `JSignPdf-1.6.4.oxt` from JSignPdf download page.

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



In extension manager window press *Add...* button, select extension file `JSignPdf-1.6.4.oxt` in the displayed open-file dialog and press *Open* button.

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


LibreOffice
File Edit View Tools Window Help
Macros
Extension Manager...
Customize...
Options...
Add-Ons
PDF Signer

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 1.6.4*

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
```

If you don't need see console output, use `javaw` command instead of `java`.

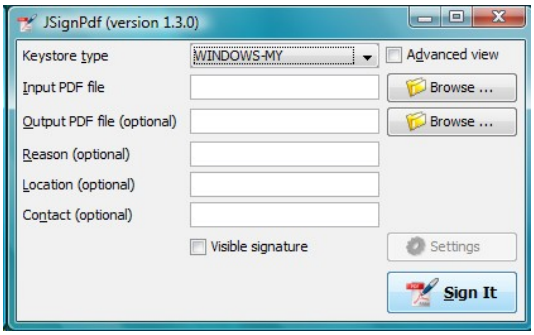
4.3 OpenOffice.org/LibreOffice Add-On

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

5 Using JSignPdf – signing PDF files

5.1 Simple version

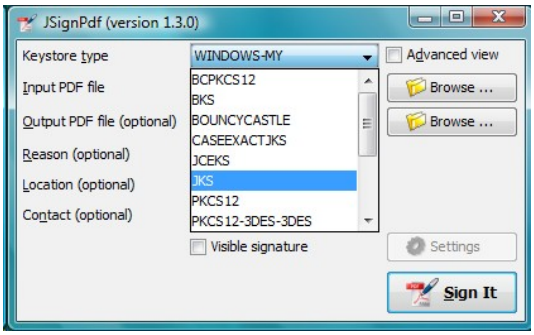
Fill text fields and press *Sign It* button.



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

JSignPdf has been extended to support remote/external keystores. The first entry is "CloudFoxy" (https://gitlab.com/cloudfoxy), which is a REST API for physical smart cards, initially developed to support eIDAS signatures.

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.

5.2.3 Input and Output PDF files

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

