NAME

certutil − Manage keys and certificate in both NSS databases and other NSS tokens

SYNOPSIS

certutil [*options*] [[*arguments*]]

STATUS

This documentation is still work in progress. Please contribute to the initial review in **Mozilla NSS bug 836477**[1]

DESCRIPTION

The Certificate Database Tool, **certutil**, is a command−line utility that can create and modify certificate and key databases. It can specifically list, generate, modify, or delete certificates, create or change the password, generate new public and private key pairs, display the contents of the key database, or delete key pairs within the key database.

Certificate issuance, part of the key and certificate management process, requires that keys and certificates be created in the key database. This document discusses certificate and key database management. For information on the security module database management, see the **modutil** manpage.

COMMAND OPTIONS AND ARGUMENTS

Running **certutil** always requires one and only one command option to specify the type of certificate operation. Each command option may take zero or more arguments. The command option **−H** will list all the command options and their relevant arguments.

Command Options

−A
: Add an existing certificate to a certificate database. The certificate database should already exist; if one is not present, this command option will initialize one by default.

−B
: Run a series of commands from the specified batch file. This requires the **−i** argument.

−C
: Create a new binary certificate file from a binary certificate request file. Use the **−i** argument to specify the certificate request file. If this argument is not used, **certutil** prompts for a filename.

−D
: Delete a certificate from the certificate database.

−−rename
: Change the database nickname of a certificate.

−E
: Add an email certificate to the certificate database.

−F
: Delete a private key and the associated certificate from a database. Specify the key to delete with the −n argument or the −k argument. Specify the database from which to delete the key with the **−d** argument.

  Some smart cards do not let you remove a public key you have generated. In such a case, only the private key is deleted from the key pair.

−G
: Generate a new public and private key pair within a key database. The key database should already exist; if one is not present, this command option will initialize one by default. Some smart cards can store only one key pair. If you create a new key pair for such a card, the previous pair is overwritten.

−H
: Display a list of the command options and arguments.

−K
: List the key ID of keys in the key database. A key ID is the modulus of the RSA key or the publicValue of the DSA key. IDs are displayed in hexadecimal ("0x" is not shown).

−L
: List all the certificates, or display information about a named certificate, in a certificate database. Use the −h tokenname argument to specify the certificate database on a particular hardware or software token.

−M
: Modify a certificate's trust attributes using the values of the −t argument.

−N
: Create new certificate and key databases.

−O
: Print the certificate chain.

−R
: Create a certificate request file that can be submitted to a Certificate Authority (CA) for processing into a finished certificate. Output defaults to standard out unless you use −o output−file argument. Use the −a argument to specify ASCII output.

−S
: Create an individual certificate and add it to a certificate database.

−T
: Reset the key database or token.

−U
: List all available modules or print a single named module.

−V
: Check the validity of a certificate and its attributes.

−W
: Change the password to a key database.

−−merge
: Merge two databases into one.

−−upgrade−merge
: Upgrade an old database and merge it into a new database. This is used to migrate legacy NSS databases (cert8.db and key3.db) into the newer SQLite databases (cert9.db and key4.db).

Arguments

Arguments modify a command option and are usually lower case, numbers, or symbols.

−a
: Use ASCII format or allow the use of ASCII format for input or output. This formatting follows RFC 1113. For certificate requests, ASCII output defaults to standard output unless redirected.

−−simple−self−signed
: When printing the certificate chain, don't search for a chain if issuer name equals to subject name.

−b validity−time
: Specify a time at which a certificate is required to be valid. Use when checking certificate validity with the **−V** option. The format of the *validity−time* argument is *YYMMDDHHMMSS[+HHMM|−HHMM|Z]*, which allows offsets to be set relative to the validity end time. Specifying seconds (*SS*) is optional. When specifying an explicit time, use a Z at the end of the term, *YYMMDDHHMMSSZ*, to close it. When specifying an offset time, use *YYMMDDHHMMSS+HHMM* or *YYMMDDHHMMSS−HHMM* for adding or subtracting time, respectively.

If this option is not used, the validity check defaults to the current system time.

−c issuer
Identify the certificate of the CA from which a new certificate will derive its authenticity. Use the exact nickname or alias of the CA certificate, or use the CA's email address. Bracket the issuer string with quotation marks if it contains spaces.

