/***

ISC Credential Management Utility (CMU) - Command Line API

Version: 2.7.1.0
Date: 28 December 2018
Authors: Michael Markowitz, Jonathan Schulze-Hewett

***/

Contents

NOTE: Significant enhancements and changes in functionality from the previous release are highlighted in blue in this document.

NAME

cmu – ISC credential management utility

SYNOPSIS

```
cmu  [-h]

cmu  a [-i akID …] [-C] [-L] [-U] [-ec efile] [-sc cfile]

cmu  b [-w timeout]

cmu  c [-i akID …] [-D] [-E] [-L] [-U] [-Z] [-z zones] [-sc cfile]

cmu  d [[-u postfile] | [-w postdata]]  url  [file…] file

cmu  e [-i akID …] [-E | -S | [-N -r dfile [[-R] -d dlist] [-f] [-k[=dbpwd]]]] [-c upcfldr]
       [-P constraints] [-p p12pwd] [-x scsp …] [-L] [-aes128 | -aes192 | -aes256]

cmu  i [-N -r dfile [[-R] -d dlist] [-f] [-k[=dbpwd]] [-w timeout]]
       [-c upcfldr] [-p p12pwd] [-q] [-L] [-T] [-TT] file…

cmu  l [-c upcfldr] [-p p12pwd]

cmu  m [-i akID …] [-D] [-F] [-I | -O] [-L] [-U] [-ec efile] [-sc cfile] [-profile name]
       [-remove] [-no-sha1] [-no-sha256] [-no-sha384] [-no-sha512] [-no-3des]
       [-no-aes128] [-no-aes192] [-no-aes256]

cmu  p [-i akID …] [-C] [-L] [-U] [-s] [-x scsp …] [[-p p12pwd] p12file]

cmu  q [-D | -a | -r display_name | prf_file]

cmu  r [-a] [-r dfile] [-C] [[-R] -d dlist] [-c upcfldr] [-w timeout]
       [-P constraints] [-k[=dbpwd]]

cmu  s [-N -r dfile [[-R] -d dlist] [-f] [-k[=dbpwd]] [-w timeout]]
       [-c upcfldr] [-P constraints] [-p p12pwd] [-q] [-L] [-T] [-TT] [file…]

cmu  u [-a] [-i akID …] [-L] [-U] [-A apps]

cmu  w [-a] [[-R] -d dlist] file

cmu  x p7file

cmu  z [-a] [-f] [-i akID …]
```

DESCRIPTION

The cmu command helps simplify the PKI experience for end-users by off-loading the 'know-how' of credential management to system administrators. A cmu-based script written by an administrator can be 'pushed down' (along with the cmu application, if necessary) to each end-user system where it can be run largely without manual intervention. cmu operations support assisted PKI enrollment, the installation of end-user credentials, and the synchronization of credentials between several supported Windows applications and, for backup and recovery purposes, a user's personal credentials folder (the "UPC"). Some useful CAPI, MAPI, S/MIME, and Outlook configuration operations are also provided.

cmu's primary action is determined by its *key* argument, the first argument placed after the program name on the command line. The function *key* is a single character from the following list: b, c, d, e, i, l, m, p, q, r, s, u, w, z.

The default UPC folder containing user credentials is '%AppData%\Information Security Corp\cmu\', but the '-c *upcfldr*' argument can be used to specify an alternate folder. To facilitate identification and avoid duplicates, credentials are maintained in the UPC folder as PKCS#12 files with filenames that are formed by appending the extension '.p12' to a dash-delimited list consisting of the subject common name, keyUsage, expiration date, and SHA-1 message digest of the end-user certificate contained in the file. (At this time we do not anticipate the need to store PKCS#7 files in the UPC.) Thus the filename coincides with the certificate's "Thumbprint" as it might be displayed by the MSIE GUI when viewing the certificate's properties using the following sequence of menu/tab/button selections: Tools | Internet Options | Content | Certificates | <select certificate> | View | Details.

The '-d *dlist*' argument can be used to specify one or more Windows directories in which to search for the user's NSS database files ('cert?.db' and `key?.db'); *dlist* denotes a comma-delimited list of directory paths. Optionally, '-r *dfile*' can be used to read a list of NSS directories (listed one per line) from a text file. '-d' directories are only searched recursively if '-R' is used; '-r' directories are never searched recursively. If both '-d' and '-r' are specified, '-r' directories are searched first. If neither `-d' nor '-r' are specified, cmu performs (recursive) searches through the D_NETSCAPE and/or D_MOZILLA directory trees.

Normally, cmu aborts execution of operations that require read or write access to one or more NSS certificate databases if it detects any version of Netscape, Mozilla, or FireFox running on the user's system. (This is done to prevent access conflicts over the certificate database files.) The '-f' option forces cmu to skip the runtime check for Netscape processes.

One or more '-i *akID*' arguments can be used to limit certificate processing to those end-user certificates issued by a CA with a matching authority key identifier (or, in version 2.6, issuer DN). This option is available with the 'a', 'c', 'e', 'm', 'p', 'u', and 'z' operations. *akID* values are now case insensitive.

The '-k[=*dbpwd*]' argument can be used to supply the user's NSS database password and is required for the reinitialize operation and for all export and synchronize operations if an NSS database is password protected. If there are NSS databases in the directory search paths and `-k' is specified without a password argument or with an empty argument ('-k=' or '-k=""'), an empty password is used. If the *dbpwd* argument begins with the string "file:", the remainder of the string will be treated as the pathname of a Windows DPAPI-encrypted password file[1] whose contents will be decrypted and used as the user's NSS password. If the `-k' option is omitted entirely, the user will be interactively prompted to enter a password at runtime.

The '-p *p12pwd*' argument supplies your PKCS#12 password and is required for all export, synchronize and publish to GAL operations ('e', 's', and 'p' operations). If the *p12pwd* argument begins with the string "file:", the remainder of the string will be treated as the pathname of a Windows DPAPI-encrypted password file[1] whose contents will be decrypted and used as the user's password. If not explicitly provided on the command line, the user will be interactively prompted for it at runtime.

[1] A file containing the user's password encrypted under the Windows Data Protection API (DPAPI) function CryptProtectData() may be created using cpf4cmu or the CSP[id] command line. See [DPAPI].

All operations accept one or more of the "common" command line switches: '-l', '-t', and '-v'. The '-l *logfile*' argument can be used to specify a pathname for the log file, thereby overriding the hard-wired default value '%AppData%\Information Security Corp\cmu\cmulog.txt'. '-t *cfg_file*' allows you to specify a configuration file with user prompt strings that replace hardwired default values; this option can also be used to relocate the NSS utilities in directories other than their default locations, and to change the formatting of dates in p12 filenames from the new (ISO 8601-compliant) default of "yyyymmdd" to the backwards-compatible "mmddyyyy" format. '-v' puts the executable in 'verbose mode' so that it produces more diagnostic output. These three options are more fully described below.

The '-r *display_name*' option can be used with the 'q' operation to remove a single Outlook query specified by display name.

One or more '-x *scsp*' arguments can be used to bypass the specified smartcard service providers during an 'e' operation. *scsp* values are either the name of a single smartcard/cryptographic service provider (SCSP), or the complete pathname of a text file containing multiple SCSP names listed one per line.

Function Letters

The function key (first command line argument) that specifies which operation cmu is to perform must be one of the following lowercase letters (or the indicated equivalent command word or unique abbreviation thereof):

Command	**Description**
a archive	Sets the ARCHIVED property (i.e., the certificate property with the tag CERT_ARCHIVED_PROP_ID) on all CAPI certificates except those explicitly specified. If none are specified, the freshest signing and/or encrypting certificates are skipped. If '-C' is specified, the ARCHIVED property is removed from all certificates that have that property.
b buttons outlook	Adds S/MIME encrypt and sign buttons to Outlook's message composition toolbar. Since this operation will fail if Outlook is configured to use Word as its e-mail editor, consider using '-w' to change user preferences to use the built-in editor. If '-w' is specified, cmu must first stop Outlook, make some HKCU registry changes, and then restart it; if Outlook refuses to terminate, the operation will fail. If '-w' is not specified, cmu must start Outlook to add the S/MIME buttons if it is not already running. In either case, cmu leaves Outlook running. (NOTE: The 'b' operation works only with Outlook 2000 and above; special builds of the cmu executable may be required for earlier releases. It is also worth noting that if Outlook 2007 is configured to use Word as its e-mail editor, S/MIME buttons will magically appear in Word once S/MIME is enabled in MAPI, e.g., by using the 'm' operation.)
c client clientauth	Without '-Z', configures TLS client authentication in CAPI to use the freshest signing certificate in the user's personal certificate store. (The "Client Authentication" setting is removed from CAPI's "Certificate Purposes" list of enhanced key usage extensions for all certificates in this store other than the freshest signing certificate. As of version 3.2.2, this setting is not changed for certificates that do not support signing.) In this case, if `-D' is specified, "Client Authentication" is also enabled for all DAS certificates. With '-Z', adds "Client Authentication" to CAPI's "Certificate Purposes" list of enhanced key usage extensions for all signing certificates in the user's personal certificate store that have an associated private key. (In this case, '-D' would be redundant and is ignored.) If one or more '-i *akID*' options are used, only certificates issued by a CA with a matching authorityKeyIdentifier (or, in version 2.6, issuer DN) are processed. If the '-z zones' option is used, cmu enables the security setting "Don't

prompt for client certificate selection when no certificates or only one certificate exists" for all specified internet zones. (For details, see "Configuring Client Authentication in CAPI" in the NOTES section below.)

To explicitly load a signing certificate, without performing a CAPI search for the freshest one, specify the option '-sc *cfile*'. In this case the '*cfile*' filename argument must reference a file containing the ASN.1 DER-encoded certificate you wish to use. NOTE: Use of the '-sc' option does not cause the specified certificate to be installed into CAPI; the certificate and its corresponding private key must already be (or appear to be) in CAPI or the command will return a ERR_NO_SIGNING_CERT error. (This functionality was added to facilitate role-based signing where the specified certificate belongs to a particular role for which signing operations are to be mediated by a DAS server.)

If `-E' is specified, the "Client Authentication" setting is also removed from encrypt-only keys.

d
down
download

Downloads specified url as file via HTTPS or HTTP. (Use of HTTPS presumes 'cmu c' has already been run so that user's freshest signing certificate can be used automatically for client authentication.) The url operand must begin with either of the protocol specification strings 'https://' or 'http://'. If the '-u' or '-w' switches are used, the specified string literal or file contents are POST'ed to the specified url operand; otherwise a simple GET is employed. Upon success, the results returned by the server are stored in the designated file operand. The `-w' argument now supports replaceable parameters that reference DPAPI-encrypted files. If used, the final command line argument is assumed to be the output file specification.

NOTE: If both '-u' and '-w' are specified, a warning message is written to the logfile and only the contents of the file (the '-u' argument) are POSTed.