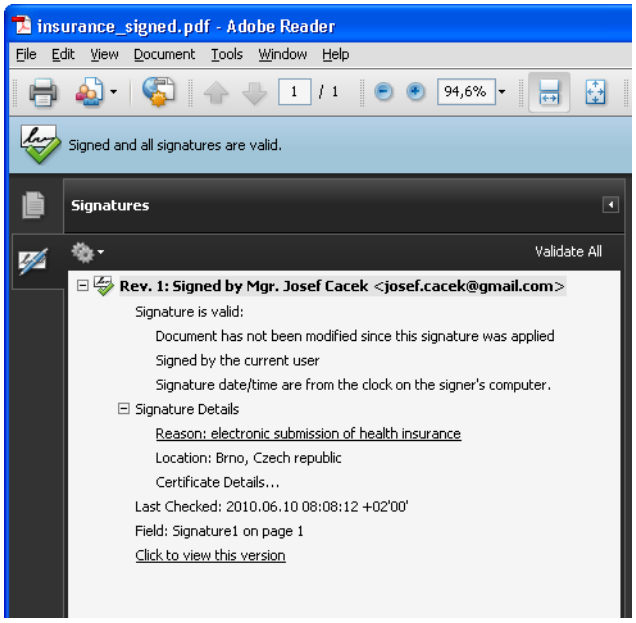
Output PDF file is a name of result PDF file. If the value is not filled, automatically will be used the

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 contact fields provide additional information about signature. Filled values will be stored in the result PDF.



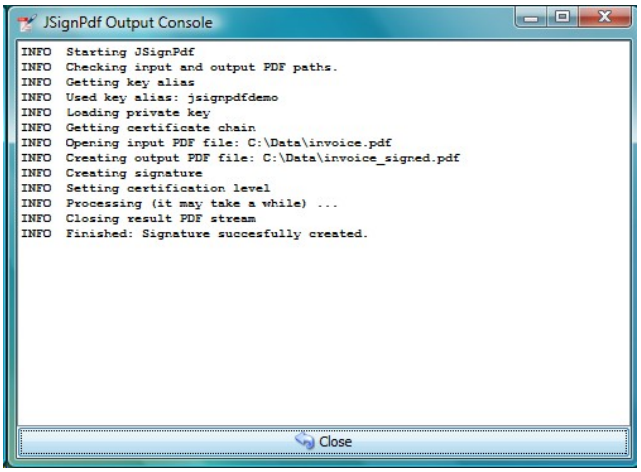
5.2.5 Remember passwords

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!

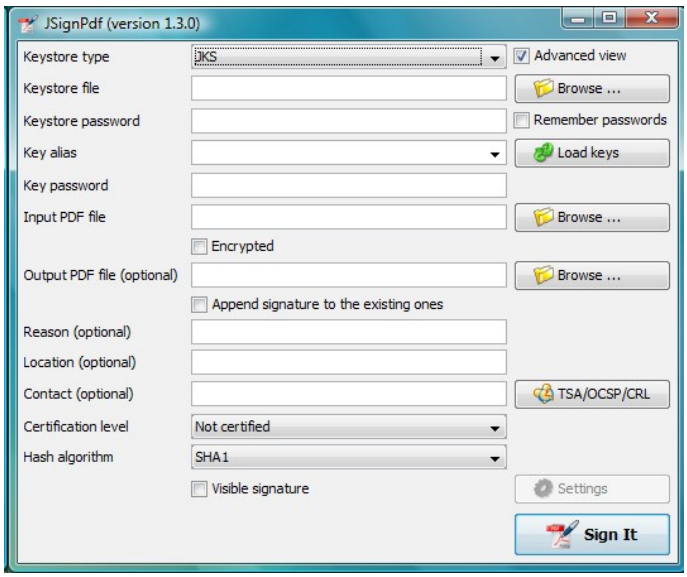
5.2.6 Sign It

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 private keys 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.[1]

If you don't fill *Key alias* field the first alias read from keystore will be used.

1 Only the private keys, which are valid (at the time of the signing) are displayed in the list. If the certificate supports Key Usage extension, the private key will only be displayed if it is meant for signing.

5.3.2 Key password

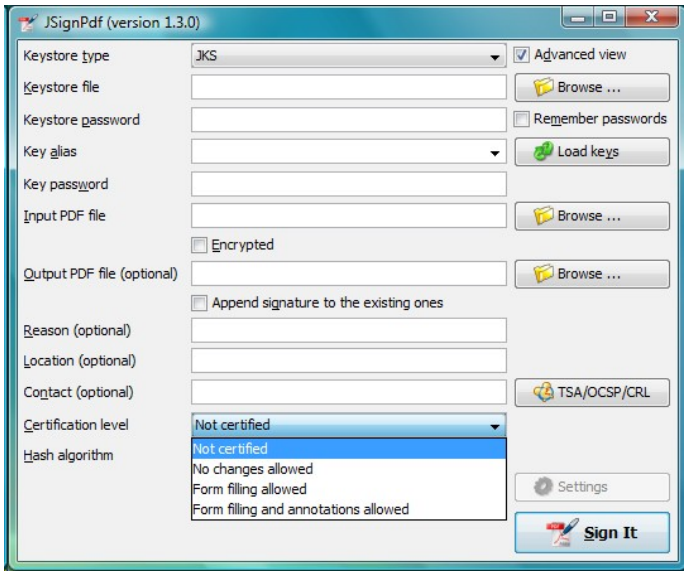
Each key in keystore can be protected with its own password. If this password differs from the password of keystore, fill it to *Key password* input field.

5.3.3 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 disabled for encrypted documents.***

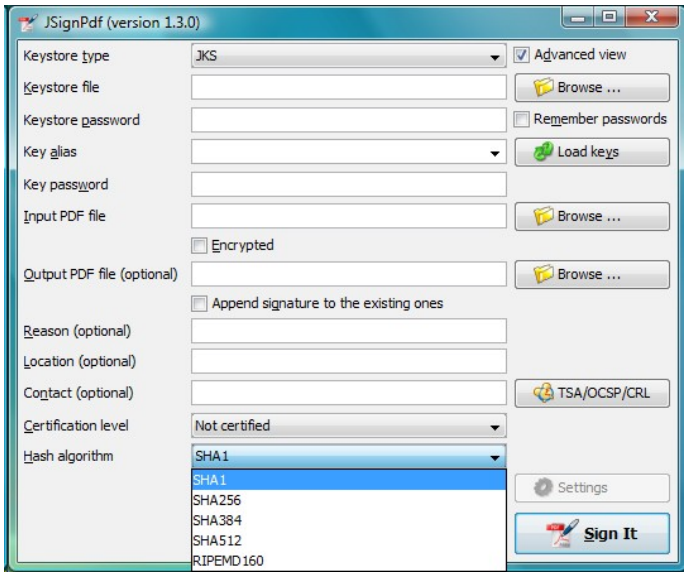
