NAME

ca - sample minimal CA application

SYNOPSIS

openssl ca [**-verbose**] [**-config filename**] [**-name section**] [**-gencrl**] [**-revoke file**] [**-status serial**] [**-updatedb**] [**-crl_reason reason**] [**-crl_hold instruction**] [**-crl_compromise time**] [**-crl_CA_compromise time**] [**-crldays days**] [**-crlhours hours**] [**-crlexts section**] [**-startdate date**] [**-enddate date**] [**-days arg**] [**-md arg**] [**-policy arg**] [**-keyfile arg**] [**-keyform PEM|DER**] [**-key arg**] [**-passin arg**] [**-cert file**] [**-selfsign**] [**-in file**] [**-out file**] [**-notext**] [**-outdir dir**] [**-infiles**] [**-spkac file**] [**-ss_cert file**] [**-preserveDN**] [**-noemailDN**] [**-batch**] [**-msie_hack**] [**-extensions section**] [**-extfile section**] [**-engine id**] [**-subj arg**] [**-utf8**] [**-multivalue-rdn**]

DESCRIPTION

The **ca** command is a minimal CA application. It can be used to sign certificate requests in a variety of forms and generate CRLs it also maintains a text database of issued certificates and their status.

The options descriptions will be divided into each purpose.

CA OPTIONS

-config filename
: specifies the configuration file to use.

-name section
: specifies the configuration file section to use (overrides **default_ca** in the **ca** section).

-in filename
: an input filename containing a single certificate request to be signed by the CA.

-ss_cert filename
: a single self signed certificate to be signed by the CA.

-spkac filename
: a file containing a single Netscape signed public key and challenge and additional field values to be signed by the CA. See the **SPKAC FORMAT** section for information on the required input and output format.

-infiles
: if present this should be the last option, all subsequent arguments are assumed to the the names of files containing certificate requests.

-out filename
: the output file to output certificates to. The default is standard output. The certificate details will also be printed out to this file in PEM format (except that **-spkac** outputs DER format).

-outdir directory
: the directory to output certificates to. The certificate will be written to a filename consisting of the serial number in hex with ‘‘.pem’’ appended.

-cert
: the CA certificate file.

-keyfile filename
: the private key to sign requests with.

-keyform PEM|DER
: the format of the data in the private key file. The default is PEM.

-key password
: the password used to encrypt the private key. Since on some systems the command line arguments are visible (e.g. Unix with the ’ps’ utility) this option should be used with caution.

-selfsign
indicates the issued certificates are to be signed with the key the certificate requests were signed with (given with **-keyfile**). Cerificate requests signed with a different key are ignored. If **-spkac**, **-ss_cert** or **-gencrl** are given, **-selfsign** is ignored.

A consequence of using **-selfsign** is that the self-signed certificate appears among the entries in the certificate database (see the configuration option **database**), and uses the same serial number counter as all other certificates sign with the self-signed certificate.

-passin arg
the key password source. For more information about the format of **arg** see the **PASS PHRASE ARGUMENTS** section in *openssl(1)*.

-verbose
this prints extra details about the operations being performed.

-notext
don't output the text form of a certificate to the output file.

-startdate date
this allows the start date to be explicitly set. The format of the date is YYMMDDHHMMSSZ (the same as an ASN1 UTCTime structure).

-enddate date
this allows the expiry date to be explicitly set. The format of the date is YYMMDDHHMMSSZ (the same as an ASN1 UTCTime structure).

-days arg
the number of days to certify the certificate for.

-md alg
the message digest to use. Possible values include md5, sha1 and mdc2. This option also applies to CRLs.

-policy arg
this option defines the CA "policy" to use. This is a section in the configuration file which decides which fields should be mandatory or match the CA certificate. Check out the **POLICY FORMAT** section for more information.

-msie_hack
this is a legacy option to make **ca** work with very old versions of the IE certificate enrollment control "certenr3". It used UniversalStrings for almost everything. Since the old control has various security bugs its use is strongly discouraged. The newer control "Xenroll" does not need this option.

-preserveDN
Normally the DN order of a certificate is the same as the order of the fields in the relevant policy section. When this option is set the order is the same as the request. This is largely for compatibility with the older IE enrollment control which would only accept certificates if their DNs match the order of the request. This is not needed for Xenroll.

-noemailDN
The DN of a certificate can contain the EMAIL field if present in the request DN, however it is good policy just having the e-mail set into the altName extension of the certificate. When this option is set the EMAIL field is removed from the certificate' subject and set only in the, eventually present, extensions. The **email_in_dn** keyword can be used in the configuration file to enable this behaviour.