As of cmu release 2.1, server redirects are not permitted (so that page or file not found errors can be properly caught). For this reason the specified url operand must now directly reference the desired data.

e
exp
export

Extracts all user key pairs from CAPI or, if '-N' is used, from NSS-based applications and saves them as PKCS#12 files in the UPC. The name of each PKCS#12 output file is constructed by appending the extension '.p12' onto a string formed by concatenating together the user's friendly name (filtered of illegal filename characters), a word indicating the keyUsage attribute of that key pair, the notAfter date (by default now formatted in an ISO 8601-compliant manner as 'yyyymmdd' and the first 20 characters of the "certificate thumbprint" (*i.e.*, the first half of the SHA-1 message digest of the user's certificate). All existing PKCS#12 files in the UPC ending with the same half certificate thumbprint are overwritten or deleted without warning.

As of cmu release 2.4, the 'e' operation issues a warning for certificates with private keys that can't be exported, but ignores certificates without private keys and certificates whose corresponding private key resides on one of the following "standard" smartcard service providers (SCSP):

Microsoft Base Smart Card Crypto Provider
Microsoft Smart Card Key Storage Provider
CSPid Key Storage Provider
CSPid Key Generation Provider

Additional third party smartcard providers may be excluded by using one or more `-x *scsp*' options.[2] Here *scsp* is the name of a single SCSP, or is the complete pathname of a text file containing multiple SCSP names listed one per line.

As of cmu release 2.5, the '-i' option can be used to filter the exported key pairs by issuer ID on the associated certificate, '-S' can be used to

[2] This can be used to eliminate unnecessary prompts to insert the specified smartcards during the backup process, and empty PKCS#12 files should no longer be generated.

extract only the freshest signing key pair associated to a matching certificate, and '-E' the freshest encryption key pair.

As of cmu release 2.6, the default cipher used for internal private key protection as well as external certificate wrapping is 3DES; for AES-CBC, use the command line option -aes128, -aes192, or -aes256. In any case, the outer encryption wrapping is 3DES-CBC for compatibility with the largest number of popular applications.

i
imp
import

Causes the contents of all specified PKCS#7, PKCS#12 and CRL file(s) to be imported into CAPI or, if '-N' is specified, into NSS key stores on the user's system. (NOTE: CRLs are not currently supported with `-N'.) The 'i' operation does not alter the contents of the UPC. (PKCS#12 input files must have a '.p12' or '.pfx' extension; PKCS#7 files must have an extension matching '.p7?'; and CRL files must have an extension of `.crl'. First all p7 files on the command line are processed, then p12/pfx files, then CRLs. Wildcards are supported.)

WARNING: `-N' may not work correctly with multiple PKCS#12 files; the workaround is to simply use multiple invocations of cmu, passing in a single PKCS#12 file on each call.

l
list

Recursively scans the directory tree rooted at the active UPC and dumps to the log a list of all PKCS#12 ('*.p12' or '*.pfx') files found there together with their existing friendly names (when available), or with the friendly names that they would be assigned during a 'synchronize' operation. If the '-v' option is specified, the output will also contain keyUsage and notAfter attribute values as well as hash values, and be echoed to the display.

m
mapi
smime

Set the freshest signing and/or encrypting certificate(s) found in CAPI as the S/MIME certificates in the user's default MAPI security profile for use with Outlook. (Creates a new MAPI security profile, if one does not already exist; see the 'MAPI Security Profile Update' section under NOTES for details.)

If '-profile' is specified, the name provided will be used to determine the profile that is added or modified instead of the default profile.

If '-remove' is specified, the profile named by the '-profile' option will be removed. If '*' is provided as the profile name, all profiles will be removed.

If '-D' is specified, the user's MAPI security profile is not modified, but information about it is written to the log in the following format:

```
MAPI signing certificate: 7834A53BF693BB2502083F383831B23D991EDFBC
  iCN= 'ISC Root'
  akID= F9B20F9778E6D5090F2AC47EBBC6E7AA353C76FB
  sCN= 'John G. Doe'
  nB= 26 FEB 2007 00:00 GMT; nA= 26 FEB 2007 00:00 GMT
  kU= e8
MAPI encryption certificate: 7834A53BF693BB2502083F383831B23D991EDFBC
  iCN= 'ISC Root'
  akID= F9B20F9778E6D5090F2AC47EBBC6E7AA353C76FB
  sCN= 'John G. Doe'
  nB= 26 FEB 2007 00:00 GMT; nA= 26 FEB 2007 00:00 GMT
  kU= e8
```

Here the certificate is identified by its SHA-1 'thumbprint', issuer common name and authorityKeyIdentifier, subject common name, validity period, and keyUsage value. If one or more '-i akID' options are used, a warning is issued (and an error code is returned by the executable) if the akID (or, in version 2.6, issuer DN) in either certificate doesn't match one of the supplied values.

To explicitly load a signing or encryption certificate, without performing a CAPI search for the freshest one, specify the option '-ec e*file*' or '-sc

c*file*'. In each case the filename argument must reference a file containing the ASN.1 DER-encoded certificate you wish to use for the indicated purpose. (This functionality was added to facilitate role-based signing and/or decryption where the specified certificates are likely to belong to a "community of interest" (CoI) or role and the signing/decryption operations are to be mediated by a DAS server.)

If '-I' is specified, the user's MAPI security profile is marked in such a way as to cause Outlook to include the user's certificates in outgoing S/MIME messages. (This is equivalent to checking the "Send certificates with messages" box in Outlook's Options settings for encrypted e-mail.) Specify '-O' to uncheck this option. If neither '-I' nor '-O' is specified, an existing default security profile is left unchanged, but if a new one must be created, it is marked to include certificates.

If '-F' is specified, the S/MIME algorithm settings in the user's default MAPI profile are reset to Outlook default values.

The '-no-sha1', '-no-3des', etc. options are used to remove algorithms from the selection boxes in Outlook for a given profile. To ensure that a specific algorithm is selected remove all other algorithms in the same family. Removing all the hashes or all the ciphers will result in an error.

p
pub
publish
gal

Publish (to GAL). Extracts the user's private key and certificate from a PKCS#12 file, if one is specified on the command line*, and uses them to create a signed 0-length S/MIME message that is "published" to the user's entry in the global address list (GAL) using MAPI to identify the appropriate Exchange Server and current user account. If no PKCS#12 filename is specified, the "freshest" encryption certificate with an associated private key in the user's personal certificate store in CAPI is used. (For details, see "Publish to GAL" in the NOTES section below.)

As of cmu release 2.4.1.1, this command ignores certificates whose corresponding private key resides on a smartcard provider specified by an `-x *scsp*' option. Here *scsp* is the name of a single SCSP, or is the complete pathname of a text file containing multiple SCSP names listed one per line. (No smartcard providers are automatically excluded as with the `e' command; if you wish to exclude one or more SCSPs, they must be explicitly specified.)

q
query
queryadd

Lists, creates, updates, or removes LDAP queries ("directories" or "address books") in Outlook using a specified *displayName* or parameters specified in one or more PRF_files. Use '-D' to display the list of existing queries; this is the default operation if no parameters are provided. Use '-r *displayName*' to remove a specific query, '-a' to remove all queries. (Be sure to place double quotes around *displayName* if it contains any spaces.) To create or update a query, specify a PRF file, *i.e.*, an ASCII text file containing lines of the form:

tag=string_value

where *tag* is one of strings defined in the following table.

Tag	Description
ServerName	server name or IP address (required)
DisplayName	display name for query in user's Address Book (required)
ConnectionPort	LDAP port on server (defaults to 389)
UseSSL	TRUE or FALSE (defaults to FALSE)
UseSPA	TRUE or FALSE (defaults to FALSE)
EnableBrowsing	0 or 1 (defaults to 1); if 1 and the LDAP directory supports browsing, the user should be able to open the address book and page up or down through it; in this case, the directory should be small or the user may experience severe performance issues, see http://support2.microsoft.com/kb/820864; note that this option may also be controlled, and overridden, by the registry key: HKCU\Software\Microsoft\Office\12.0\Outlook\LDAP\DisableVLVBrowsing)
UserName	user name (for authentication; we suggest omitting this, or leaving its value blank, and setting 'PromptUser=1' if authentication is required)
SearchBase	search base (defaults to none)
SearchTimeout	timeout value in seconds (defaults to 60)
MaxEntriesReturned	maximum number of entries to return (defaults to 100)
CheckNames	filter (defaults to none)
PromptUser	0 or 1 (defaults to 0); if 1, user will be prompted to finish configuring the query (e.g., by supplying his user name and password if authentication is required)

Note that the tags DisplayName and ServerName must be included and assigned non-empty string values or the query creation/update operation will fail. All other tags are optional and, if omitted, will be assigned their MAPI default values as indicated in the table.

Reference: http://technet.microsoft.com/en-us/library/cc179232%28v=office.12%29.aspx

r
init
reinit

Reinitialize NSS databases. Unless '-C' (for "Clear all") is specified, all NSS key stores associated with the user's default profiles are copied into the active UPC and renamed (by prefixing their existing names with the current date and time). Then the original database files are deleted and new database files are created with the specified password. (Regardless of how many profile folders containing a 'cert?.db' file are found, only the first is processed unless the '-a' option is used.)

s
synch

Synchronize. Recursively scans the directory tree rooted at the active UPC for PKCS#12 ('*.p12' or '*.pfx') files and either imports them into CAPI or, if '-N' is used, into the specified NSS databases. All PKCS#7 ('*.p7b') and PKCS#12 input files specified on the command line are also imported. Wildcards are supported.

u
upd
update
: Update the cryptographic profiles of supported releases of Adobe Acrobat and Acrobat Reader (or ISC applications). Installs the user's freshest signing certificate (and, for ISC products, encryption certificate) into the user's profiles for Adobe Acrobat, Acrobat Reader, SecretAgent, and SpyProof!. By default, user profiles for all installed applications are updated; '-A *apps*' restricts the operation to those applications named in the argument *apps*: if *apps* contains the letter 'a', all Adobe products are updated; if *apps* contains the letter 's', SecretAgent is updated; and if *apps* contains the letter 'p', SpyProof! is updated. If the user has multiple SecretAgent or SpyProof! profiles, only the default profile is updated unless the '-a' option is specified. (Note: An error is returned only if the specified application is found to be installed, but cannot be updated; not finding any of the applications is considered success.)

  The current release of cmu supports Adobe Acrobat and Acrobat Reader versions 10, 11, and DC; it is not needed (and therefore does not work) with SecretAgent 6.x and above and SpyProof! 2.x and above.