5.3.4 Certification level

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:

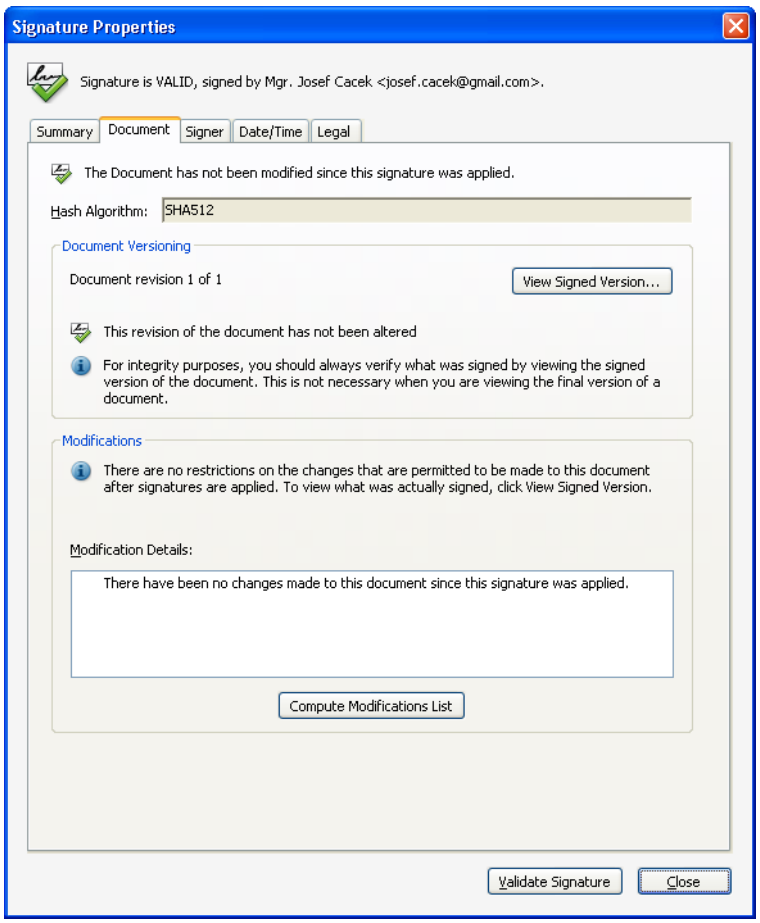


5.3.5 Hash algorithms

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 the 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 properly.)

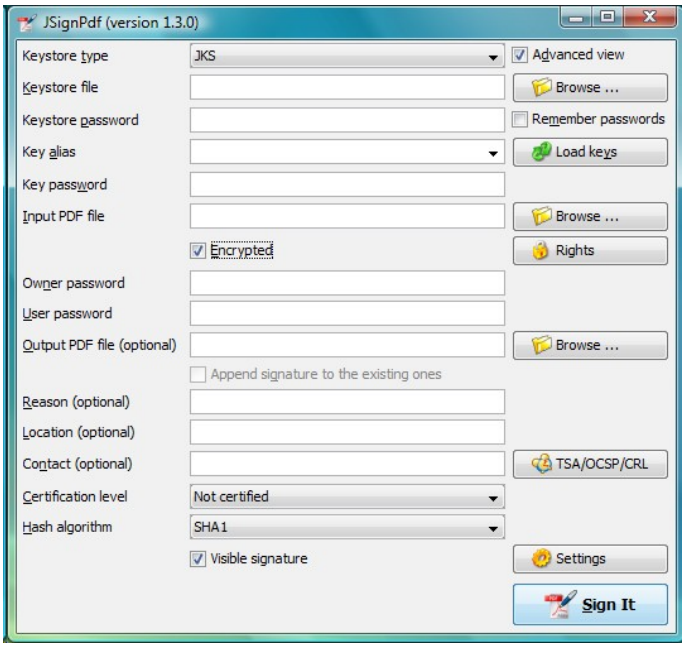


In Acrobat reader, you can display signature properties and on tab Document is the information about used algorithm.



5.4 Encryption

PDF Encryption combobox 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.



Outdated screenshot (without certificate encryption support)

5.4.1 Encryption: Passwords

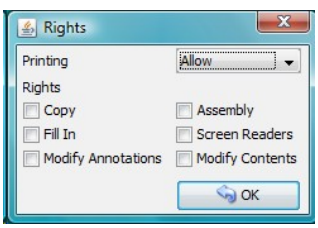
Fill owner and user passwords to set it in secured result PDF. If the input PDF is encrypted, the *Owner password* field has to match to owner password of input PDF.

5.4.2 Encryption: Certificate

Fill path to a certificate file (*.cer, *.crt, …) which should be used for the PDF encryption. Only the user, which has the private key for the certificate will be able to open the file.

5.4.3 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.

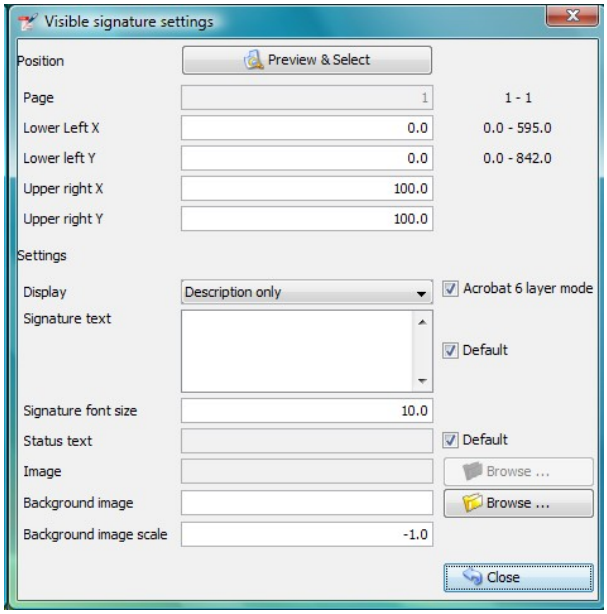


Normal rights are represented by checkboxes. Printing right has 3 levels, so the combobox is used

for it.

5.5 Visible signature