−d [prefix]directory
Specify the database directory containing the certificate and key database files.

certutil supports two types of databases: the legacy security databases (cert8.db, key3.db, and secmod.db) and new SQLite databases (cert9.db, key4.db, and pkcs11.txt).

NSS recognizes the following prefixes:

- **sql:** requests the newer database
- **dbm:** requests the legacy database

If no prefix is specified the default type is retrieved from NSS_DEFAULT_DB_TYPE. If NSS_DEFAULT_DB_TYPE is not set then **sql:** is the default.

−−dump−ext−val OID
For single cert, print binary DER encoding of extension OID.

−e
Check a certificate's signature during the process of validating a certificate.

−−email email−address
Specify the email address of a certificate to list. Used with the −L command option.

−−extGeneric OID:critical−flag:filename[,OID:critical−flag:filename]...
Add one or multiple extensions that certutil cannot encode yet, by loading their encodings from external files.

- OID (example): 1.2.3.4
- critical−flag: critical or not−critical
- filename: full path to a file containing an encoded extension

−f password−file
Specify a file that will automatically supply the password to include in a certificate or to access a certificate database. This is a plain−text file containing one password. Be sure to prevent unauthorized access to this file.

−g keysize
Set a key size to use when generating new public and private key pairs. The minimum is 512 bits and the maximum is 16384 bits. The default is 2048 bits. Any size between the minimum and maximum is allowed.

−h tokenname
Specify the name of a token to use or act on. If not specified the default token is the internal database slot.

The name can also be a PKCS #11 URI. For example, the NSS internal certificate store can be unambiguously specified as "pkcs11:token=NSS%20Certificate%20DB". For details about the format, see RFC 7512.

−i input_file
Pass an input file to the command. Depending on the command option, an input file can be a specific certificate, a certificate request file, or a batch file of commands.

−k key−type−or−id

Specify the type or specific ID of a key.

The valid key type options are rsa, dsa, ec, or all. The default value is rsa. Specifying the type of key can avoid mistakes caused by duplicate nicknames. Giving a key type generates a new key pair; giving the ID of an existing key reuses that key pair (which is required to renew certificates).

−l
Display detailed information when validating a certificate with the −V option.

−m serial−number
Assign a unique serial number to a certificate being created. This operation should be performed by a CA. If no serial number is provided a default serial number is made from the current time. Serial numbers are limited to integers

−n nickname
Specify the nickname of a certificate or key to list, create, add to a database, modify, or validate. Bracket the nickname string with quotation marks if it contains spaces.

The nickname can also be a PKCS #11 URI. For example, if you have a certificate named "my−server−cert" on the internal certificate store, it can be unambiguously specified as "pkcs11:token=NSS%20Certificate%20DB;object=my−server−cert". For details about the format, see RFC 7512.

−o output−file
Specify the output file name for new certificates or binary certificate requests. Bracket the output−file string with quotation marks if it contains spaces. If this argument is not used the output destination defaults to standard output.

−P dbPrefix
Specify the prefix used on the certificate and key database file. This argument is provided to support legacy servers. Most applications do not use a database prefix.

−p phone
Specify a contact telephone number to include in new certificates or certificate requests. Bracket this string with quotation marks if it contains spaces.

−q pqgfile or curve−name
Read an alternate PQG value from the specified file when generating DSA key pairs. If this argument is not used, **certutil** generates its own PQG value. PQG files are created with a separate DSA utility.

Elliptic curve name is one of the ones from nistp256, nistp384, nistp521, curve25519.

If a token is available that supports more curves, the foolowing curves are supported as well: sect163k1, nistk163, sect163r1, sect163r2, nistb163, sect193r1, sect193r2, sect233k1, nistk233, sect233r1, nistb233, sect239k1, sect283k1, nistk283, sect283r1, nistb283, sect409k1, nistk409, sect409r1, nistb409, sect571k1, nistk571, sect571r1, nistb571, secp160k1, secp160r1, secp160r2, secp192k1, secp192r1, nistp192, secp224k1, secp224r1, nistp224, secp256k1, secp256r1, secp384r1, secp521r1, prime192v1, prime192v2, prime192v3, prime239v1, prime239v2, prime239v3, c2pnb163v1, c2pnb163v2, c2pnb163v3, c2pnb176v1, c2tnb191v1, c2tnb191v2, c2tnb191v3, c2pnb208w1, c2tnb239v1, c2tnb239v2, c2tnb239v3, c2pnb272w1, c2pnb304w1, c2tnb359w1, c2pnb368w1, c2tnb431r1, secp112r1, secp112r2, secp128r1, secp128r2, sect113r1, sect113r2, sect131r1, sect131r2

−r
Display a certificate's binary DER encoding when listing information about that certificate with the −L option.