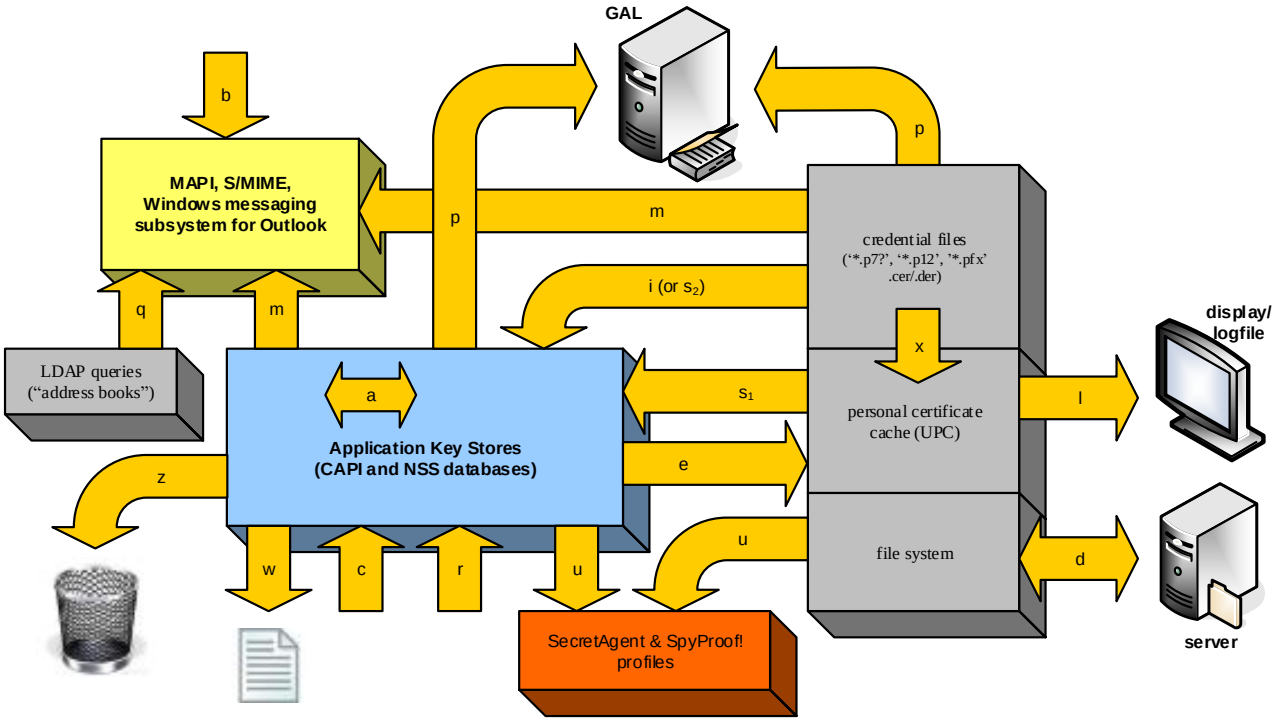
w
write
: Write a list of discovered NSS databases to a specified output text file; the search process for NSS databases is the same as that for the 'e' command.

x
extract
: Extract the first leaf certificate from a PKCS#7 input file. Assuming that the input file consists of a single certificate chain, this command extracts the first leaf node (i.e., final end-user certificate in the chain) and writes it into the UPC as a DER-encoded .cer file whose filename is the certificate fingerprint. (.cer files may be opened for inspection in Windows Explorer using the Microsoft Crypto Shell Extensions.)

z
zap
: Erase all credentials (except those that are self-signed) from the user's personal certificate store in CAPI. Normally, the user is prompted for confirmation before each certificate is erased; specify '-f' to suppress these prompts. If '-a' is specified, all non-EFS self-signed credentials are also removed. Applying one or more `-i AKID' options limits the credentials to be erased to only those issued by a CA with a matching authorityKeyIdentifier (or, in version 2.6, issuer DN).

NOTE: Starting with release 2.0, multiple operations can no longer be combined in a single invocation of the program; if multiple cmu operations are required, multiple invocations of the executable must be made.

The following diagram attempts to illustrate the actions of the various operations provided by cmu:



OPTIONS

The following options are supported:

-A *app* With 'u': update only user profiles for the specified applications: *app* = 'a' for Adobe Acrobat, *app* = 's' for SecretAgent, *app* = 'p' for SpyProof! Omit '-A' or use *app* = 'asp' to update profiles for all three applications.

-C With 'a': remove the archived property from all CAPI certificates that have that property. With 'p': clear an existing certificate from GAL before publishing the new ones. With 'r': clear Mozilla database(s) but don't backup keys first.

-D With 'c': enable client authentication for all DAS certificates.

With 'q': display list of existing Outlook queries.

With 'm': display current MAPI security profile.

-E With 'e': only the freshest encryption key pair associated to a matching certificate in CAPI is exported; may be combined with '-S' but not with '-N'. cmu returns ERR_NO_CAPI_CERT if '-E' is specified, but no suitable encryption certificate with a private key can be found in CAPI.

-F with 'm': force a reset of the MAPI security profile's algorithm selections.

-I With 'm': mark the user's MAPI security profile so that Outlook includes the user's certificates in outgoing S/MIME messages. (This is equivalent to checking the "Send certificates with messages" box in Outlook's Options settings for encrypted e-mail.)

-L Perform CAPI-related operations using the local machine store (CERT_SYSTEM_STORE_LOCAL_MACHINE) rather than the default current user store (CERT_SYSTEM_STORE_CURRENT_USER). This allows processes with administrative privileges to silently install trusted root certificates on the user's system.

-N Perform requested operation on (or using) NSS databases rather than CAPI stores.

-O With 'm': mark the user's MAPI security profile so that Outlook omits the user's certificates from outgoing S/MIME messages. (This is equivalent to unchecking the "Send certificates with messages" box in Outlook's Options settings for encrypted e-mail.)

-P *constraints*

Specify the set of requirements that user-created passwords must meet. *Constraints* is a word with an optional numeric prefix followed by zero or more letters from the following table:

Item	Description
#	A number specifying the minimum password length
a	The password must include alphabetic characters
n	The password must include numeric characters
s	The password must include special characters **~@#$%^&*()-_+={}[]\\|<>/**
p	The password must include punctuation **`.,?"';:!**

For example, the command line option '*-P 17ap*' requires any user-supplied password to be at least 17 characters long and to contain at least one letter and at least one punctuation mark.

-R Causes all '-d' directories to be searched recursively for NSS databases.

WARNING: If ‘-R’ is used and a specified ‘-d’ argument is close to the root of a large directory tree, the recursive search for NSS databases can be quite expensive in terms of execution time! For this reason, one should either avoid using ‘-R’ altogether or use it wisely with narrowly targeted ‘-d’ directory lists.

-U When searching CAPI for freshest certificates, ignore DAS certificates and consider only true end-user certificates with associated private keys.

-S With ‘e’: only the freshest signing key pair associated to a matching certificate in CAPI is exported; may be combined with ‘-E’ but not with ‘-N’. cmu returns ERR_NO_CAPI_CERT if ‘-S’ is specified, but no suitable signing certificate with a private key can be found in CAPI.

-T Import end-entity certificates in found PKCS#7 files into the “Trusted Publishers” CAPI store rather than the “Other People” store.

-TT Import all certificates found on the command line into the “Trusted Publishers” CAPI store.

-Z With ‘c’, reset the CAPI usage settings for all personal certificates (or just those whose issuer authorityKeyIdentifier or, in version 2.6, issuer DN, matches the accompanying ‘-i akID’ parameter), making those signing certificates available for TLS client authentication. While this is essentially an “undo” option for the ‘c’ operation, it may have the unwanted side-effect of making too many certificates available for client authentication purposes, thereby annoying the poor end-user who will have to deal with IE’s signing certificate selection prompts for each HTTPS connection.

-a With ‘q’: remove *all* existing LDAP queries.

With ‘r’: process all databases in the NSS database search paths. (If ‘-a’ is omitted, cmu processes only the *first* database found in the D_NETSCAPE and/or D_MOZILLA directory trees), or in a “default” subdirectory of a ‘-d’ argument.)

With ‘u’: update the signing and encryption credentials in all user profiles rather than just the user-selected default profile.

With ‘w’: append/merge NSS database paths to the specified output file instead of overwriting it if it already exists.

With ‘z’: erase all (non-EFS) self-signed credentials as well as credentials that aren’t self-signed.

-c *upcfldr*
Use the specified *upcfldr* rather than ‘%AppData%\Information Security Corp\cmu\’ as the UPC. (The specified folder is tested for existence -- and created if necessary -- only when used with the ‘e’ and ‘s’ operations.)

-d *dlist*
Scan each element of the comma-delimited *dlist* (and recursively their child directories if ‘-R’ is used) for NSS database files; applies only if ‘-N’ is specified explicitly (or implicitly as for ‘r’ and ‘w’). For example, one might wish to use the command line arguments

```
-N -d u:\_sys\Phoenix,g:\apps\Mozilla
```

or

```
-N -d “c:\Documents and Settings\John Doe\Application Data\Phoenix\Profiles”
```

to search for NSS databases in the user’s FireFox profiles folder.

Note that if ‘-d’ is used, the specified *dlist* arguments (and, with ‘-R’, their sub-directories) are scanned *in addition* to either:
a) the directories specified in a ‘-r’ file argument, or
b) the D_NETSCAPE and/or D_MOZILLA directory trees

-ec *cfile*
Specify a certificate file rather than letting cmu find the freshest encryption certificate in CAPI. The filename argument must reference a file containing an ASN.1 DER-encoded certificate.

-f With 'e -N': forces execution regardless of whether the user is currently running a Netscape, Mozilla, or FireFox application. (Since for both NSS 3.90 and 3.10 'certutil -N' will almost certainly crash and 'pk12util -d' will frequently abort when executed while Firefox is running, '-f' is no longer supported with the 'r', 'i', and 's' operations.)

With 'z': user is not asked for confirmation before each certificate is erased.

-h Display brief usage summary and exit.

-i *akID* May be repeated. Each *akID* must be either an explicit authorityKeyIdentifier value or the complete pathname of a text file containing a single authorityKeyIdentifier on each line (*i.e.*, newlines in the input file are regarded as field separators; the final line need not be terminated with a newline). As of version 2.6, a properly quoted issuer DN may also be used.

Note: An authorityKeyIdentifier is normally specified as 40 uppercase hex digits - representing the SHA-1 hash of the issuer's public key -- with no embedded spaces; in any case, cmu will only support embedded spaces in authorityKeyIdentifiers if they are provided via the input file option.

With 'a': only certificates from the specified issuers are marked as archived.

With 'c': for TLS client authentication, CAPI is configured to use the freshest signing certificate issued by any of the specified CAs with a matching authorityKeyIdentifier.

With 'e': only certificates from the specified issuers are exported.

With 'p': the freshest encryption certificate issued by one of the specified CAs is published to GAL.

With 'm -D': the authorityKeyIdentifier of the user's S/MIME certificates are compared with the specified *akIDs* and a warning is issued if no match is found. Otherwise when used with 'm', Outlook S/MIME is configured to use the freshest signing and encryption certificate(s) issued by one of the specified CAs.

With 'u': the specified ISC application profiles are updated with the freshest certificates issued by one of the specified CAs.

-l *logfile*
Use specified file rather than '%AppData%\Information Security Corp\cmu\cmulog.txt' as the log file in which to record all operations performed by the current invocation of the cmu command. If the specified file does not exist and cannot be created, the file '.\cmulog.txt' is used instead.