Checkbox *Visible signature* allows you to create visible field with signature directly in the signed PDF. If the checkbox 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.

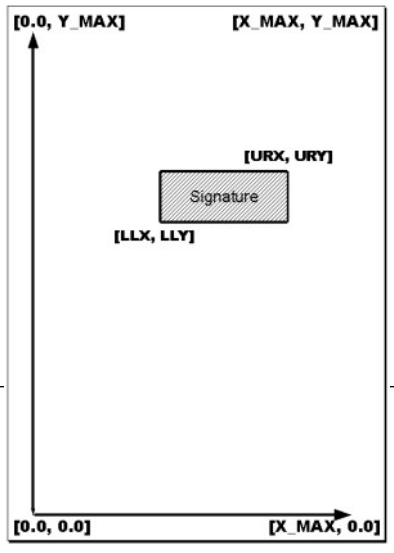
5.5.1 Page

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

5.5.2 Signature corners

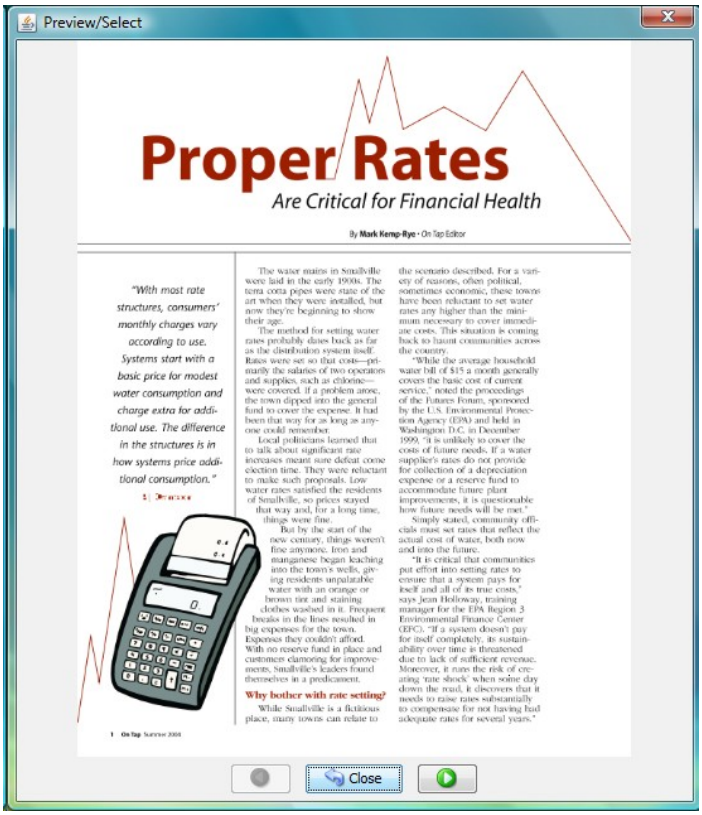
Next four inputs *Lower Left (X, Y)* and *Upper Right (X, Y)* defines position of signature on the page. You can fill in float numbers (with decimal places) as input. If you have already selected input PDF in the main window you will see possible range for X and Y values on the right side of *Lower Left (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 Acrobat 6 layers

The checkbox *Acrobat 6 layer mode* (checked by default) allows you to control which signature layers will be added to the signed document. Acrobat 6.0 and higher recommends that only layer n2 and n4 be present. If the checkbox is not selected then all layers will be created.

5.5.6 Texts and Images

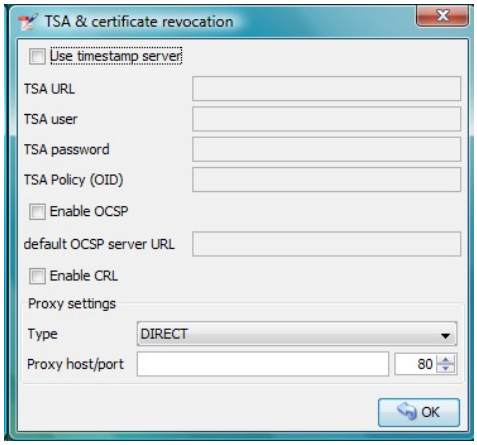
Signature Text, *Status Text*, *Image* and *Background Image* inputs define content of fields in visible signature. *Signature Font Size* is used for setting size of *Signature Text*, it should contain positive decimal number.

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 choose the authentication type

and fill either *TSA User* and *TSA Password* fields or path to the certificate's private key (it has to be PKCS#12 keystore) and the password. 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 URLs of free timestamping servers for testing (doesn't require authentication):

http://dse200.ncipher.com/TSS/HttpTspServer
http://tsa.starfieldtech.com
https://timestamp.geotrust.com/tsa

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 for the certificate, the OCSP request will be created and response will be stored in signed PDF. The URL of OCSP server is retrieved from certificate. If the OCSP part is not found in the signing certificate, the value from *default OCSP server URL* field will be used.

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 Using hardware tokens for signing

Steps to sign documents using hardware tokens:

1. Install PKCS#11 driver for your token. Check the vendor's documentation and install a proper driver for your system;
2. Create a configuration file pkcs11.cfg somewhere on your system. It will be used to configure a Java SunPKCS11 security provider. (see https://docs.oracle.com/javase/8/docs/technotes/guides/security/p11guide.html)
   The content depends on your driver, you can try to start with a simple 2 lines:

```
name=Test
library=/path/to/your/PKCSDriver.so
```

   If it doesn't work, try to add

```
slotIndex=1
```

3. Configure SunPKCS11 security provider to be used by your Java runtime. Edit the `java.security` file in your Java installation
   - `jre/lib/security/java.security` for Java 8 and older
   - `conf/security/java.security` for Java 9 and newer

   Add line:

```
security.provider.7=sun.security.pkcs11.SunPKCS11 /path/to/pkcs11.cfg
```

   The number 7 in `security.provider.7` property name means a preference order. You can either reuse an existing number or use the lowest unused one.
4. Try to run JSignPdf with PKCS11 debug enabled:

```
java -Djava.security.debug=pkcs11keystore \
    -Djava.security.debug=sunpkcs11 \
    -jar JSignPdf.jar
```

If the PKCS11 keystore type works properly in the GUI and you can use the certificate on your token, you're ready to use it also in the batch mode.

```
java -jar JSignPdf.jar -kst PKCS11 -ksp 123456 document.pdf
```

7 Advanced application configuration

Some advanced options are not controlled from GUI or command line. They can be only set directly in the appropriate configuration file.

7.1 conf.properties

The property file `conf/conf.properties` contain several option groups:

- visible signature font settings
- control the certificate checks
- PKCS#11 support
- enable more strict SSL handling

7.2 Java VM options using EXE launchers

If the Java VM properties has to be changed (e.g. maximum memory allowed) and the EXE wrapper is used, you can edit the appropriate `.l4j.ini` file (e.g. `JSignPdf.l4j.ini`).

The arguments should be separated with spaces or new lines, environment variable expansion is supported, for example:

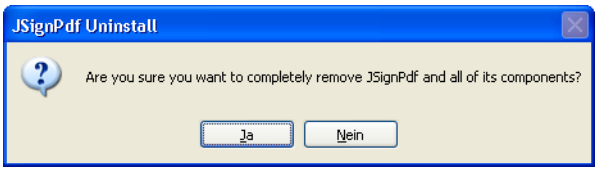
```
# Launch4j runtime config
-Dswing.aatext=true
-Dsomevar="%SOMEVAR%"
-Xms32m
-Xmx512m
```

8 Uninstall

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

8.1 Windows uninstaller

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



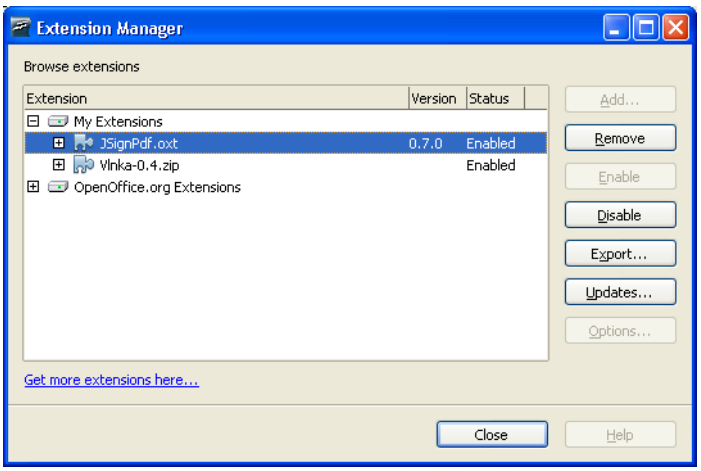
If you've installed OpenOffice.org/LibreOffice Add-On during installation, you have to remove it manually, see chapter 8.3.

8.2 Zip package

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

8.3 OpenOffice.org/LibreOffice Add-On

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



Restart OpenOffice.org/LibreOffice to complete uninstallation.

9 Solving problems

9.1 Out of memory error

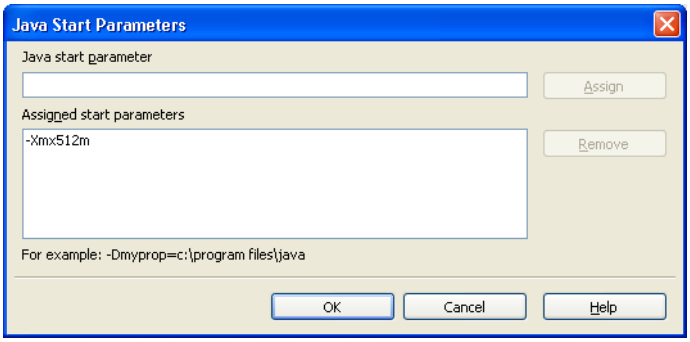
If you will see OutOfMemoryError in the program console, you need to allow java to use more memory.

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

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

9.1.1 OpenOffice.org/LibreOffice Add-On

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



9.2 Changing the application language

The language used by the program should be fit the regional settings in your system (if the translation is available), or the English is used (the default).

If you want use another translation, specify the java system property `user.language` and as the value use the country code (cs, de, fr, ja, pl, ...).

E.g. for the czech language:

```
$java -Duser.language=cs -jar InstallCert.jar
```

9.2.1 Change language when running through Windows EXE launcher

Open `JSignPdf.l4j.ini` (or `JSignPdfC.l4j.ini`) in any text editor like `Notepad.exe` and add a line with `user.language` specified at the end of file:

```
-Duser.language=cs
```

9.3 Other problems – consult online FAQ

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

10 Command line (batch mode)

