



Full credit is given to the above companies including the Operating System (OS) that this PDF file was generated!

Rocky Enterprise Linux 9.2 Manual Pages on command 'openssl-pkcs12.1ssl'

\$ man openssl-pkcs12.1ssl

OPENSSL-PKCS12(1SSL) OpenSSL OPENSSL-PKCS12(1SSL)

NAME

openssl-pkcs12 - PKCS#12 file command

SYNOPSIS

openssl pkcs12 [-help] [-passin arg] [-passout arg] [-password arg] [-twopass] [-in filename|uri] [-out filename] [-nokeys] [-nocerts] [-noout] [-legacy] [-engine id] [-provider name] [-provider-path path] [-propquery propq] [-rand files] [-writerand file]

PKCS#12 input (parsing) options: [-info] [-nomacver] [-clcerts] [-cacerts] [-aes128] [-aes192] [-aes256] [-aria128] [-aria192] [-aria256] [-camellia128] [-camellia192] [-camellia256] [-des] [-des3] [-idea] [-noenc] [-nodes]

PKCS#12 output (export) options:

[-export] [-inkey filename|uri] [-certfile filename] [-passcerts arg] [-chain] [-untrusted filename] [-CAfile file] [-no-CAfile] [-CApath dir] [-no-CApath] [-CAstore uri] [-no-CAstore] [-name name] [-caname name] [-CSP name] [-LMK] [-keyex] [-keysig] [-keypbe cipher] [-certpbe cipher] [-descert] [-macalg digest] [-iter count] [-noiter] [-nomaciter] [-maciter] [-nomac]

DESCRIPTION

This command allows PKCS#12 files (sometimes referred to as PFX files) to be created and parsed. PKCS#12 files are used by several programs including Netscape, MSIE and MS Outlook.

OPTIONS

There are a lot of options the meaning of some depends of whether a PKCS#12 file is being created or parsed. By default a PKCS#12 file is parsed. A PKCS#12 file can be created by

using the `-export` option (see below). The PKCS#12 export encryption and MAC options such as `-certpbe` and `-iter` and many further options such as `-chain` are relevant only with `-export`. Conversely, the options regarding encryption of private keys when outputting PKCS#12 input are relevant only when the `-export` option is not given.

The default encryption algorithm is AES-256-CBC with PBKDF2 for key derivation.

When encountering problems loading legacy PKCS#12 files that involve, for example, RC2-40-CBC, try using the `-legacy` option and, if needed, the `-provider-path` option.

`-help`

Print out a usage message.

`-passin arg`

The password source for the input, and for encrypting any private keys that are output. For more information about the format of `arg` see `openssl-passphrase-options(1)`.

`-passout arg`

The password source for output files.

`-password arg`

With `-export`, `-password` is equivalent to `-passout`, otherwise it is equivalent to `-passin`.

`-twopass`

Prompt for separate integrity and encryption passwords: most software always assumes these are the same so this option will render such PKCS#12 files unreadable. Cannot be used in combination with the options `-password`, `-passin` if importing from PKCS#12, or `-passout` if exporting.

`-nokeys`

No private keys will be output.

`-nocerts`

No certificates will be output.

`-noout`

This option inhibits all credentials output, and so the input is just verified.

`-legacy`

Use legacy mode of operation and automatically load the legacy provider. If OpenSSL is not installed system-wide, it is necessary to also use, for example,

`"-provider-path ./providers"` or to set the environment variable `OPENSSL_MODULES` to

point to the directory where the providers can be found.

In the legacy mode, the default algorithm for certificate encryption is RC2_CBC or 3DES_CBC depending on whether the RC2 cipher is enabled in the build. The default algorithm for private key encryption is 3DES_CBC. If the legacy option is not specified, then the legacy provider is not loaded and the default encryption algorithm for both certificates and private keys is AES_256_CBC with PBKDF2 for key derivation.

-engine id

See "Engine Options" in openssl(1). This option is deprecated.

-provider name

-provider-path path

-propquery propq

See "Provider Options" in openssl(1), provider(7), and property(7).

-rand files, -writerand file

See "Random State Options" in openssl(1) for details.

PKCS#12 input (parsing) options

-in filename|uri

This specifies the input filename or URI. Standard input is used by default. Without the -export option this must be PKCS#12 file to be parsed. For use with the -export option see the "PKCS#12 output (export) options" section.