-k[=*dbpwd*]
Use specified NSS database password. Must be specified for 'i', 'e' and 's' operations if your key database is password protected; provides the required password for your new NSS database with the 'r' operation. If there are NSS databases in the directory search paths and `-k' is provided on the command line without an explicit *dbpwd* argument or with an empty argument ('-k="" ' or simply '-k='), an empty password string will be used. If '-k' is omitted entirely, the user will be interactively prompted for a password at runtime. (NOTE: '-k *dbpwd*' is no longer legitimate syntax for passing in a database password; an equal sign before the option argument is required!)

If the *dbpwd* argument begins with the string "file:" the remainder of the argument is assumed to be the path to a Windows DPAPI-encrypted password file whose contents are decrypted and used as the actual password.

WARNING: Do not group any other simple command line switches before the '-k' switch. Doing so may cause its *dbpwd* argument to be leaked to the log file.

-no-*algorithm*

Do not include *algorithm* in the list of available options from which the user can select when configuring a MAPI security profile (*e.g.*, 'cmu m –no-aes128' will remove AES-128 from the symmetric cipher list in the default profile).

-p *p12pwd*
Use the specified password *whenever possible* when importing private keys into (or extracting exportable private keys out of) one of the supported applications or, with the 'p' operation, from the supplied *p12file*. (Use of this option with the 'e' operation may or may not suppress an application's native password dialog.) This argument is required for the 'e', 'l' and 's' operations; if not explicitly provided on the command line, the user will be interactively prompted for it at runtime. If the *p12pwd* argument contains spaces, it must be double quoted.

If the *p12pwd* argument begins with the string "file:" the remainder of the argument is assumed to be the path to a Windows DPAPI-encrypted password file whose contents are decrypted and used as the actual password.

WARNING: Do not group any other simple command line switches before the '-p' switch. Doing so may cause its *p12pwd* argument to be leaked to the log file.

-profile *name*
Use the specified name as the MAPI security profile name.

-q Quiet operation. When importing PKCS#12 files into the CAPI store, suppresses the "Importing a new private exchange key" dialog that normally allows a use to specify a security level (and optionally to set a password) for the private key being imported.

-r *display_name*
With 'q': remove named LDAP query from Outlook.

-r *dfile*
With all NSS-related operations: read from the specified text file a list of directories (one per line) and operate on the NSS database files they contain; applies only if '-N' is specified explicitly or implicitly (as with the 'r' operation). For example, one might create a text file 'nssdirs.txt' containing the lines:

```
C:\Documents and Settings\John Doe\Application Data\Mozilla\Firefox\Profiles\default.b70
C:\Documents and Settings\John Doe\Application Data\Phoenix\Profiles\default\3vud3lhz.slt
```

and use the '-r nssdirs.txt' command line switch to target only the NSS databases in these two directories.

Note: '-r' directories are never scanned recursively, so they must be carefully targeted. For this reason, it is suggested that they always be constructed by running the 'w' operation on the user's system.

Note: If '-r' is used, the default NSS directory trees D_NETSCAPE and/or D_MOZILLA are not processed (unless they are explicitly mentioned in the '-r' file argument or included in the directory list of the '-d' option).

-remove
Remove the named MAPI security profile or all MAPI security profiles if the name is '*' (*e.g.*, cmu m –profile * -remove).

-s Do not modify the userSMIMECertificate attribute when publishing to GAL; only userCertificate is updated.

-sc *cfile*
Specify a certificate file rather than letting cmu find the freshest signing certificate in CAPI. The filename argument must reference a file containing an ASN.1 DER-encoded certificate.

-t *cfg_file*
Use text strings in *cfg_file* to override various hardwired default values, such as user prompts and NSS database directories. *cfg_file* must be an ASCII text file containing lines of the form:

tag=string_value

where *tag* is one of strings defined in the following table.

Tag	Description
P_P12PWD	prompt for PKCS#12 password
P_P12CONFIRM	prompt to confirm PKCS#12 password
P_PWDMATCH	prompt to reenter password after confirmation mismatch
P_PWDEMPTY	inform user that an empty password is not acceptable
P_NSSPWD	prompt for NSS database password
P_CLOSEBROWSER	tell user that browser is running and ask that it be closed
P_NSSNEWPWD	prompt for new NSS database password
P_NSSCONFIRM	prompt to confirm new NSS password
D_NETSCAPE	root folder for Netscape profiles (defaults to '%AppData%\Netscape'; if empty, search is skipped)
D_MOZILLA	root folder for Mozilla profiles (defaults to '%AppData%\Mozilla'; if empty, search is skipped)
~~D_NSS320~~	~~NSS 3.2.0 tools folder (defaults to '.\nss320'); these tools are used for a few operations on NSS "cert7.db" database files; removed from distribution; request from tech support, if needed~~
~~D_NSS361~~	~~NSS 3.6.1 tools folder (defaults to '.\nss361'); these tools are used for most operations on NSS "cert7.db" database files; request from tech support, if needed~~
~~D_NSS390~~	~~NSS 3.9.0 tools folder (defaults to '.\nss390') (these tools were previously used for all operations on NSS "cert8.db" database files; deprecated in favor of NSS 3.10.0 or later builds)~~
D_NSSTOOLS	NSS tools folder (defaults to '.\nss3a0'); tools found in this folder are used for all operations on NSS "cert8.db" database files; to substitute NSS 3.9.x, 3.11.x, 3.12.x or 3_40 tools, simply point this variable at the appropriate subdirectory (which may be obtained from tech support, if not already included in your distribution)
D_NSS_3_40	NSS 3_40 tools folder (defaults to '.\nss_3_40'); these tools are used for all operations on NSS "cert9.db" database files
F_DATEFMT	date format in PKCS#12 filenames; defaults to "yyyymmdd", but can be changed to "mmddyyyy"

If the '-t' option is omitted, or any tags are missing from the specified *cfg_file*, string values hard-wired into the cmu executable are used. (See supplied sample 'default.cfg' file for the default values.) Unless redirected, the NSS tools directories are expected to be found immediately below the directory containing the cmu executable.

NOTE: If you provide an empty *string_value* for either D_NETSCAPE or D_MOZILLA, no directories will be searched for user databases belonging to the corresponding application.

-u *file*
: With `d': the contents of *file* are POSTed to the specified *url* operand and the result is returned in the designated target *file*.

-v
: Verbose mode. Output to log file extended execution information, including all internally generated 'certutil' and 'pk12util' command lines. (For security reasons password arguments are always suppressed from this output.)

-w *postdata*
: With `d': the specified *postdata* is POSTed to the specified *url* operand and the result is returned in the designated target *file*. *postdata* may contain replaceable parameters that are successively read out of one or more DPAPI-encrypted files. For example, the command:

```
cmu d -w "pin1=%1&pin2=%2" https://ccms.com/pickup f1.bin f2.bin output.der
```

will cause the contents of f1.bin and f2.bin to be read from disk, decrypted, and substituted into the *postdata* argument as if it were

"pin1=*<plaintext from f1.bin>*&pin2=*<plaintext from f2.bin>*"

Only single digit parameters ('%[1..9]') are supported but they need not be referenced sequentially. There must be at least as many file specifications on the command line as are referenced by the different replaceable parameters, but not all must actually be used. For example:

```
cmu d -w "pin1=%3&pin2=%1&pin3=%1" https://ccms.com/pickup \
      f1.bin f2.bin f3.bin f4.bin output.der
```

will result in the following string being posted:

"pin1=*<contents of f3.bin>*&pin2=*<contents of f1.bin>*&pin3=*<contents of f1.bin>*"

-w *timeout*
: With `b': attempts to terminate Outlook if it is already running and then disables Word as the e-mail and RTF editor. The operation fails if Outlook cannot be shut down in the specified number of seconds.

With `i', 'r', and 's': attempts to terminate a Mozilla-based browser (Netscape, Mozilla, or Firefox) if one is running. The operation fails if the browser cannot be shut down in the specified number of seconds.

-x *scsp*
: With `e' and `p': ignore those certificates whose private keys reside in the specified (third party) smartcard (or cryptographic) service *provider*. May be repeated to exclude multiple smartcards and/or HSMs. (As of version 2.4.1.1, the standard SCSPs "Microsoft Base Smart Card Crypto Provider," "Microsoft Smart Card Key Storage Provider," and ISC's "CSPid Key Storage Provider" and "CSPid Key Generation Provider" are all automatically excluded with `e' but, if desired, must be explicitly specified with 'p'.) The *scsp* argument is either the name of a single SCSP or the complete pathname of a text file with an SCSP name on each line (*i.e.*, newlines in the input file are regarded as field separators; the final line need not be terminated with a newline).

-z *zones*
: Internet zones for which the security setting "Don't prompt for client certificate selection when no certificates or only one certificate exists" is to be enabled. Only works in conjunction with the 'c' operation. *zones* is a word composed of one or more of the following decimal digits:

Digit	Description
0	URLZONE_LOCAL_MACHINE - local machine
1	URLZONE_INTRANET - intranet
2	URLZONE_TRUSTED - trusted sites

3	URLZONE_INTERNET – internet sites
4	URLZONE_UNTRUSTED – untrusted sites

OPERANDS

The following operands are supported:

file The pathname of a PKCS#7 or PKCS#12 file containing the user's credentials or organizational certificate chain, or the name of a file in which to store the data returned from the specified *url* by the 'd' operation, or the PRF-like file containing LDAP parameters for the 'q' operation. For the 'i' and 's' operations, an arbitrary number of p7 and/or p12/pfx filespecs (with or without wildcards) may be specified in a whitespace-delimited list on the same command line. If a filename argument is not supplied on the command line for the 'p' operation, the "freshest" PKCS#12 file in the cache is published to GAL. For the 'q' operation, the pathname of the PRF-like file defining the LDAP query must be specified.

File and path names that contain spaces must be double quoted.

url For the 'd' operation, the (HTTP, HTTPS or FTP) URL from which data is to be requested.

ENVIRONMENT VARIABLES

If the TMP environment variable is defined and set to a valid folder name, temporary certificate and PKCS#12 files will be written into that folder using unique file names during the import of PKCS#7 certificate chains and PKCS#12 files. If the TMP environment variable is not defined, or if it is set to the name of a folder that does not exist, the active UPC folder is used for this purpose. Temporary certificate and PKCS#12 files are deleted once they are no longer required.

NOTE: With some versions of Firefox (e.g., 1.5.0.12 and 2.0, but strangely not with 1.5.0.7 or 3.0!) we've found that importing PKCS#7 files can fail with the following error message: "This application has failed to start because plc4.dll was not found." To fix the problem, simply put the Firefox application folder ("c:\Program Files\Mozilla Firefox") into the user's PATH.

EXIT STATUS

The following values are returned:

0 successful completion of all requested operations.

>0 an error occurred or a warning was issued -- not all requested operations completed successfully. See the list below for brief descriptions of all possible error codes. Consult the log file for additional details whenever an error occurs. (Some serious command line errors will cause the program to exit before any log file entries are created; in such cases error messages are written directly to cerr.)