```
usage: java -jar JSignPdf.jar [file1.pdf [file2.pdf ...]] [-a] [--bg-path
       <file>] [--bg-scale <scale>] [-c <contact>] [-cl <level>] [--crl] [-d
       <path>] [--disable-acrobat6-layer-mode] [--disable-assembly]
       [--disable-copy] [--disable-fill] [--disable-modify-annotations]
       [--disable-modify-content] [--disable-screen-readers] [-e] [-ec <file>]
       [-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]
       [--ocsp-server-url] [-op <prefix>] [-opwd <password>] [-os <suffix>] [-pe
       <mode>] [-pg <pageNumber>] [-pr <right>] [--proxy-host <hostname>]
       [--proxy-port <port>] [--proxy-type <type>] [-q] [-r <reason>]
       [--render-mode <mode>] [-ta <method>] [-ts <URL>] [--tsa-policy-oid
       <policyOID>] [-tscf <file>] [-tscp <password>] [-tsct <ks-type>] [-tsh
       <algorithm>] [-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.
 -c,--contact <contact>                   signer's contact details (a signature
                                          field)
 -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-acrobat6-layer-mode         disables the Acrobat 6 layer mode i.e.
                                          all signature layers will be created.
                                          Acrobat 6.0 and higher recommends that
                                          only layer n2 and n4 be present.
    --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                           This property is deprecated, use
```

```
                                        -encryption PASSWORD instead!
-ec,--encryption-certificate <file>     path to the certificate file, which is
                                        used to encrypt output PDF in case of
                                        -encryption CERTIFICATE
-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, you can also use
                                        placeholders for signature properties
                                        (${signer}, ${timestamp}, ${location},
                                        ${reason}, ${contact})
   --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
   --ocsp-server-url                default OCSP server URL, which will be
                                    used in case the signing certificate
                                    doesn't contain this information
-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)
-pe,--encryption <mode>             encryption mode for the output PDF
                                    Default value is NONE. Possible values
                                    are NONE, PASSWORD, CERTIFICATE. Use
                                    togethter with -upwd and -opwd in case
                                    of PASSWORD mode, and -ec in case of
                                    CERTIFICATE
-pg,--page <pageNumber>             page with visible signature. Default
                                    value is 1 (first page). If the
                                    provided page number is out of bounds,
                                    then the last page is used.
-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 DESCRIPTION_ONLY.
                                    Possible values are DESCRIPTION_ONLY,
                                    GRAPHIC_AND_DESCRIPTION,
                                    SIGNAME_AND_DESCRIPTION
-ta,--tsa-authentication <method>   authentication method used when
                                    contacting TSA server. Default value is
                                    NONE. Possible values are NONE,
                                    PASSWORD, CERTIFICATE
-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/HttpTspSe
                                    rver)
   --tsa-policy-oid <policyOID>     TSA policy OID which should be set to
                                    timestamp request.
-tscf,--tsa-cert-file <file>        path to keystore file, which contains
                                    private key used to authentication
                                    against TSA server, when CERTIFICATE
```

```
                                          authentication method is used
-tscp,--tsa-cert-password <password>      password used to open PKCS#12 file (see
                                          -tscf option) with a private key
-tsct,--tsa-cert-file-type <ks-type>      keystore type for TSA CERTIFICATE
                                          authentication - the default is PKCS12
-tsh,--tsa-hash-algorithm <algorithm>     hash algorithm used to in query to
                                          time-stamping server (TSA); the default
                                          is SHA-1
-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
```

10.1 Program exit codes

Code	Meaning
0	program finished without errors
1	command line is in a wrong format
2	no operation requested - e.g. no file for signing provided
3	signing of some, but not all, files failed
4	signing of all files failed

10.2 Examples

10.2.1 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

10.2.2 PKCS12 signature with encryption

```
$ java -jar JSignPdf.jar -kst PKCS12 -ksf my_certificate.pfx -ksp
    myPrivateKeystorePassword -ka cert23 -pe PASSWORD -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

10.2.3 Listing KeyStore types

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

lists keystore types

10.2.4 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.