-out filename

The filename to write certificates and private keys to, standard output by default.

They are all written in PEM format.

-info

Output additional information about the PKCS#12 file structure, algorithms used and iteration counts.

-nomacver

Don't attempt to verify the integrity MAC.

-clcerts

Only output client certificates (not CA certificates).

-cacerts

Only output CA certificates (not client certificates).

-aes128, -aes192, -aes256

Use AES to encrypt private keys before outputting.

-aria128, -aria192, -aria256

Use ARIA to encrypt private keys before outputting.

-camellia128, -camellia192, -camellia256

Use Camellia to encrypt private keys before outputting.

-des

Use DES to encrypt private keys before outputting.

-des3

Use triple DES to encrypt private keys before outputting.

-idea

Use IDEA to encrypt private keys before outputting.

-noenc

Don't encrypt private keys at all.

-nodes

This option is deprecated since OpenSSL 3.0; use -noenc instead.

PKCS#12 output (export) options

-export

This option specifies that a PKCS#12 file will be created rather than parsed.

-out filename

This specifies filename to write the PKCS#12 file to. Standard output is used by default.

-in filename|uri

This specifies the input filename or URI. Standard input is used by default. With the -export option this is a file with certificates and a key, or a URI that refers to a key accessed via an engine. The order of credentials in a file doesn't matter but one private key and its corresponding certificate should be present. If additional certificates are present they will also be included in the PKCS#12 output file.

-inkey filename|uri

The private key input for PKCS12 output. If this option is not specified then the input file (-in argument) must contain a private key. If no engine is used, the argument is taken as a file. If the -engine option is used or the URI has prefix "org.openssl.engine:" then the rest of the URI is taken as key identifier for the given engine.

-certfile filename

An input file with extra certificates to be added to the PKCS#12 output if the -export option is given.

-passcerts arg

The password source for certificate input such as -certfile and -untrusted. For more information about the format of arg see openssl-passphrase-options(1).

-chain

If this option is present then the certificate chain of the end entity certificate is built and included in the PKCS#12 output file. The end entity certificate is the first one read from the -in file if no key is given, else the first certificate matching the given key. The standard CA trust store is used for chain building, as well as any untrusted CA certificates given with the -untrusted option.

-untrusted filename

An input file of untrusted certificates that may be used for chain building, which is relevant only when a PKCS#12 file is created with the -export option and the -chain option is given as well. Any certificates that are actually part of the chain are added to the output.

-CAfile file, -no-CAfile, -CApath dir, -no-CApath, -CAstore uri, -no-CAstore

See "Trusted Certificate Options" in openssl-verification-options(1) for details.

-name friendlyname

This specifies the "friendly name" for the certificates and private key. This name is typically displayed in list boxes by software importing the file.

-caname friendlyname

This specifies the "friendly name" for other certificates. This option may be used multiple times to specify names for all certificates in the order they appear.

Netscape ignores friendly names on other certificates whereas MSIE displays them.

-CSP name

Write name as a Microsoft CSP name. The password source for the input, and for encrypting any private keys that are output. For more information about the format of arg see openssl-passphrase-options(1).

-LMK

Add the "Local Key Set" identifier to the attributes.

-keyex|-keysig

Specifies that the private key is to be used for key exchange or just signing. This

option is only interpreted by MSIE and similar MS software. Normally "export grade" software will only allow 512 bit RSA keys to be used for encryption purposes but arbitrary length keys for signing. The -keysig option marks the key for signing only. Signing only keys can be used for S/MIME signing, authenticode (ActiveX control signing) and SSL client authentication, however, due to a bug only MSIE 5.0 and later support the use of signing only keys for SSL client authentication.

-keypbe alg, -certpbe alg

These options allow the algorithm used to encrypt the private key and certificates to be selected. Any PKCS#5 v1.5 or PKCS#12 PBE algorithm name can be used (see "NOTES" section for more information). If a cipher name (as output by "openssl list -cipher-algorithms") is specified then it is used with PKCS#5 v2.0. For interoperability reasons it is advisable to only use PKCS#12 algorithms.

Special value "NONE" disables encryption of the private key and certificates.

-descert

Encrypt the certificates using triple DES. By default the private key and the certificates are encrypted using AES-256-CBC unless the '-legacy' option is used. If '-descert' is used with the '-legacy' then both, the private key and the certificates are encrypted using triple DES.