NOTE: Starting with release 2.1, cmu will normally continue processing after the occurrence of non-fatal errors. In such cases, the ERRORLEVEL set upon exit is typically that of the first significant error encountered. The log file must be consulted for any additional errors that might have occurred.

```
#define ERRBASE 1000

enum {
  SUCCESS = 0,

  // command line and configuration file errors
  ERR_CMDLINE = ERRBASE,               // attempt to combine functions or illegal command line
  ERR_OPTION,                          // unsupported or illegal command line option
  ERR_CFG_FILE,                        // configuration file error
  ERR_OS_ERROR,                        // OS or runtime library call failed
  ERR_ADMIN,                           // admin privileges required (-L)

  // user abort and password-related errors
  ERR_USER_ABORT = ERRBASE+100,        // user canceled operation
  ERR_PWD_NOT_SPECIFIED,               // password not specified
  ERR_PWD_INVALID,                     // incorrect password (or unsupported key type)
  ERR_PWD_REJECTED,                    // password fails to meet security requirements
  ERR_PWD_DPAPI_FAILED,                // Windows DPAPI failed to decrypt an encrypted password file

  // general file-related errors
  ERR_FILE_NOT_SPECIFIED = ERRBASE+200, // a required filename not supplied on the command line
  ERR_FILE_NOT_FOUND,                  // specified file not found
  ERR_FILE_EMPTY,                      // specified file must be non-empty
  ERR_FILE_CREATE,                     // cannot create specified output file
  ERR_FILE_OPEN,                       // cannot open file
  ERR_FILE_READ,                       // failed to read as many bytes as expected
  ERR_FILE_WRITE,                      // cannot write file

  // PDU processing errors
  ERR_P12_NOT_SPECIFIED = ERRBASE+300,  // missing or invalid p12 filespec
  ERR_P12_PROCESSING,                  // cannot open or parse p12 file
  ERR_PDU_PARSE,                       // failed to parse PDU
  ERR_PDU_CREATE,                      // cannot create appropriate PDU

  // certificate handling errors
  ERR_UPC = ERRBASE+400,               // cannot find (or create) personal certificate folder
  ERR_CERT_IMPORT,                     // cannot import certificate
  ERR_CERT_EXPORT,                     // cannot export certificate

  // CAPI/MAPI/GAL errors
  ERR_NO_CAPI_CERT = ERRBASE+500,      // no appropriate key pair found in CAPI
  ERR_CAPI_DLL,                        // CAPI DLL not found or unsupported version
  ERR_MAPI_COM,                        // MAPI communications/Outlook configuration error
  ERR_GAL_UPDATE,                      // cannot update GAL entry
  ERR_P7_IMPORT,                       // failed to import into CAPI the contents of a p7 file
  ERR_P12_IMPORT,                      // failed to import into CAPI the contents of a p12 file
  ERR_CAPI_EXPORT,                     // cannot export credentials from CAPI
  ERR_NO_SIGNING_CERT,                 // signing certificate not found in CAPI
  ERR_ISSUER_NOT_FOUND,                // cannot find certificate with specified issuer in CAPI
  ERR_CAPI_CLIENTAUTH,                 // failed to set client authentication properties for one
                                       //   or more signing certificates
  ERR_CAPI_SMIME,                      // cannot set signing and encrypting certificates
  ERR_MAPI_PROFILE,                    // user has no default MAPI security profile
  ERR_MAPI_CERT,                       // user's MAPI profile lack's cert references or references
                                       //   missing certificate or certificate with incorrect AKID
```

```
    ERR_CAPI_CLEAR,                       // could not clear all certs with private keys from CAPI

    // HTTP communication errors
    ERR_URL_NOT_SPECIFIED = ERRBASE+600,  // URL missing
    ERR_HTTP_PARM,                        // URL or capture filespec invalid or not specified
    ERR_HTTP_COM,                         // communications error, authentication error, or URL not found

    // NSS-related errors
    ERR_CLOSE_BROWSER = ERRBASE+700,      // database ops cannot be performed while browser is running
    ERR_NDB_INIT,                         // cannot initialize Netscape databases
    ERR_NDB_IMPORT,                       // cannot import certificates into Netscape databases
    ERR_NDB_EXPORT,                       // cannot export credentials from Netscape databases
    ERR_CERTUTIL,                         // cannot find or execute appropriate certutil.exe
    ERR_PK12UTIL,                         // cannot find or execute appropriate pk12util.exe
    ERR_NDB_BACKUP_CREATE,                // cannot backup database files
    ERR_NSS_PID,                          // cannot obtain PID to terminate Mozilla browser
    ERR_NSS_CERTDB,                       // unsupported cert?.db file (? must be 7,8, or 9)

    // Outlook errors
    ERR_OUTLOOK_PID = ERRBASE+800,        // cannot obtain PID to terminate Outlook
    ERR_OUTLOOK_SMIME,                    // cannot add buttons to Outlook toolbar
    ERR_OUTLOOK_TERMINATE,                // cannot terminate Outlook to change editor preferences
    ERR_OUTLOOK_PREFS,                    // cannot update Outlook user preferences
    ERR_OUTLOOK_QUERY_NOT_FOUND,          // specified query not found in Outlook

    // ISC application errors
    ERR_APP_CONFIG = ERRBASE+900,         // requested application not installed or configuration error

    // audit trail errors
    ERR_AUDIT_TRAIL = ERRBASE+1000,       // auditing system error

    // Acrobat-related errors
    ERR_ACROBAT_ERROR = ERRBASE+1100,

}
```

EXAMPLES

The following command:

```
cmu  i -p MyPassword  mykeys.p12
```

uses 'MyPassword' to decrypt the input PKCS#12 file 'mykeys.p12' and import its key pair into CAPI. (Use '-N' if you prefer to import the user credentials into all supported NSS-based applications installed on the user's system rather than CAPI. In this case, if one or more cert?.db files are found in the default NSS database search paths, the user will be prompted for a database password.) An audit trail of the operations performed by this command is appended to the default log file:
'%AppData%\Information Security Corp\cmu\cmulog.txt'

The command:

```
cmu  i -p "file:c:\tmp\mypassword.txt"  mykeys.p12
```

will perform the same operation, but take the user's password to be the plaintext result of decrypting the contents of the specified `mypassword.txt' file using the Windows DPAPI.

The command:

```
cmu  e  -f  -c c:\tmp\credentials
```

exports the key pairs from all supported and installed applications (even if a Netscape browser is currently running) and stores them as PKCS#12 files in the folder 'c:\tmp\credentials' (which is created if it doesn't already exist). Since a PKCS#12 password is not supplied on the command line, the user will be prompted to enter it at runtime. Similarly, if one or more cert?.db files are found in the default NSS database search paths, the user will also be prompted for a database password. An audit trail of the operations performed by this command is appended to the default log file.

The command:

```
cmu  l
```

prompts the user for his/her PKCS#12 password, then recursively scans the directory tree rooted at the default UPC folder for PKCS#12 files in an attempt to print their friendly names. Friendly names listed in closed braces ([]) actually appear in the corresponding PKCS#12 file. If a PKCS#12 file doesn't contain a friendly name, this command prints in angle brackets (<>) what CAPI thinks the friendly name should be. To view the keyUsage and notAfter attribute values of each certificate, append the '-v' switch to the command line.

The command:

```
cmu  r
```

attempts to locate the user's NSS database folder. If it is found, the user is prompted for a new database password and asked to confirm it. Existing database files are renamed and moved into the UPC, then an empty database with the new password is created in the original location.

The command:

```
cmu  d  https://myserver.com/testpages/getfile.jsp  c:\tmp\testfile.bin
```

downloads the data provided by the referenced URL (in this case, a JSP page) and stores it in the specified file 'c:\tmp\testfile.bin'. Use the '-u' or '-w' options to optionally POST a literal string ('-w') or the contents of a file ('-u') to the specified URL before storing the POST results in the designated output file.

The command:

```
cmu  c  -i F9B20F9778E6D5090F2AC47EBBC6E7AA353C76FB  -z 123
```

configures the user's personal CAPI store so that only the freshest signing certificate issued by the CA with the specified authority key identifier will be used for client

authentication. In addition, prompting for client certificate selection is disabled for internet zones 1, 2 and 3 (intranet, trusted sites, and internet sites not otherwise categorized).

The command:

```
cmu  q  ldap.txt
```

installs the LDAP query defined by parameters in the file 'ldap.txt' into the user's registry so that it is available to them for e-mail addressing purposes in Outlook. The following is a sample parameter definition file:

```
DisplayName=Watergate
ServerName=192.168.0.23
ConnectionPort=389
UserName=workgroup\nixon
SearchBase=cn=users, dc=oakpark, dc=local
CheckNames=(&(mail=*)(|(mail=%s*)(|(cn=%s*)(|(sn=%s*)(givenName=%s*)))))
SearchTimeout=10
MaxEntriesReturned=60
UseSSL=FALSE
UseSPA=FALSE
PromptUser=1
```

For more sample filters and other relevant information, see the references [LDAP1] and [LDAP2].

The command:

```
cmu  w  -d "C:\Documents and Settings\Bush\Application Data\Mozilla\" –R nssfiles.txt
```

recursively scans the D_NETSCAPE and D_MOZILLA directory trees as well as that given by the '-d' argument and then writes the complete pathname of each folder containing an NSS database to the specified output file, 'nssfiles.txt'. The option '-r nssfiles.txt' can then be used with any cmu command to input this file and avoid repeating the search for NSS database files.

The command:

```
cmu  m  -D
```

dumps MAPI security profile information to the log file (and to the screen, if '-v' is used) in the following format:

```
MAPI signing certificate: 286CAA5B02CAC086B8774BEEC53EC9B9DDC947C9
  iCN= 'ISC Test CA'
  mismatched akID= <not present>
  sCN= 'CMU Test User 1'
  nB= 12 JUN 2008 00:00 GMT ; nA= 12 JUN 2018 00:00 GMT
  kU= e8
MAPI encryption certificate: 286CAA5B02CAC086B8774BEEC53EC9B9DDC947C9
  iCN= 'ISC Test CA'
  mismatched akID= <not present>
  sCN= 'CMU Test User 1'
  nB= 12 JUN 2008 00:00 GMT ; nA= 12 JUN 2018 00:00 GMT
  kU= e8
```

SEE ALSO

This section lists some references on credential management issues that we found especially relevant and upon which we relied quite heavily during development of cmu.

How Users Manage Cryptographic Digital IDs in Outlook, Office 2003 Editions Resource Kit, Messaging, Administering Cryptography in Outlook 2003, Microsoft, Sept. 4, 2003.

How To Assign an S/MIME Certificate to a MAPI Profile for Use with Outlook, Microsoft Knowledge Base Article 312900, Revision 2.4, Microsoft, Aug. 25, 2005.

How To Automate Outlook Using Visual C++/MFC, Microsoft Knowledge Base Article 220600, Revision 4.2, Microsoft, Jan. 24, 2007.

How To Modify Recipients of Exchange Global Address List, Microsoft Knowledge Base Article No. 197191, Revision 3.3, Microsoft, Aug. 25, 2005. (MAPI-based approach to modifying GAL entries.)

How to use an Outlook Object Model From Visual C++ by using a #import statement, Microsoft Knowledge Base Article 259298, Revision 3.2, Microsoft, Sept. 9, 2005.

How to programmatically install SSL certificates for Internet Information Server (IIS), Microsoft Knowledge Base Article 313624, Revision 2.1, Microsoft, July 11, 2006.