−s subject
Identify a particular certificate owner for new certificates or certificate requests. Bracket this string with quotation marks if it contains spaces. The subject identification format follows RFC #1485.

−t trustargs
: Specify the trust attributes to modify in an existing certificate or to apply to a certificate when creating it or adding it to a database. There are three available trust categories for each certificate, expressed in the order *SSL, email, object signing* for each trust setting. In each category position, use none, any, or all of the attribute codes:

  - **p** − Valid peer
  - **P** − Trusted peer (implies p)
  - **c** − Valid CA
  - **C** − Trusted CA (implies c)
  - **T** − trusted CA for client authentication (ssl server only)

  The attribute codes for the categories are separated by commas, and the entire set of attributes enclosed by quotation marks. For example:

  −t "TC,C,T"

  Use the −L option to see a list of the current certificates and trust attributes in a certificate database.

  Note that the output of the −L option may include "u" flag, which means that there is a private key associated with the certificate. It is a dynamic flag and you cannot set it with certutil.

−u certusage
: Specify a usage context to apply when validating a certificate with the −V option.

  The contexts are the following:

  - **C** (as an SSL client)
  - **V** (as an SSL server)
  - **L** (as an SSL CA)
  - **A** (as Any CA)
  - **Y** (Verify CA)
  - **S** (as an email signer)
  - **R** (as an email recipient)
  - **O** (as an OCSP status responder)
  - **J** (as an object signer)
  - **I** (as an IPSEC user)

−v valid−months
: Set the number of months a new certificate will be valid. The validity period begins at the current system time unless an offset is added or subtracted with the **−w** option. If this argument is not used, the default validity period is three months.

−w offset−months
: Set an offset from the current system time, in months, for the beginning of a certificate's validity period. Use when creating the certificate or adding it to a database. Express the offset in integers, using a minus sign (−) to indicate a negative offset. If this argument is not used, the validity period begins at the current system time. The length of the validity period is set with the −v argument.

−X
: Force the key and certificate database to open in read−write mode. This is used with the **−U** and **−L** command options.

−x
: Use **certutil** to generate the signature for a certificate being created or added to a database, rather than obtaining a signature from a separate CA.

−y exp
: Set an alternate exponent value to use in generating a new RSA public key for the database, instead of the default value of 65537. The available alternate values are 3 and 17.

−−pss
: Restrict the generated certificate (with the **−S** option) or certificate request (with the **−R** option) to be used with the RSA−PSS signature scheme. This only works when the private key of the certificate or certificate request is RSA.

−−pss−sign
: Sign the generated certificate with the RSA−PSS signature scheme (with the **−C** or **−S** option). This only works when the private key of the signer's certificate is RSA. If the signer's certificate is restricted to RSA−PSS, it is not necessary to specify this option.

−z noise−file
: Read a seed value from the specified file to generate a new private and public key pair. This argument makes it possible to use hardware−generated seed values or manually create a value from the keyboard. The minimum file size is 20 bytes.

−Z hashAlg
: Specify the hash algorithm to use with the −C, −S or −R command options. Possible keywords:

  - MD2
  - MD4
  - MD5
  - SHA1
  - SHA224
  - SHA256
  - SHA384
  - SHA512

−0 SSO_password
: Set a site security officer password on a token.

−1 | −−keyUsage keyword,keyword
: Set an X.509 V3 Certificate Type Extension in the certificate. There are several available keywords:

  - digitalSignature
  - nonRepudiation
  - keyEncipherment
  - dataEncipherment
  - keyAgreement
  - certSigning
  - crlSigning
  - critical

−2
: Add a basic constraint extension to a certificate that is being created or added to a database. This extension supports the certificate chain verification process. **certutil** prompts for the certificate constraint extension to select.

X.509 certificate extensions are described in RFC 5280.

−3

Add an authority key ID extension to a certificate that is being created or added to a database. This extension supports the identification of a particular certificate, from among multiple certificates associated with one subject name, as the correct issuer of a certificate. The Certificate Database Tool will prompt you to select the authority key ID extension.