-macalg digest

Specify the MAC digest algorithm. If not included SHA1 will be used.

-iter count

This option specifies the iteration count for the encryption key and MAC. The default value is 2048.

To discourage attacks by using large dictionaries of common passwords the algorithm that derives keys from passwords can have an iteration count applied to it: this causes a certain part of the algorithm to be repeated and slows it down. The MAC is used to check the file integrity but since it will normally have the same password as the keys and certificates it could also be attacked.

-noiter, -nomaciter

By default both encryption and MAC iteration counts are set to 2048, using these options the MAC and encryption iteration counts can be set to 1, since this reduces the file security you should not use these options unless you really have to. Most software supports both MAC and encryption iteration counts. MSIE 4.0 doesn't support

MAC iteration counts so it needs the -nomaciter option.

-maciter

This option is included for compatibility with previous versions, it used to be needed to use MAC iterations counts but they are now used by default.

-nomac

Do not attempt to provide the MAC integrity. This can be useful with the FIPS provider as the PKCS12 MAC requires PKCS12KDF which is not an approved FIPS algorithm and cannot be supported by the FIPS provider.

NOTES

Although there are a large number of options most of them are very rarely used. For PKCS#12 file parsing only -in and -out need to be used for PKCS#12 file creation -export and -name are also used.

If none of the -clcerts, -cacerts or -nocerts options are present then all certificates will be output in the order they appear in the input PKCS#12 files. There is no guarantee that the first certificate present is the one corresponding to the private key. Certain software which tries to get a private key and the corresponding certificate might assume that the first certificate in the file is the one corresponding to the private key, but that may not always be the case. Using the -clcerts option will solve this problem by only outputting the certificate corresponding to the private key. If the CA certificates are required then they can be output to a separate file using the -nokeys -cacerts options to just output CA certificates.

The -keypbe and -certpbe algorithms allow the precise encryption algorithms for private keys and certificates to be specified. Normally the defaults are fine but occasionally software can't handle triple DES encrypted private keys, then the option -keypbe PBE-SHA1-RC2-40 can be used to reduce the private key encryption to 40 bit RC2. A complete description of all algorithms is contained in openssl-pkcs8(1).

Prior 1.1 release passwords containing non-ASCII characters were encoded in non-compliant manner, which limited interoperability, in first hand with Windows. But switching to standard-compliant password encoding poses problem accessing old data protected with broken encoding. For this reason even legacy encodings is attempted when reading the data. If you use PKCS#12 files in production application you are advised to convert the data, because implemented heuristic approach is not MT-safe, its sole goal is to facilitate the data upgrade with this command.

EXAMPLES

Parse a PKCS#12 file and output it to a PEM file:

```
openssl pkcs12 -in file.p12 -out file.pem
```

Output only client certificates to a file:

```
openssl pkcs12 -in file.p12 -clcerts -out file.pem
```

Don't encrypt the private key:

```
openssl pkcs12 -in file.p12 -out file.pem -noenc
```

Print some info about a PKCS#12 file:

```
openssl pkcs12 -in file.p12 -info -noout
```

Print some info about a PKCS#12 file in legacy mode:

```
openssl pkcs12 -in file.p12 -info -noout -legacy
```

Create a PKCS#12 file from a PEM file that may contain a key and certificates:

```
openssl pkcs12 -export -in file.pem -out file.p12 -name "My PSE"
```

Include some extra certificates:

```
openssl pkcs12 -export -in file.pem -out file.p12 -name "My PSE" \  
-certfile othercerts.pem
```

Export a PKCS#12 file with data from a certificate PEM file and from a further PEM file containing a key, with default algorithms as in the legacy provider:

```
openssl pkcs12 -export -in cert.pem -inkey key.pem -out file.p12 -legacy
```

SEE ALSO

openssl(1), openssl-pkcs8(1), ossl_store-file(7)

HISTORY

The `-engine` option was deprecated in OpenSSL 3.0. The `-nodes` option was deprecated in OpenSSL 3.0, too; use `-noenc` instead.

COPYRIGHT

Copyright 2000-2021 The OpenSSL Project Authors. All Rights Reserved.

Licensed under the Apache License 2.0 (the "License"). You may not use this file except in compliance with the License. You can obtain a copy in the file LICENSE in the source distribution or at <https://www.openssl.org/source/license.html>.