Common Access Card Setup, DLA On-Line Documentation. (Outlines the process of using Outlook to publish CAC credentials to GAL.)

NSS Security Services, The Mozilla Organization, June 18, 2008. (Detailed information on NSS up through release 3.12.)

Using the Certificate Database Tool, The Mozilla Organization, July 31, 2008. (Documents the command line syntax and usage of the 'certutil' tool used by cmu to maintain NSS 'cert8.db' and 'key3.db' database files.)

Using the PKCS #12 Tool (p12util), The Mozilla Organization, Oct. 13, 2008. (Documents the NSS 'p12util' tool that is used to import (resp. export) key pairs as PKCS#12 files into (resp. from) the NSS databases used by . Related information is available on Sun's USPTO webpage. Although it discusses an extension to 'p12util', HP's page on 'certmig' contains the best usage guide we've been able to find – most of their sample 'certmig' commands work with 'p12util' without change and they actually describe the command line arguments! The failure of 'p12util' to import PKCS12 files without friendly names or DNs is discussed in this mozilla-crypto mailing list entry.)

Henson, Stephen N., Netscape Certificate Database Information and Netscape Communicator Key Database Format, DrH Consultancy, Stoke-on-Trent, Staffordshire, UK.

Luvisetto, M.L., A Brief Guide to Certificate Management, INFN-Bologna, Nov. 18, 2008. (Browser-centric introduction, with some details on the use of OpenSSL.)

[DPAPI] MSDN Library: Windows Data Protection, NAI Labs, Network Associates, Oct. 2001. https://msdn.microsoft.com/en-us/library/ms995355.aspx

[DPAPI-NG] Microsoft Windows Dev Center, CNG DPAPI, May 30, 2018. https://docs.microsoft.com/en-us/windows/desktop/seccng/cng-dpapi

[LDAP1] Configure LDAP options in Outlook 2007, Microsoft TechNet, Jan. 17, 2007.

[LDAP2] Howes, T., The String Representation of LDAP Search Filters, IETF RFC 2254, Dec. 1997.

DIAGNOSTICS

Diagnostic messages are output to the console only for fatal errors. For details on other warnings and errors encountered during execution, the user should consult the active log file. (Use the '-v' switch to include verbose trace information in the diagnostic output.)

BUGS AND LIMITATIONS

There are some rare conditions under which the new DPAPI-encrypted password file support will cause problems when used with NSS databases. For this reason, we suggest avoiding the use of the double quote character '"' in your passwords. You should also not create an encrypted password file in which the password ends with a single backslash '\'. We plan to address this problem in a future release.

The only platforms that we currently support are Windows 8, 8.1 and 10. The supported applications are:

- Microsoft Outlook 2003 and above
- Microsoft Internet Explorer 8.0 and above
- FireFox 2.x and above (use '-R -d' to locate its NSS databases)

Previous versions of cmu supported (and the current version may also support):

- Netscape 4.75 and above, and
- Mozilla 1.1, 1.6, and above
- SecretAgent 5.7.x and above
- SpyProof! 1.4 and above

SecretAgent, SpyProof!, and the Microsoft products are all supported through the standard CAPI certificate stores and Microsoft-supplied Windows runtime libraries, while the Netscape- and Mozilla-based applications are supported through their respective proprietary certificate databases (cert?.db, key?.db) and Network Security Services runtime libraries (NSS 3.2.0, 3.6.1, 3.10.0 and 3_40 using only the 'frozen' NSS 3.4 public functions). Slightly modified versions of the NSS 'certutil' and 'pk12util' tools are bundled into the cmu distribution for this purpose.

A future release of cmu will handle NSS databases in such a way that only the 3_40 tools will be needed, but this will mean dropping support for the older cert7.db files.

The 'u' operation is not useful with SecretAgent 6.x or SpyProof! 6.x as appropriate credential update functionality is already present in those applications.

NOTES

Import

Existing certificates in an NSS database are never overwritten, so using cmu to import a PKCS#7 or PKCS#12 file will not update the friendly name of an existing certificate. Trust attributes of existing CA certificates in the database are never modified, but the trust attribute on an existing end-user certificate will be modified to 'u,pu,u' during import of a PKCS#12 file that contains it.

All new CA certificates found in a PKCS#7 or PKCS#12 input file are imported into NSS certificate databases using the trust attribute 'TC,C,C', which corresponds to manually setting the three trust checkboxes in the Netscape certificate management GUI. New end-user certificates imported from a PKCS#12 file are given a trust attribute of 'u,pu,u'.

The 3.9.0 version of the NSS 'p12util' tool will successfully import PKCS#12 files without friendly names, but it does so by generating very unfriendly substitute names (md5 hash values?). The 3.2.0 and 3.6.1 versions simply fail to import PKCS12 files that do not already have a friendly name. In an attempt to avoid these problems, cmu "invents" a friendly name for each PKCS#12 input file that lacks one using a CAPI call of the form CertGetNameString(…,CERT_NAME_SIMPLE_DISPLAY_TYPE,…). When necessary, it creates a temporary file containing the invented friendly name and the key information from the original file, imports the temporary PKCS#12 file, and then deletes it.)

Export

The specified certificate stores are searched for end-user certificate and private key pairs so that they may be exported as PKCS#12 files into the active UPC folder '%AppData%\Information Security Corp\cmu\' if not overridden using the '-c' switch).

If the '-N' switch is used, the directory search order is:

- the directories listed in the '-r' file argument, if this option is supplied, or
- the D_NETSCAPE and/or D_MOZILLA directory trees
- any directories specified as '-d' arguments (and their subdirectories if '-R' is supplied)

Otherwise key pairs are exported as PKCS#12 files from the user's personal ("MY") CAPI store.

We regard as end-user certificates all those certificates in an NSS store with a trust attribute matching "*,*,*u".

It is recommended that the user run the cmu export operation ('cmu e') upon initial installation of the cmu software package, and whenever a new browser certificate has been obtained. If certificates are not obtained via a browser-based enrollment process, cmu's import (and possibly the synchronize) operation may be all that is required to maintain the user's credentials after an initial discovery run.

Reinitialize

The 'r' ("reinitialize databases") operation functions as follows:

1. directories listed in a '-r' file argument are scanned for 'cert[78].db' files, or if '-r' is omitted, the D_NETSCAPE and/or D_MOZILLA directory trees are recursively scanned for 'cert[78].db' files
2. all '-d' arguments are (recursively, if '-R' is used) searched for 'cert[78].db' files with 'default' appearing in their full pathnames
3. for the first match in each tree, or for all matches if '-a' is specified, cmu
   - moves into the UPC all existing '*.db' files in the folder and renames them with a date and time prefix
   - recreates the NSS database files in their original location with a new password provided via by the '-k' argument or, if '-k' is specified but its optional password argument is omitted, via an interactive prompt (with confirmation)

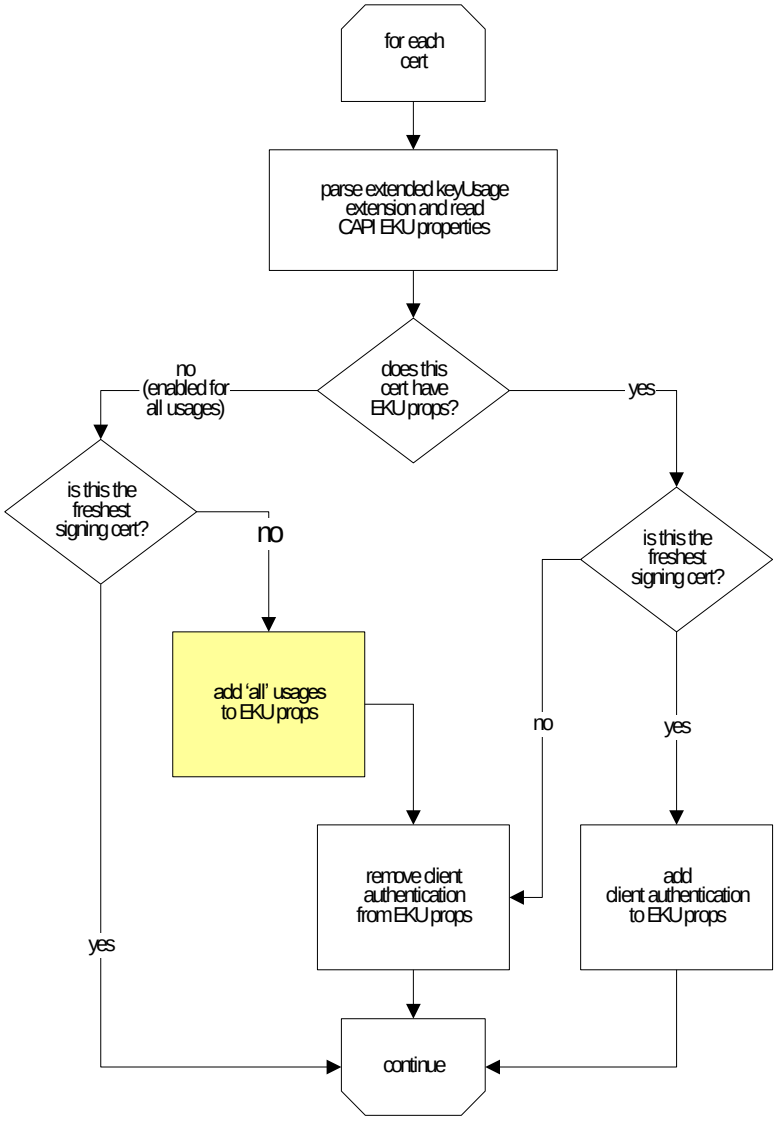
Synchronize

All PKCS#12 files (files ending with '.p12' and '.pfx') found in the directory tree rooted at the active UPC folder ('%AppData%\Information Security Corp\cmu\', if not overridden using the -c switch) are imported into CAPI or, if the '-N' switch is specified, into all NSS certificate stores found on the user's system. Next PKCS#7 files in the UPC are processed in the same way. Finally, PKCS#12 and PKCS#7 files on the command line are imported. (Application certificate stores are located using the same search order as is used for the export operation described above.)

Configure Client Authentication in CAPI

The 'c' operation (when invoked without '-Z') identifies the user's freshest signing certificate in CAPI and makes it the default certificate for client authentication. The freshest certificate is defined as the certificate having the greatest notBefore value. Once it is run, selection of the appropriate certificate will occur automatically each time the user visits a secure website requiring client authentication.

The operation works by manipulating the enhanced key usage (EKU) properties associated with certificates in the user's personal CAPI store as diagrammed in the following flowchart. Certificates containing an enhanced key usage extension (rather than just having an enhanced key usage property) which does not permit client authentication are ignored during processing (as are certificates with issuer authorityKeyIdentifier values that don't match the supplied '-i' argument, if one is specified).