X.509 certificate extensions are described in RFC 5280.

−4

Add a CRL distribution point extension to a certificate that is being created or added to a database. This extension identifies the URL of a certificate's associated certificate revocation list (CRL). **certutil** prompts for the URL.

X.509 certificate extensions are described in RFC 5280.

−5 | −−nsCertType keyword,keyword

Add an X.509 V3 certificate type extension to a certificate that is being created or added to the database. There are several available keywords:

- sslClient
- sslServer
- smime
- objectSigning
- sslCA
- smimeCA
- objectSigningCA
- critical

X.509 certificate extensions are described in RFC 5280.

−6 | −−extKeyUsage keyword,keyword

Add an extended key usage extension to a certificate that is being created or added to the database. Several keywords are available:

- serverAuth
- clientAuth
- codeSigning
- emailProtection
- timeStamp
- ocspResponder
- stepUp
- msTrustListSign
- critical
- x509Any
- ipsecIKE
- ipsecIKEEnd

- ipsecIKEIntermediate
- ipsecEnd
- ipsecTunnel
- ipsecUser

X.509 certificate extensions are described in RFC 5280.

−7 emailAddrs
Add a comma−separated list of email addresses to the subject alternative name extension of a certificate or certificate request that is being created or added to the database. Subject alternative name extensions are described in Section 4.2.1.7 of RFC 3280.

−8 dns−names
Add a comma−separated list of DNS names to the subject alternative name extension of a certificate or certificate request that is being created or added to the database. Subject alternative name extensions are described in Section 4.2.1.7 of RFC 3280.

−−extAIA
Add the Authority Information Access extension to the certificate. X.509 certificate extensions are described in RFC 5280.

−−extSIA
Add the Subject Information Access extension to the certificate. X.509 certificate extensions are described in RFC 5280.

−−extCP
Add the Certificate Policies extension to the certificate. X.509 certificate extensions are described in RFC 5280.

−−extPM
Add the Policy Mappings extension to the certificate. X.509 certificate extensions are described in RFC 5280.

−−extPC
Add the Policy Constraints extension to the certificate. X.509 certificate extensions are described in RFC 5280.

−−extIA
Add the Inhibit Any Policy Access extension to the certificate. X.509 certificate extensions are described in RFC 5280.

−−extSKID
Add the Subject Key ID extension to the certificate. X.509 certificate extensions are described in RFC 5280.

−−extNC
Add a Name Constraint extension to the certificate. X.509 certificate extensions are described in RFC 5280.

−−extSAN type:name[,type:name]...
Create a Subject Alt Name extension with one or multiple names.

−type: directory, dn, dns, edi, ediparty, email, ip, ipaddr, other, registerid, rfc822, uri, x400, x400addr

−−empty−password
Use empty password when creating new certificate database with −N.

−−keyAttrFlags attrflags
PKCS #11 key Attributes. Comma separated list of key attribute flags, selected from the following list of choices: {token | session} {public | private} {sensitive | insensitive} {modifiable | unmodifiable} {extractable | unextractable}

−−keyOpFlagsOn opflags, −−keyOpFlagsOff opflags
: PKCS #11 key Operation Flags. Comma separated list of one or more of the following: {token | session} {public | private} {sensitive | insensitive} {modifiable | unmodifiable} {extractable | unextractable}

−−new−n nickname
: A new nickname, used when renaming a certificate.

−−source−dir certdir
: Identify the certificate database directory to upgrade.

−−source−prefix certdir
: Give the prefix of the certificate and key databases to upgrade.

−−upgrade−id uniqueID
: Give the unique ID of the database to upgrade.

−−upgrade−token−name name
: Set the name of the token to use while it is being upgraded.

−@ pwfile
: Give the name of a password file to use for the database being upgraded.

USAGE AND EXAMPLES

Most of the command options in the examples listed here have more arguments available. The arguments included in these examples are the most common ones or are used to illustrate a specific scenario. Use the **−H** option to show the complete list of arguments for each command option.

Creating New Security Databases

Certificates, keys, and security modules related to managing certificates are stored in three related databases:

- cert8.db or cert9.db
- key3.db or key4.db
- secmod.db or pkcs11.txt

These databases must be created before certificates or keys can be generated.

certutil −N −d directory

Creating a Certificate Request

A certificate request contains most or all of the information that is used to generate the final certificate. This request is submitted separately to a certificate authority and is then approved by some mechanism (automatically or by human review). Once the request is approved, then the certificate is generated.