-batch
this sets the batch mode. In this mode no questions will be asked and all certificates will be certified automatically.

-extensions section
the section of the configuration file containing certificate extensions to be added when a certificate is issued (defaults to **x509_extensions** unless the **-extfile** option is used). If no extension section is present then, a V1 certificate is created. If the extension section is present (even if it is empty), then a V3 certificate is created. See the:w *x509v3_config(5)* manual page for details of the extension section format.

-extfile file
an additional configuration file to read certificate extensions from (using the default section unless the **-extensions** option is also used).

-engine id
specifying an engine (by its unique **id** string) will cause **ca** to attempt to obtain a functional reference to the specified engine, thus initialising it if needed. The engine will then be set as the default for all available algorithms.

-subj arg
supersedes subject name given in the request. The arg must be formatted as */type0=value0/type1=value1/type2=...*, characters may be escaped by \ (backslash), no spaces are skipped.

-utf8
this option causes field values to be interpreted as UTF8 strings, by default they are interpreted as ASCII. This means that the field values, whether prompted from a terminal or obtained from a configuration file, must be valid UTF8 strings.

-multivalue-rdn
this option causes the -subj argument to be interpretedt with full support for multivalued RDNs. Example:

/DC=org/DC=OpenSSL/DC=users/UID=123456+CN=John Doe

If -multi-rdn is not used then the UID value is *123456+CN=John Doe*.

CRL OPTIONS

-gencrl
this option generates a CRL based on information in the index file.

-crldays num
the number of days before the next CRL is due. That is the days from now to place in the CRL nextUpdate field.

-crlhours num
the number of hours before the next CRL is due.

-revoke filename
a filename containing a certificate to revoke.

-status serial
displays the revocation status of the certificate with the specified serial number and exits.

-updatedb
Updates the database index to purge expired certificates.

-crl_reason reason
revocation reason, where **reason** is one of: **unspecified**, **keyCompromise**, **CACompromise**, **affiliationChanged**, **superseded**, **cessationOfOperation**, **certificateHold** or **removeFromCRL**. The matching of **reason** is case insensitive. Setting any revocation reason will make the CRL v2.

In practive **removeFromCRL** is not particularly useful because it is only used in delta CRLs which are not currently implemented.

-crl_hold instruction
This sets the CRL revocation reason code to **certificateHold** and the hold instruction to **instruction** which must be an OID. Although any OID can be used only **holdInstructionNone** (the use of which is discouraged by RFC2459) **holdInstructionCallIssuer** or **holdInstructionReject** will normally be used.

-crl_compromise time
This sets the revocation reason to **keyCompromise** and the compromise time to **time**. **time** should be in GeneralizedTime format that is **YYYYMMDDHHMMSSZ**.

-crl_CA_compromise time
This is the same as **crl_compromise** except the revocation reason is set to **CACompromise**.

-crlexts section
the section of the configuration file containing CRL extensions to include. If no CRL extension section is present then a V1 CRL is created, if the CRL extension section is present (even if it is empty) then a V2 CRL is created. The CRL extensions specified are CRL extensions and **not** CRL entry extensions. It should be noted that some software (for example Netscape) can't handle V2 CRLs. See *x509v3_config(5)* manual page for details of the extension section format.

CONFIGURATION FILE OPTIONS

The section of the configuration file containing options for **ca** is found as follows: If the **-name** command line option is used, then it names the section to be used. Otherwise the section to be used must be named in the **default_ca** option of the **ca** section of the configuration file (or in the default section of the configuration file). Besides **default_ca**, the following options are read directly from the **ca** section: RANDFILE preserve msie_hack With the exception of **RANDFILE**, this is probably a bug and may change in future releases.

Many of the configuration file options are identical to command line options. Where the option is present in the configuration file and the command line the command line value is used. Where an option is described as mandatory then it must be present in the configuration file or the command line equivalent (if any) used.

oid_file
This specifies a file containing additional **OBJECT IDENTIFIERS**. Each line of the file should consist of the numerical form of the object identifier followed by white space then the short name followed by white space and finally the long name.

oid_section
This specifies a section in the configuration file containing extra object identifiers. Each line should consist of the short name of the object identifier followed by = and the numerical form. The short and long names are the same when this option is used.

new_certs_dir
the same as the **-outdir** command line option. It specifies the directory where new certificates will be placed. Mandatory.

certificate
the same as **-cert**. It gives the file containing the CA certificate. Mandatory.

private_key
same as the **-keyfile** option. The file containing the CA private key. Mandatory.