11 Other command line tools

11.1 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:

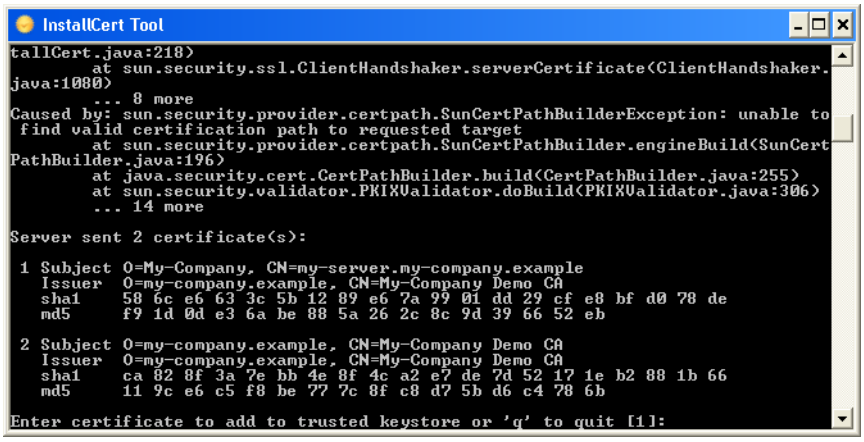
```
$java -jar InstallCert.jar
```

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.



11.2 Verifier

```
usage: java -jar Verifier.jar [file1.pdf [file2.pdf ...]] [-c
       <certificates>] [-e <folder>] [-ff <arg>] [-h] [-kf <file>]
       [-kp <password>] [-kt <keystore_type>] [-lc] [-lk] [-p
       <password>]
JSignPdf Verifier is a command line tool for verifying signed PDF
documents.
 -c,--cert <certificates>                use external semicolon
                                         separated X.509 certificate
                                         files
 -e,--extract <folder>                   extract signed PDF revisions to
                                         given folder
 -ff,--fail-fast <arg>                   flag which sets the Verifier to
                                         exit with error code on the
                                         first validation failure
 -h,--help                               print this message
 -kf,--keystore-file <file>              use given keystore file
```

```
 -kp,--keystore-password <password>    password for keystore file
                                       (look on -kf option)
 -kt,--keystore-type <keystore_type>   use keystore type with given
                                       name
 -lc,--list-certificates               list certificate aliases in a
                                       KeyStore
 -lk,--list-keystore-types             list keystore types provided by
                                       java
 -p,--password <password>              set password for opening PDF
```

11.2.1 Program exit codes

Code	Meaning
0	SIG_STAT_CODE_INFO_SIGNATURE_VALID
10	SIG_STAT_CODE_WARNING_NO_SIGNATURE
15	SIG_STAT_CODE_WARNING_ANY_WARNING
20	SIG_STAT_CODE_WARNING_NO_REVOCATION_INFO
30	SIG_STAT_CODE_WARNING_TIMESTAMP_INVALID
40	SIG_STAT_CODE_WARNING_NO_TIMESTAMP_TOKEN
50	SIG_STAT_CODE_WARNING_SIGNATURE_OCSP_INVALID
60	SIG_STAT_CODE_WARNING_CERTIFICATE_CANT_BE_VERIFIED
61	SIG_STAT_CODE_WARNING_CERTIFICATE_EXPIRED
62	SIG_STAT_CODE_WARNING_CERTIFICATE_NOT_YET_VALID
63	SIG_STAT_CODE_WARNING_CERTIFICATE_REVOKED
64	SIG_STAT_CODE_WARNING_CERTIFICATE_UNSUPPORTED_CRITICAL_EXTENSION
65	SIG_STAT_CODE_WARNING_CERTIFICATE_INVALID_STATE
66	SIG_STAT_CODE_WARNING_CERTIFICATE_PROBLEM
70	SIG_STAT_CODE_WARNING_UNSIGNED_CONTENT
101	SIG_STAT_CODE_ERROR_FILE_NOT_READABLE
102	SIG_STAT_CODE_ERROR_UNEXPECTED_PROBLEM
105	SIG_STAT_CODE_ERROR_ANY_ERROR
110	SIG_STAT_CODE_ERROR_CERTIFICATION_BROKEN
120	SIG_STAT_CODE_ERROR_REVISION_MODIFIED

11.3 SignatureCounter

```
usage: java -jar SignatureCounter.jar [file1.pdf [file2.pdf ...]] [-d]
       [-h] [-n] [-p <password>]
JSignPdf SignatureCounter is a command line tool which prints count of
signatures in given PDF document.
 -d,--debug                  enables debug output
 -h,--help                   print this message
 -n,--names                  print comma separated signature names
                             instead of the count
 -p,--password <password>    set password for opening PDF
```

11.3.1 Program exit codes

Code	Meaning
0	program finished without errors
1	command line is in a wrong format
5	unable to read the file from given path
6	counting signatures in the PDF failed