In order to disable usage for client authentication, the following EKU properties are added to all certificates, other than the freshest signing certificate, that initially have no EKU properties. (See yellow box in flowchart.)

```
        "1.3.6.1.4.1.311.10.3.1",       // CTL signing
        "1.3.6.1.4.1.311.10.3.2",       // time stamping
        "1.3.6.1.4.1.311.10.3.4",       // EFS crypto
        "1.3.6.1.4.1.311.10.3.4.1",     // EFS file recovery
        "1.3.6.1.4.1.311.10.3.5",       // WHQL driver verification
        "1.3.6.1.4.1.311.10.3.6",       // NT5 signed
        "1.3.6.1.4.1.311.10.3.7",       // WHQL OEM
        "1.3.6.1.4.1.311.10.3.8",       // embedded NT
        "1.3.6.1.4.1.311.10.3.9",       // signer of CTL containing trusted roots
        "1.3.6.1.4.1.311.10.3.10",      // signer of cross-cert and subordinate CA requests
        "1.3.6.1.4.1.311.10.3.11",      // encrypt/recover escrowed keys
        "1.3.6.1.4.1.311.10.3.12",      // document signer
        "1.3.6.1.4.1.311.10.3.13",

        "1.3.6.1.4.1.311.10.5.1",       // music
        "1.3.6.1.4.1.311.10.6.1",       // licenses
        "1.3.6.1.4.1.311.10.6.2",       // license server
        "1.3.6.1.4.1.311.10.12.1",      // any application policy

        "1.3.6.1.4.1.311.20.2.1",       // enrollment agent
```

```
        "1.3.6.1.4.1.311.20.2.2",       // smartcard logon

        "1.3.6.1.4.1.311.21.5",
        "1.3.6.1.4.1.311.21.6",
        "1.3.6.1.4.1.311.21.19",

        "1.3.6.1.5.5.7.3.1",            // TLS webserver authentication
        "1.3.6.1.5.5.7.3.3",            // code signing
        "1.3.6.1.5.5.7.3.4",            // e-mail protection
        "1.3.6.1.5.5.7.3.5",            // ipsec end system (deprecated by PKIX)
        "1.3.6.1.5.5.7.3.6",            // ipsec tunnel termination (deprecated by PKIX)
        "1.3.6.1.5.5.7.3.7",            // ipsec user (deprecated by PKIX)
        "1.3.6.1.5.5.7.3.8",            // timestamping
        "1.3.6.1.5.5.7.3.9",            // OCSP signing
        "1.3.6.1.5.5.8.2.2",            // PKIX iKEIntermediate (RFC2409)
```

Most of the Microsoft Cryptographic OIDs listed above are documented here; the remaining OIDs are taken from RFC 3280.

If ‘-Z’ is specified, the operation adds the TLS client authentication usage OID ("1.3.6.1.5.5.7.3.2") to the EKU properties of all signing certificates in the user’s personal store that have associated private keys (and a matching issuer AKID or DN, if one is specified with ‘-i’).

As of release 2.1, the ‘c’ operation will avoid DAS certificates only if the ‘-U’ option is specified.

MAPI Security Profile Update

If you specify the 'm' operation the cmu will find the freshest signing and/or encrypting certificates in the user's CAPI store (using the same algorithm as the 'c' operation described above). The freshest certificate is defined as the certificate having the greatest notBefore value. It will then iterate through the user's MAPI profiles looking for their default S/MIME settings. If a default S/MIME profile is found, cmu will modify it to reference these freshest certificates.

If no MAPI security profile exists, cmu will create one with an Outlook display name of 'default' and associate those freshest certificate(s) as the user's default certificate(s) for S/MIME (and 'all other formats').

If '-I' is specified, the MAPI security profile is marked in such a way as to cause Outlook to include the user's certificates in outgoing S/MIME messages. (This is equivalent to checking the "Send certificates with messages" box Outlook's Options settings for encrypted e-mail.) Specify '-O' to uncheck this option. If neither '-I' nor '-O' is specified, an existing default security profile is left unchanged, but if a new one must be created, it is marked to include certificates.

If '-F' is specified, the S/MIME algorithm settings in the user's default MAPI profile are reset to Outlook default values.

Certificates containing an enhanced key usage extension (rather than just having an enhanced key usage property) which does not allow secure e-mail are not considered during the search process.

As of release 2.0, the 'm' operation supports AKID filtering using the '-i' option. As of release 2.1, the 'm' operation will avoid DAS COI certificates only if the '-U' option is specified. As of release 2.6, '-i' also accepts a properly quoted issuer DN.

When a new S/MIME profile is created, cmu 2.2.1 encodes the following algorithm OIDs into its capabilities BLOB:

```
2.16.840.1.101.3.4.1.42       // id-aes256-CBC
2.16.840.1.101.3.4.1.22       // id-aes192-CBC
1.2.840.113549.3.7            // des-ede3-cbc
2.16.840.1.101.3.4.1.2        // id-aes128-CBC
1.3.14.3.2.26                 // id-sha1
2.16.840.1.101.3.4.2.3        // id-sha512
2.16.840.1.101.3.4.2.2        // id-sha384
2.16.840.1.101.3.4.2.1        // id-sha256
```

Versions of cmu prior to 2.2.1 used the following Outlook XP capability OIDs:

```
(1 2 840 113549 3 7)          // des-ede3-cbc
(1 2 840 113549 3 2)          // RC2 CBC (128-bit)
   0080 (128)
(1 2 840 113549 3 2)          // RC2 CBC (64-bit)
   40 (64)
(1 3 14 3 2 7)                // des-ede3-cbc
(1 2 840 113549 3 2)          // RC2 CBC (40-bit)
   28 (40)
(1 3 14 3 2 26)               // id-sha1
(1 2 840 113549 2 5)          // MD5
```

Publish to GAL

The 'publish to GAL' operation ('cmu p [-p<password>] [-s] [p12file]') performs the following tasks in a manner nearly identical to that of Outlook itself:

1. First, it uses the supplied password (from the command line or typed in by the user in response to a runtime prompt) to extract the user's certificate and private key from the supplied PKCS#12 file (or the freshest credentials in CAPI[3], if a PKCS#12 file is not specified on the command line).
2. Next it reads the current userCertificate attribute in the user's Global Address Book entry, if such an attribute is present. If the certificate provided in the PKCS#12 file is *not* already contained in the attribute value, it is appended and user's the GAL entry is updated.
3. Finally, it produces a signed (S/MIME) CMS PDU containing the user's certificate and S/MIME capabilities (namely that the user supports TDES), and stores it into GAL as the user's userSMIMEcertificate attribute (overwriting any existing attribute value).

For convenience (and to simplify the API), steps 2 and 3 use MAPI calls rather than communicating directly with the LDAP front-end on an Exchange Server hosting the GAL.

The ASN.1 dump of a sample CMS PDU that might be produced in step 3 is displayed below. The SEQUENCE specifying the user's sMIMECapabilities, highlighted in red, begins at offset 1345.

```
  0 30 1671: SEQUENCE {
  4 06    9:   OBJECT IDENTIFIER signedData (1 2 840 113549 1 7 2)
 15 A0 1656:   [0] {
 19 30 1652:     SEQUENCE {
 23 02    1:       INTEGER 1
 26 31   11:       SET {
 28 30    9:         SEQUENCE {
 30 06    5:           OBJECT IDENTIFIER sha1 (1 3 14 3 2 26)
 37 05    0:           NULL
           :           }
           :         }
 39 30   25:       SEQUENCE {
 41 06    9:         OBJECT IDENTIFIER data (1 2 840 113549 1 7 1)
 52 A0   12:         [0] {
 54 04   10:           OCTET STRING 'Empty Body'
           :           }
           :         }
 66 A0 1029:       [0] {
 70 30 1025:         SEQUENCE {
 74 30  745:           SEQUENCE {
 78 A0    3:             [0] {
 80 02    1:               INTEGER 2
           :               }
 83 02   20:             INTEGER
           :               14 D8 EC 7F B0 61 E3 BB 78 0A 13 AD 04 50 95 4D
           :               5F 55 61 14
105 30   13:             SEQUENCE {
107 06    9:               OBJECT IDENTIFIER
           :                 sha1withRSAEncryption (1 2 840 113549 1 1 5)
118 05    0:               NULL
           :               }
120 30  102:             SEQUENCE {
122 31   11:               SET {
124 30    9:                 SEQUENCE {
126 06    3:                   OBJECT IDENTIFIER countryName (2 5 4 6)
131 13    2:                   PrintableString 'US'
           :                   }
           :                 }
135 31   11:               SET {
137 30    9:                 SEQUENCE {
139 06    3:                   OBJECT IDENTIFIER stateOrProvinceName (2 5 4 8)
144 13    2:                   PrintableString 'IL'
           :                   }
           :                 }
148 31   35:               SET {
150 30   33:                 SEQUENCE {
```