$ certutil −R −k key−type−or−id [−q pqgfile|curve−name] −g key−size −s subject [−h tokenname] −d directory [−p phone] [−

The **−R** command options requires four arguments:

- **−k** to specify either the key type to generate or, when renewing a certificate, the existing key pair to use
- **−g** to set the keysize of the key to generate
- **−s** to set the subject name of the certificate
- **−d** to give the security database directory

The new certificate request can be output in ASCII format (**−a**) or can be written to a specified file (**−o**).

For example:

$ certutil −R −k rsa −g 1024 −s "CN=John Smith,O=Example Corp,L=Mountain View,ST=California,C=US" −d $HOME/nss

Generating key. This may take a few moments...

Creating a Certificate

A valid certificate must be issued by a trusted CA. This can be done by specifying a CA certificate (**−c**) that is stored in the certificate database. If a CA key pair is not available, you can create a self−signed certificate using the **−x** argument with the **−S** command option.

$ certutil −S −k rsa|dsa|ec −n certname −s subject [−c issuer |−x] −t trustargs −d directory [−m serial−number] [−v valid−mont

The series of numbers and **−−ext*** options set certificate extensions that can be added to the certificate when it is generated by the CA. Interactive prompts will result.

For example, this creates a self−signed certificate:

$ certutil −S −s "CN=Example CA" −n my−ca−cert −x −t "C,C,C" −1 −2 −5 −m 3650

The interative prompts for key usage and whether any extensions are critical and responses have been ommitted for brevity.

From there, new certificates can reference the self−signed certificate:

$ certutil −S −s "CN=My Server Cert" −n my−server−cert −c "my−ca−cert" −t ",," −1 −5 −6 −8 −m 730

Generating a Certificate from a Certificate Request

When a certificate request is created, a certificate can be generated by using the request and then referencing a certificate authority signing certificate (the *issuer* specified in the **−c** argument). The issuing certificate must be in the certificate database in the specified directory.

certutil −C −c issuer −i cert−request−file −o output−file [−m serial−number] [−v valid−months] [−w offset−months] −d direct

For example:

$ certutil −C −c "my−ca−cert" −i /home/certs/cert.req −o cert.cer −m 010 −v 12 −w 1 −d $HOME/nssdb −1 nonRepudiation,d

Listing Certificates

The **−L** command option lists all of the certificates listed in the certificate database. The path to the directory (**−d**) is required.

$ certutil −L −d /home/my/sharednssdb

Certificate Nickname	Trust Attributes SSL,S/MIME,JAR/XPI
CA Administrator of Instance pki−ca1's Example Domain ID	u,u,u
TPS Administrator's Example Domain ID	u,u,u
Google Internet Authority	,,
Certificate Authority − Example Domain	CT,C,C

Using additional arguments with **−L** can return and print the information for a single, specific certificate. For example, the **−n** argument passes the certificate name, while the **−a** argument prints the certificate in ASCII format:

$ certutil −L −d $HOME/nssdb −a −n my−ca−cert
−−−−−BEGIN CERTIFICATE−−−−−
MIIB1DCCAT2gAwIBAgICDkIwDQYJKoZIhvcNAQEFBQAwFTETMBEGA1UEAxMKRXhh
bXBsZSBDQTAeFw0xMzAzMTMxOTEwMjlaFw0xMzA2MTMxOTEwMjlaMBUxEzARBgNV
BAMTCkV4YW1wbGUgQ0EwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAJ4Kzqvz
JyBVgFqDXRYSyTBNw1DrxUU/3GvWA/ngjAwHEv0Cul/6sO/gsCvnABHiH6unns6x

```
XRzPORlC2WY3gkk7vmlsLvYpyecNazAi/NAwVnU/66HOsaoVFWE+gBQo99UrN2yk
0BiK/GMFlLm5dXQROgA9ZKKyFdI0LIXtf6SbAgMBAAGjMzAxMBEGCWCGSAGG+EIB
AQQEAwIHADAMBgNVHRMEBTADAQH/MA4GA1UdDwEB/wQEAwICBDANBgkqhkiG9w0B
AQUFAAOBgQA6chkzkACN281d1jKMrc+RHG2UMaQyxiteaLVZO+Ro1nnRUvseDf09
XKYFwPMJjWCihVku6bw/ihZfuMHhxK22Nue6inNQ6eDu7WmrqL8z3iUrQwxs+WiF
ob2rb8XRVVJkzXdXxlk4uo3UtNvw8sAz7sWD71qxKaIHU5q49zijfg==
−−−−−END CERTIFICATE−−−−−
```