RANDFILE
a file used to read and write random number seed information, or an EGD socket (see *RAND_egd(3)*).

default_days
the same as the **-days** option. The number of days to certify a certificate for.

default_startdate
the same as the **-startdate** option. The start date to certify a certificate for. If not set the current time is used.

default_enddate
the same as the **-enddate** option. Either this option or **default_days** (or the command line equivalents) must be present.

default_crl_hours default_crl_days
the same as the **-crlhours** and the **-crldays** options. These will only be used if neither command line option is present. At least one of these must be present to generate a CRL.

default_md
the same as the **-md** option. The message digest to use. Mandatory.

database
the text database file to use. Mandatory. This file must be present though initially it will be empty.

unique_subject
if the value **yes** is given, the valid certificate entries in the database must have unique subjects. if the value **no** is given, several valid certificate entries may have the exact same subject. The default value is **yes**, to be compatible with older (pre 0.9.8) versions of OpenSSL. However, to make CA certificate roll-over easier, it's recommended to use the value **no**, especially if combined with the **-selfsign** command line option.

serial
a text file containing the next serial number to use in hex. Mandatory. This file must be present and contain a valid serial number.

crlnumber
a text file containing the next CRL number to use in hex. The crl number will be inserted in the CRLs only if this file exists. If this file is present, it must contain a valid CRL number.

x509_extensions
the same as **-extensions**.

crl_extensions
the same as **-crlexts**.

preserve
the same as **-preserveDN**

email_in_dn
the same as **-noemailDN**. If you want the EMAIL field to be removed from the DN of the certificate simply set this to 'no'. If not present the default is to allow for the EMAIL filed in the certificate's DN.

msie_hack
the same as **-msie_hack**

policy
the same as **-policy**. Mandatory. See the **POLICY FORMAT** section for more information.

name_opt, **cert_opt**
these options allow the format used to display the certificate details when asking the user to confirm signing. All the options supported by the **x509** utilities **-nameopt** and **-certopt** switches can be used here, except the **no_signame** and **no_sigdump** are permanently set and cannot be disabled (this is because the certificate signature cannot be displayed because the certificate has not been signed at this point).

For convenience the values **ca_default** are accepted by both to produce a reasonable output.

If neither option is present the format used in earlier versions of OpenSSL is used. Use of the

old format is **strongly** discouraged because it only displays fields mentioned in the **policy** section, mishandles multicharacter string types and does not display extensions.

copy_extensions
determines how extensions in certificate requests should be handled. If set to **none** or this option is not present then extensions are ignored and not copied to the certificate. If set to **copy** then any extensions present in the request that are not already present are copied to the certificate. If set to **copyall** then all extensions in the request are copied to the certificate: if the extension is already present in the certificate it is deleted first. See the **WARNINGS** section before using this option.

The main use of this option is to allow a certificate request to supply values for certain extensions such as subjectAltName.

POLICY FORMAT

The policy section consists of a set of variables corresponding to certificate DN fields. If the value is "match" then the field value must match the same field in the CA certificate. If the value is "supplied" then it must be present. If the value is "optional" then it may be present. Any fields not mentioned in the policy section are silently deleted, unless the **-preserveDN** option is set but this can be regarded more of a quirk than intended behaviour.

SPKAC FORMAT

The input to the **-spkac** command line option is a Netscape signed public key and challenge. This will usually come from the **KEYGEN** tag in an HTML form to create a new private key. It is however possible to create SPKACs using the **spkac** utility.

The file should contain the variable SPKAC set to the value of the SPKAC and also the required DN components as name value pairs. If you need to include the same component twice then it can be preceded by a number and a '.'.

When processing SPKAC format, the output is DER if the **-out** flag is used, but PEM format if sending to stdout or the **-outdir** flag is used.

EXAMPLES

Note: these examples assume that the **ca** directory structure is already set up and the relevant files already exist. This usually involves creating a CA certificate and private key with **req**, a serial number file and an empty index file and placing them in the relevant directories.

To use the sample configuration file below the directories demoCA, demoCA/private and demoCA/newcerts would be created. The CA certificate would be copied to demoCA/cacert.pem and its private key to demoCA/private/cakey.pem. A file demoCA/serial would be created containing for example "01" and the empty index file demoCA/index.txt.

