NAME

openssl−pkcs12, pkcs12 − PKCS#12 file utility

SYNOPSIS

openssl pkcs12 [**−help**] [**−export**] [**−chain**] [**−inkey file_or_id**] [**−certfile filename**] [**−name name**] [**−caname name**] [**−in filename**] [**−out filename**] [**−noout**] [**−nomacver**] [**−nocerts**] [**−clcerts**] [**−cacerts**] [**−nokeys**] [**−info**] [**−des | −des3 | −idea | −aes128 | −aes192 | −aes256 | −aria128 | −aria192 | −aria256 | −camellia128 | −camellia192 | −camellia256 | −nodes**] [**−noiter**] [**−maciter | −nomaciter | −nomac**] [**−twopass**] [**−descert**] [**−certpbe cipher**] [**−keypbe cipher**] [**−macalg digest**] [**−keyex**] [**−keysig**] [**−password arg**] [**−passin arg**] [**−passout arg**] [**−rand file...**] [**−writerand file**] [**−CAfile file**] [**−CApath dir**] [**−no−CAfile**] [**−no−CApath**] [**−CSP name**]

DESCRIPTION

The **pkcs12** 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).

PARSING OPTIONS

−help

Print out a usage message.

−in filename

This specifies filename of the PKCS#12 file to be parsed. Standard input is used by default.

−out filename

The filename to write certificates and private keys to, standard output by default. They are all written in PEM format.

−passin arg

The PKCS#12 file (i.e. input file) password source. For more information about the format of **arg** see the **PASS PHRASE ARGUMENTS** section in **openssl** (1).

−passout arg

Pass phrase source to encrypt any outputted private keys with. For more information about the format of **arg** see the **PASS PHRASE ARGUMENTS** section in **openssl** (1).

−password arg

With −export, −password is equivalent to −passout. Otherwise, −password is equivalent to −passin.

−noout

This option inhibits output of the keys and certificates to the output file version of the PKCS#12 file.

−clcerts

Only output client certificates (not CA certificates).

−cacerts

Only output CA certificates (not client certificates).

−nocerts

No certificates at all will be output.

−nokeys

No private keys will be output.

−info

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

−des

Use DES to encrypt private keys before outputting.

−des3
Use triple DES to encrypt private keys before outputting, this is the default.

−idea
Use IDEA to encrypt private keys before outputting.

−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.

−nodes
Don't encrypt the private keys at all.

−nomacver
Don't attempt to verify the integrity MAC before reading the file.

−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) or −passout (if exporting).

FILE CREATION 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
The filename to read certificates and private keys from, standard input by default. They must all be in PEM format. The order 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 file.

−inkey file_or_id
File to read private key from. If not present then a private key must be present in the input file. If no engine is used, the argument is taken as a file; if an engine is specified, the argument is given to the engine as a key identifier.

−name friendlyname
This specifies the ''friendly name'' for the certificate and private key. This name is typically displayed in list boxes by software importing the file.

−certfile filename
A filename to read additional certificates from.

−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.

−pass arg, **−passout arg**
The PKCS#12 file (i.e. output file) password source. For more information about the format of **arg** see the **PASS PHRASE ARGUMENTS** section in **openssl** (1).

−passin password
Pass phrase source to decrypt any input private keys with. For more information about the format of **arg** see the **PASS PHRASE ARGUMENTS** section in **openssl** (1).

−chain
If this option is present then an attempt is made to include the entire certificate chain of the user certificate. The standard CA store is used for this search. If the search fails it is considered a fatal error.

−descert
Encrypt the certificate using triple DES, this may render the PKCS#12 file unreadable by some "export grade" software. By default the private key is encrypted using triple DES and the certificate using 40 bit RC2.

−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 the **list-cipher-algorithms** command is specified then it is used with PKCS#5 v2.0. For interoperability reasons it is advisable to only use PKCS#12 algorithms.

−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.

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

−nomaciter, **−noiter**
These options affect the iteration counts on the MAC and key algorithms. Unless you wish to produce files compatible with MSIE 4.0 you should leave these options alone.

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. By default both MAC and encryption 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 key 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
Don't attempt to provide the MAC integrity.

−rand file...
A file or files containing random data used to seed the random number generator. Multiple files can be specified separated by an OS-dependent character. The separator is **;** for MS-Windows, **,** for OpenVMS, and **:** for all others.

[**−writerand file**]
Writes random data to the specified *file* upon exit. This can be used with a subsequent **−rand** flag.

−CAfile file
CA storage as a file.

−CApath dir
CA storage as a directory. This directory must be a standard certificate directory: that is a hash of each subject name (using **x509 −hash**) should be linked to each certificate.

−no−CAfile
Do not load the trusted CA certificates from the default file location.

−no−CApath
Do not load the trusted CA certificates from the default directory location.

−CSP name
Write **name** as a Microsoft CSP name.

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 requires a private key and certificate and assumes the first certificate in the file is the one corresponding to the private key: this 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 the **pkcs8** manual page.

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 utility.

EXAMPLES

Parse a PKCS#12 file and output it to a 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 −nodes
```

Print some info about a PKCS#12 file:

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

Create a PKCS#12 file:

```
openssl pkcs12 −export −in file.pem −out file.p12 −name "My Certificate"
```

Include some extra certificates:

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

SEE ALSO

pkcs8 (1)

COPYRIGHT