For a human−readable display

```
$ certutil −L −d $HOME/nssdb −n my−ca−cert
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 3650 (0xe42)
        Signature Algorithm: PKCS #1 SHA−1 With RSA Encryption
        Issuer: "CN=Example CA"
        Validity:
            Not Before: Wed Mar 13 19:10:29 2013
            Not After : Thu Jun 13 19:10:29 2013
        Subject: "CN=Example CA"
        Subject Public Key Info:
            Public Key Algorithm: PKCS #1 RSA Encryption
            RSA Public Key:
                Modulus:
                    9e:0a:ce:ab:f3:27:20:55:80:5a:83:5d:16:12:c9:30:
                    4d:c3:50:eb:c5:45:3f:dc:6b:d6:03:f9:e0:8c:0c:07:
                    12:fd:02:ba:5f:fa:b0:ef:e0:b0:2b:e7:00:11:e2:1f:
                    ab:a7:9e:ce:b1:5d:1c:cf:39:19:42:d9:66:37:82:49:
                    3b:be:69:6c:2e:f6:29:c9:e7:0d:6b:30:22:fc:d0:30:
                    56:75:3f:eb:a1:ce:b1:aa:15:15:61:3e:80:14:28:f7:
                    d5:2b:37:6c:a4:d0:18:8a:fc:63:05:94:b9:b9:75:74:
                    11:3a:00:3d:64:a2:b2:15:d2:34:2c:85:ed:7f:a4:9b
                Exponent: 65537 (0x10001)
        Signed Extensions:
            Name: Certificate Type
            Data: none

            Name: Certificate Basic Constraints
            Data: Is a CA with no maximum path length.

            Name: Certificate Key Usage
            Critical: True
            Usages: Certificate Signing

    Signature Algorithm: PKCS #1 SHA−1 With RSA Encryption
    Signature:
        3a:72:19:33:90:00:8d:db:cd:5d:d6:32:8c:ad:cf:91:
        1c:6d:94:31:a4:32:c6:2b:5e:68:b5:59:3b:e4:68:d6:
        79:d1:52:fb:1e:0d:fd:3d:5c:a6:05:c0:f3:09:8d:60:
        a2:85:59:2e:e9:bc:3f:8a:16:5f:b8:c1:e1:c4:ad:b6:
        36:e7:ba:8a:73:50:e9:e0:ee:ed:69:ab:a8:bf:33:de:
        25:2b:43:0c:6c:f9:68:85:a1:bd:ab:6f:c5:d1:55:52:
        64:cd:77:57:c6:59:38:ba:8d:d4:b4:db:f0:f2:c0:33:
        ee:c5:83:ef:5a:b1:29:a2:07:53:9a:b8:f7:38:a3:7e
```

```
    Fingerprint (MD5):
      86:D8:A5:8B:8A:26:BE:9E:17:A8:7B:66:10:6B:27:80
    Fingerprint (SHA1):
      48:78:09:EF:C5:D4:0C:BD:D2:64:45:59:EB:03:13:15:F7:A9:D6:F7

    Certificate Trust Flags:
      SSL Flags:
        Valid CA
        Trusted CA
        User
      Email Flags:
        Valid CA
        Trusted CA
        User
      Object Signing Flags:
        Valid CA
        Trusted CA
        User
```

Listing Keys

Keys are the original material used to encrypt certificate data. The keys generated for certificates are stored separately, in the key database.

To list all keys in the database, use the **−K** command option and the (required) **−d** argument to give the path to the directory.

```
$ certutil −K −d $HOME/nssdb
certutil: Checking token "NSS Certificate DB" in slot "NSS User Private Key and Certificate Services                "
< 0> rsa      455a6673bde9375c2887ec8bf8016b3f9f35861d   Thawte Freemail Member's Thawte Consulting (Pty) Ltd. ID
< 1> rsa      40defeeb522ade11090eacebaaf1196a172127df   Example Domain Administrator Cert
< 2> rsa      1d0b06f44f6c03842f7d4f4a1dc78b3bcd1b85a5   John Smith user cert
```

There are ways to narrow the keys listed in the search results:

- To return a specific key, use the **−n** *name* argument with the name of the key.
- If there are multiple security devices loaded, then the **−h** *tokenname* argument can search a specific token or all tokens.
- If there are multiple key types available, then the **−k** *key−type* argument can search a specific type of key, like RSA, DSA, or ECC.