Sign a certificate request:

```
openssl ca -in req.pem -out newcert.pem
```

Sign a certificate request, using CA extensions:

```
openssl ca -in req.pem -extensions v3_ca -out newcert.pem
```

Generate a CRL

```
openssl ca -gencrl -out crl.pem
```

Sign several requests:

```
openssl ca -infiles req1.pem req2.pem req3.pem
```

Certify a Netscape SPKAC:

```
openssl ca -spkac spkac.txt
```

A sample SPKAC file (the SPKAC line has been truncated for clarity):

```
SPKAC=MIG0MGAwXDANBgkqhkiG9w0BAQEFAANLADBIAkEAn7PDhCeV/xIxUg8V70YRxK2A5
CN=Steve Test
emailAddress=steve@openssl.org
0.OU=OpenSSL Group
1.OU=Another Group
```

A sample configuration file with the relevant sections for **ca**:

```
[ ca ]
default_ca = CA_default # The default ca section

[ CA_default ]

dir = ./demoCA # top dir
database = $dir/index.txt # index file.
new_certs_dir = $dir/newcerts # new certs dir

certificate = $dir/cacert.pem # The CA cert
serial = $dir/serial # serial no file
private_key = $dir/private/cakey.pem# CA private key
RANDFILE = $dir/private/.rand # random number file

default_days = 365 # how long to certify for
default_crl_days= 30 # how long before next CRL
default_md = md5 # md to use

policy = policy_any # default policy
email_in_dn = no # Don't add the email into cert DN

name_opt = ca_default # Subject name display option
cert_opt = ca_default # Certificate display option
copy_extensions = none # Don't copy extensions from request

[ policy_any ]
countryName = supplied
stateOrProvinceName = optional
organizationName = optional
organizationalUnitName = optional
commonName = supplied
emailAddress = optional
```

FILES

Note: the location of all files can change either by compile time options, configuration file entries, environment variables or command line options. The values below reflect the default values.

```
/usr/local/ssl/lib/openssl.cnf - master configuration file
./demoCA - main CA directory
./demoCA/cacert.pem - CA certificate
./demoCA/private/cakey.pem - CA private key
./demoCA/serial - CA serial number file
./demoCA/serial.old - CA serial number backup file
./demoCA/index.txt - CA text database file
./demoCA/index.txt.old - CA text database backup file
./demoCA/certs - certificate output file
./demoCA/.rnd - CA random seed information
```

ENVIRONMENT VARIABLES

OPENSSL_CONF reflects the location of master configuration file it can be overridden by the **-config** command line option.

RESTRICTIONS

The text database index file is a critical part of the process and if corrupted it can be difficult to fix. It is theoretically possible to rebuild the index file from all the issued certificates and a current CRL: however there is no option to do this.

V2 CRL features like delta CRLs are not currently supported.

Although several requests can be input and handled at once it is only possible to include one SPKAC or self signed certificate.

BUGS

The use of an in memory text database can cause problems when large numbers of certificates are present because, as the name implies the database has to be kept in memory.

The **ca** command really needs rewriting or the required functionality exposed at either a command or interface level so a more friendly utility (perl script or GUI) can handle things properly. The scripts **CA.sh** and **CA.pl** help a little but not very much.

Any fields in a request that are not present in a policy are silently deleted. This does not happen if the **-preserveDN** option is used. To enforce the absence of the EMAIL field within the DN, as suggested by RFCs, regardless the contents of the request' subject the **-noemailDN** option can be used. The behaviour should be more friendly and configurable.

Cancelling some commands by refusing to certify a certificate can create an empty file.

WARNINGS

The **ca** command is quirky and at times downright unfriendly.

The **ca** utility was originally meant as an example of how to do things in a CA. It was not supposed to be used as a full blown CA itself: nevertheless some people are using it for this purpose.

The **ca** command is effectively a single user command: no locking is done on the various files and attempts to run more than one **ca** command on the same database can have unpredictable results.

The **copy_extensions** option should be used with caution. If care is not taken then it can be a security risk. For example if a certificate request contains a basicConstraints extension with CA:TRUE and the **copy_extensions** value is set to **copyall** and the user does not spot this when the certificate is displayed then this will hand the requestor a valid CA certificate.

This situation can be avoided by setting **copy_extensions** to **copy** and including basicConstraints with CA:FALSE in the configuration file. Then if the request contains a basicConstraints extension it will be ignored.

It is advisable to also include values for other extensions such as **keyUsage** to prevent a request supplying its own values.

Additional restrictions can be placed on the CA certificate itself. For example if the CA certificate has:

```
basicConstraints = CA:TRUE, pathlen:0
```

then even if a certificate is issued with CA:TRUE it will not be valid.

SEE ALSO

req(1), *spkac(1)*, *x509(1)*, *CA.pl*(1), *config(5)*, *x509v3_config(5)*