Listing Security Modules

The devices that can be used to store certificates −− both internal databases and external devices like smart cards −− are recognized and used by loading security modules. The **−U** command option lists all of the security modules listed in the secmod.db database. The path to the directory (**−d**) is required.

```
$ certutil −U −d /home/my/sharednssdb

    slot: NSS User Private Key and Certificate Services
   token: NSS Certificate DB
     uri: pkcs11:token=NSS%20Certificate%20DB;manufacturer=Mozilla%20Foundation;serial=0000000000000000;model=N

    slot: NSS Internal Cryptographic Services
   token: NSS Generic Crypto Services
     uri: pkcs11:token=NSS%20Generic%20Crypto%20Services;manufacturer=Mozilla%20Foundation;serial=000000000000
```

Adding Certificates to the Database

Existing certificates or certificate requests can be added manually to the certificate database, even if they were generated elsewhere. This uses the **−A** command option.

certutil −A −n certname −t trustargs −d directory [−a] [−i input−file]

For example:

$ certutil −A −n "CN=My SSL Certificate" −t ",," −d /home/my/sharednssdb −i /home/example−certs/cert.cer

A related command option, **−E**, is used specifically to add email certificates to the certificate database. The **−E** command has the same arguments as the **−A** command. The trust arguments for certificates have the format *SSL,S/MIME,Code−signing*, so the middle trust settings relate most to email certificates (though the others can be set). For example:

$ certutil −E −n "CN=John Smith Email Cert" −t ",P," −d /home/my/sharednssdb −i /home/example−certs/email.cer

Deleting Certificates to the Database

Certificates can be deleted from a database using the **−D** option. The only required options are to give the security database directory and to identify the certificate nickname.

certutil −D −d directory −n "nickname"

For example:

$ certutil −D −d /home/my/sharednssdb −n "my−ssl−cert"

Validating Certificates

A certificate contains an expiration date in itself, and expired certificates are easily rejected. However, certificates can also be revoked before they hit their expiration date. Checking whether a certificate has been revoked requires validating the certificate. Validation can also be used to ensure that the certificate is only used for the purposes it was initially issued for. Validation is carried out by the **−V** command option.

certutil −V −n certificate−name [−b time] [−e] [−u cert−usage] −d directory

For example, to validate an email certificate:

$ certutil −V −n "John Smith's Email Cert" −e −u S,R −d /home/my/sharednssdb

Modifying Certificate Trust Settings

The trust settings (which relate to the operations that a certificate is allowed to be used for) can be changed after a certificate is created or added to the database. This is especially useful for CA certificates, but it can be performed for any type of certificate.

certutil −M −n certificate−name −t trust−args −d directory

For example:

$ certutil −M −n "My CA Certificate" −d /home/my/sharednssdb −t "CT,CT,CT"

Printing the Certificate Chain

Certificates can be issued in *chains* because every certificate authority itself has a certificate; when a CA issues a certificate, it essentially stamps that certificate with its own fingerprint. The **−O** prints the full chain of a certificate, going from the initial CA (the root CA) through ever intermediary CA to the actual certificate. For example, for an email certificate with two CAs in the chain:

$ certutil −d /home/my/sharednssdb −O −n "jsmith@example.com"
"Builtin Object Token:Thawte Personal Freemail CA" [E=personal−freemail@thawte.com,CN=Thawte Personal Freemail CA

"Thawte Personal Freemail Issuing CA − Thawte Consulting" [CN=Thawte Personal Freemail Issuing CA,O=Thawte Consu

"(null)" [E=jsmith@example.com,CN=Thawte Freemail Member]

Resetting a Token

The device which stores certificates −− both external hardware devices and internal software databases −− can be blanked and reused. This operation is performed on the device which stores the data, not directly on the security databases, so the location must be referenced through the token name (**−h**) as well as any directory path. If there is no external token used, the default value is internal.

certutil −T −d directory −h token−name −0 security−officer−password

Many networks have dedicated personnel who handle changes to security tokens (the security officer). This person must supply the password to access the specified token. For example:

$ certutil −T −d /home/my/sharednssdb −h nethsm −0 secret

Upgrading or Merging the Security Databases

Many networks or applications may be using older BerkeleyDB versions of the certificate database (cert8.db). Databases can be upgraded to the new SQLite version of the database (cert9.db) using the **−−upgrade−merge** command option or existing databases can be merged with the new cert9.db databases using the **−−−merge** command.

The **−−upgrade−merge** command must give information about the original database and then use the standard arguments (like **−d**) to give the information about the new databases. The command also requires information that the tool uses for the process to upgrade and write over the original database.

certutil −−upgrade−merge −d directory [−P dbprefix] −−source−dir directory −−source−prefix dbprefix −−upgrade−id id −−up

For example:

$ certutil −−upgrade−merge −d /home/my/sharednssdb −−source−dir /opt/my−app/alias/ −−source−prefix serverapp− −−upgra

The **−−merge** command only requires information about the location of the original database; since it doesn't change the format of the database, it can write over information without performing interim step.

certutil −−merge −d directory [−P dbprefix] −−source−dir directory −−source−prefix dbprefix [−@ password−file]

For example:

$ certutil −−merge −d /home/my/sharednssdb −−source−dir /opt/my−app/alias/ −−source−prefix serverapp−

Running certutil Commands from a Batch File

A series of commands can be run sequentially from a text file with the **−B** command option. The only argument for this specifies the input file.

$ certutil −B −i /path/to/batch−file

NSS DATABASE TYPES

NSS originally used BerkeleyDB databases to store security information. The last versions of these *legacy* databases are:

- cert8.db for certificates
- key3.db for keys
- secmod.db for PKCS #11 module information

BerkeleyDB has performance limitations, though, which prevent it from being easily used by multiple applications simultaneously. NSS has some flexibility that allows applications to use their own, independent database engine while keeping a shared database and working around the access issues. Still,

NSS requires more flexibility to provide a truly shared security database.

In 2009, NSS introduced a new set of databases that are SQLite databases rather than BerkeleyDB. These new databases provide more accessibility and performance:

- cert9.db for certificates
- key4.db for keys
- pkcs11.txt, a listing of all of the PKCS #11 modules, contained in a new subdirectory in the security databases directory

Because the SQLite databases are designed to be shared, these are the *shared* database type. The shared database type is preferred; the legacy format is included for backward compatibility.

By default, the tools (**certutil**, **pk12util**, **modutil**) assume that the given security databases use the SQLite type. Using the legacy databases must be manually specified by using the **dbm:** prefix with the given security directory. For example:

$ certutil −L −d dbm:/home/my/sharednssdb

To set the legacy database type as the default type for the tools, set the **NSS_DEFAULT_DB_TYPE** environment variable to **dbm**:

export NSS_DEFAULT_DB_TYPE="dbm"

This line can be set added to the ˜/.bashrc file to make the change permanent.

- https://wiki.mozilla.org/NSS_Shared_DB_Howto

For an engineering draft on the changes in the shared NSS databases, see the NSS project wiki:

- https://wiki.mozilla.org/NSS_Shared_DB

SEE ALSO

pk12util (1)

modutil (1)

certutil has arguments or operations that use features defined in several IETF RFCs.

- http://tools.ietf.org/html/rfc5280
- http://tools.ietf.org/html/rfc1113
- http://tools.ietf.org/html/rfc1485

The NSS wiki has information on the new database design and how to configure applications to use it.

- https://wiki.mozilla.org/NSS_Shared_DB_Howto
- https://wiki.mozilla.org/NSS_Shared_DB

ADDITIONAL RESOURCES

For information about NSS and other tools related to NSS (like JSS), check out the NSS project wiki at **http://www.mozilla.org/projects/security/pki/nss/**. The NSS site relates directly to NSS code changes and releases.

Mailing lists: https://lists.mozilla.org/listinfo/dev−tech−crypto

IRC: Freenode at #dogtag−pki

AUTHORS

The NSS tools were written and maintained by developers with Netscape, Red Hat, Sun, Oracle, Mozilla, and Google.

Authors: Elio Maldonado <emaldona@redhat.com>, Deon Lackey <dlackey@redhat.com>.

LICENSE

NOTES

1. Mozilla NSS bug 836477
   https://bugzilla.mozilla.org/show_bug.cgi?id=836477