[3] As of release 2.1, the 'p' operation will avoid DAS certificates only if the '-U' option is specified.

```
152 06    3:                    OBJECT IDENTIFIER organizationName (2 5 4 10)
157 13   26:                    PrintableString 'Information Security Corp.'
           :                    }
           :                  }
185 31   18:                  SET {
187 30   16:                    SEQUENCE {
189 06    3:                      OBJECT IDENTIFIER localityName (2 5 4 7)
194 13    9:                      PrintableString 'Deerfield'
           :                      }
           :                  }
205 31   17:                  SET {
207 30   15:                    SEQUENCE {
209 06    3:                      OBJECT IDENTIFIER commonName (2 5 4 3)
214 13    8:                      PrintableString 'ISC Root'
           :                      }
           :                  }
           :                }
224 30   30:              SEQUENCE {
226 17   13:                UTCTime '030303000000Z'
241 17   13:                UTCTime '050303000019Z'
           :                }
256 30  197:              SEQUENCE {
259 31   11:                SET {
261 30    9:                  SEQUENCE {
263 06    3:                    OBJECT IDENTIFIER countryName (2 5 4 6)
268 13    2:                    PrintableString 'US'
           :                    }
           :                  }
272 31   11:                  SET {
274 30    9:                    SEQUENCE {
276 06    3:                      OBJECT IDENTIFIER stateOrProvinceName (2 5 4 8)
281 13    2:                      PrintableString 'IL'
           :                      }
           :                  }
285 31   34:                  SET {
287 30   32:                    SEQUENCE {
289 06    3:                      OBJECT IDENTIFIER organizationName (2 5 4 10)
294 13   25:                      PrintableString 'Information Security Corp'
           :                      }
           :                  }
321 31   18:                  SET {
323 30   16:                    SEQUENCE {
325 06    3:                      OBJECT IDENTIFIER localityName (2 5 4 7)
330 13    9:                      PrintableString 'Deerfield'
           :                      }
           :                  }
341 31   32:                  SET {
343 30   30:                    SEQUENCE {
345 06    3:                      OBJECT IDENTIFIER title (2 5 4 12)
350 13   23:                      PrintableString 'Director of Development'
           :                      }
           :                  }
375 31   32:                  SET {
377 30   30:                    SEQUENCE {
379 06    3:                      OBJECT IDENTIFIER commonName (2 5 4 3)
384 13   23:                      PrintableString 'Jonathan Schulze-Hewett'
           :                      }
           :                  }
409 31   45:                  SET {
411 30   43:                    SEQUENCE {
413 06    9:                      OBJECT IDENTIFIER
           :                        emailAddress (1 2 840 113549 1 9 1)
424 16   30:                      IA5String 'schulze-hewett@infoseccorp.com'
           :                      }
           :                  }
           :                }
456 30  290:              SEQUENCE {
460 30   13:                SEQUENCE {
462 06    9:                  OBJECT IDENTIFIER
           :                    rsaEncryption (1 2 840 113549 1 1 1)
473 05    0:                  NULL
           :                  }
475 03  271:                BIT STRING 0 unused bits
           :                  30 82 01 0A 02 82 01 01 00 E4 F1 7E C4 5A 8B B1
           :                  44 B3 05 B0 30 DD 81 7D 5E B0 95 95 1C F9 6A CB
           :                  00 57 C5 E2 D3 85 D1 02 70 93 07 55 60 75 F0 EA
           :                  A2 D3 E9 B0 F6 B5 53 64 C5 E0 EE 94 E1 10 C6 83
```

```
           :                 88 D4 9B 5F 86 8E 73 4E 13 F2 B3 3F A5 27 4F 93
           :                 51 58 5B 80 97 F5 CE 96 07 9C 1A 62 AE 30 E2 D5
           :                 B8 49 8C 2F DF 27 62 14 46 C8 AA 65 10 D5 EE 5D
           :                 37 C7 A1 57 46 EE 57 84 B3 66 13 33 E3 01 2C E9
           :                         [ Another 142 bytes skipped ]
           :               }
 750 A3   71:             [3] {
 752 30   69:               SEQUENCE {
 754 30   12:                 SEQUENCE {
 756 06    3:                   OBJECT IDENTIFIER basicConstraints (2 5 29 19)
 761 01    1:                   BOOLEAN TRUE
 764 04    2:                   OCTET STRING
           :                   30 00
           :                   }
 768 30   14:                 SEQUENCE {
 770 06    3:                   OBJECT IDENTIFIER keyUsage (2 5 29 15)
 775 01    1:                   BOOLEAN TRUE
 778 04    4:                   OCTET STRING
           :                   03 02 00 E8
           :                   }
 784 30   37:                 SEQUENCE {
 786 06    3:                   OBJECT IDENTIFIER subjectAltName (2 5 29 17)
 791 04   30:                   OCTET STRING 'schulze-hewett@infoseccorp.com'
           :                   }
           :                 }
           :               }
           :             }
 823 30   13:           SEQUENCE {
 825 06    9:             OBJECT IDENTIFIER
           :               sha1withRSAEncryption (1 2 840 113549 1 1 5)
 836 05    0:             NULL
           :             }
 838 03  257:           BIT STRING 0 unused bits
           :             52 F5 64 A0 B5 71 8C 93 16 DC D5 FA 38 21 9B 8E
           :             14 01 E7 6F 8F D1 97 5A 76 03 A6 3F 76 B5 0D F5
           :             0F 5B 6B FC 85 AA D0 EB AE 86 E2 3E 6D FC A4 0B
           :             3C C8 47 A0 A4 35 B2 2E 83 A5 79 DA 09 16 67 AD
           :             B8 95 58 C5 40 04 9A B7 DC 67 9F 02 09 89 04 CC
           :             95 FE B7 8D 4A 34 23 C9 31 0A 51 2E F0 FC 1A E5
           :             1D 46 DA 7F D1 05 85 0B 7F 06 C4 3B 59 67 C9 26
           :             69 32 3C 68 D7 F4 D5 1A E2 5A 99 8B C5 CA 84 B0
           :                     [ Another 128 bytes skipped ]
           :           }
           :         }
1099 31  572:       SET {
1103 30  568:         SEQUENCE {
1107 02    1:           INTEGER 1
1110 30  126:           SEQUENCE {
1112 30  102:             SEQUENCE {
1114 31   11:               SET {
1116 30    9:                 SEQUENCE {
1118 06    3:                   OBJECT IDENTIFIER countryName (2 5 4 6)
1123 13    2:                   PrintableString 'US'
           :                   }
           :                 }
1127 31   11:               SET {
1129 30    9:                 SEQUENCE {
1131 06    3:                   OBJECT IDENTIFIER stateOrProvinceName (2 5 4 8)
1136 13    2:                   PrintableString 'IL'
           :                   }
           :                 }
1140 31   35:               SET {
1142 30   33:                 SEQUENCE {
1144 06    3:                   OBJECT IDENTIFIER organizationName (2 5 4 10)
1149 13   26:                   PrintableString 'Information Security Corp.'
           :                   }
           :                 }
1177 31   18:               SET {
1179 30   16:                 SEQUENCE {
1181 06    3:                   OBJECT IDENTIFIER localityName (2 5 4 7)
1186 13    9:                   PrintableString 'Deerfield'
           :                   }
           :                 }
1197 31   17:               SET {
1199 30   15:                 SEQUENCE {
1201 06    3:                   OBJECT IDENTIFIER commonName (2 5 4 3)
1206 13    8:                   PrintableString 'ISC Root'
```

```
            :                   }
            :                 }
            :               }
1216 02   20:             INTEGER
            :               14 D8 EC 7F B0 61 E3 BB 78 0A 13 AD 04 50 95 4D
            :               5F 55 61 14
            :             }
1238 30    9:           SEQUENCE {
1240 06    5:             OBJECT IDENTIFIER sha1 (1 3 14 3 2 26)
1247 05    0:             NULL
            :             }
1249 A0  148:           [0] {
1252 30   24:             SEQUENCE {
1254 06    9:               OBJECT IDENTIFIER
            :                 contentType (1 2 840 113549 1 9 3)
1265 31   11:               SET {
1267 06    9:                 OBJECT IDENTIFIER data (1 2 840 113549 1 7 1)
            :                 }
            :               }
1278 30   28:             SEQUENCE {
1280 06    9:               OBJECT IDENTIFIER
            :                 signingTime (1 2 840 113549 1 9 5)
1291 31   15:               SET {
1293 17   13:                 UTCTime '040526151102Z'
            :                 }
            :               }
1308 30   35:             SEQUENCE {
1310 06    9:               OBJECT IDENTIFIER
            :                 messageDigest (1 2 840 113549 1 9 4)
1321 31   22:               SET {
1323 04   20:                 OCTET STRING
            :                   68 DD 2D D6 34 98 03 66 79 AD 90 81 C6 9A 56 4A
            :                   4A 6A 0E 1A
            :                 }
            :               }
1345 30   53:             SEQUENCE {
1347 06    9:               OBJECT IDENTIFIER
            :                 sMIMECapabilities (1 2 840 113549 1 9 15)
1358 31   40:               SET {
1360 30   38:                 SEQUENCE {
1362 30   10:                   SEQUENCE {
1364 06    8:                     OBJECT IDENTIFIER
            :                       des-EDE3-CBC (1 2 840 113549 3 7)
            :                     }
1374 30   13:                   SEQUENCE {
1376 06    8:                     OBJECT IDENTIFIER rc2CBC (1 2 840 113549 3 2)
1386 02    1:                     INTEGER 40
            :                     }
1389 30    9:                   SEQUENCE {
1391 06    7:                     OBJECT IDENTIFIER '0 6 5 43 14 3 2 29'
            :                     }
            :                   }
            :                 }
            :               }
1400 30   13:             SEQUENCE {
1402 06    9:               OBJECT IDENTIFIER
            :                 rsaEncryption (1 2 840 113549 1 1 1)
1413 05    0:               NULL
            :               }
1415 04  256:             OCTET STRING
            :               CC 65 2D DA A8 F0 35 2C F8 D5 4D DD CB 42 48 B4
            :               61 1D 66 0F E8 1A C4 B5 95 ED AB B3 07 2E B9 C8
            :               8A BC 09 CB A4 B4 A2 64 FA 95 DC 2D E6 53 C4 AF
            :               32 2E A6 5D 3F 30 D7 BC 65 81 BB 0E 03 9C F8 A6
            :               C2 BC AD E1 52 FC FF D8 7F D8 6B 74 A4 AA B7 21
            :               BD 0D CE 84 01 19 56 F4 FB 50 0D 1A 3E 19 CF E1
            :               2D D1 D1 EE A4 F3 05 30 50 6D B8 FD B0 95 35 E4
            :               33 FB A6 82 90 65 55 CB 47 76 43 5B 84 11 CA 73
            :                       [ Another 128 bytes skipped ]
            :           }
            :         }
            :       }
            :     }
            :   }
```

```
/*************************************************************************

cpf4cmu

Copyright© 1991-2018 Information Security Corp. All rights reserved.

*************************************************************************/
```

USER COMMANDS **cpf4cmu(1)**

NAME

cpf4cmu – create password file for cmu

SYNOPSIS

```
cpf4cmu [-h | --help]

cpf4cmu [[-t | --title] t_str] [[-p | --prompt] p_str] [[-l | --label] l_str] \
   [[-m | --minlen] len] [[-s | --strength] s_str]  pwd_file
```

DESCRIPTION

This utility prompts the user to enter a password, encrypts it using the Windows Data Protection API (DPAPI) function CryptProtectData(), and stores the encrypted version of the password in a specified file on the user's system. Its command line syntax is summarized above.

The first form simply outputs a command line summary. The second form displays the password entry dialog that the user completes to create the protected password file. In the second form, the only required command line parameter is *pwd_file*, a complete pathname for the output file to be created. (NOTE: If the specified file already exists, it is overwritten without warning!)

OPTIONS

The -h option causes cpf4cmu to display copyright and command line syntax information and exit with a return code of 0:

```
cpf4cmu - create password file for cmu, Version 1.0.0
          (ISC CDK version 8.0.0.7)
(c)1991-2018 Information Security Corp. All rights reserved.

Usage:
   cpf4cmu [-h]
   cpf4cmu [-t t_str] [-p p_str] [-l l_str] [-m len] [-s s_str] pfile

Options:
    -h, --help      display this command line syntax information
    -t, --title     set title of password dialog
    -p, --prompt    set prompt above password entry box
    -l, --label     set label on password entry box
    -m, --minlen    set minimum password length
    -s, --strength  set password strength requirements (w/ confirmation)

Arguments:
    t_str     title of password dialog
    p_str     prompt above password entry box
    l_str     label on password entry box
    len       minimum accepted password length
    s_str     password strength mask (see documentation)
```

If the program detects invalid command line arguments, it may or may not display the same syntax information (depending on the nature of the error), but it will exis with a return code of `ERR_CMD_LINE` (1; see section on error codes below)). If the user cancels the password entry dialog, nothing is written to *stdout* or *stderr*, but the program exits with a return code of `RET_CANCELED (-1)`. Other possible return codes are listed in the following section.

If the -s (or --strength) option appears, the password must be entered twice, both inputs must agree and satisfy the constraints represented by *s_str* (explained below); otherwise there is only one password input field, but -m may still be specified to set the minimum acceptable password length.

The remaining optional input parameters include a window title for the password dialog (-t *t_str*), prompt text to be displayed above the password input field(s) (-p *p_str*), and text to be used as the label on that box (-l *l_str*), along with a minimum password length setting (-m *len*) that applies to both forms of the main dialog.

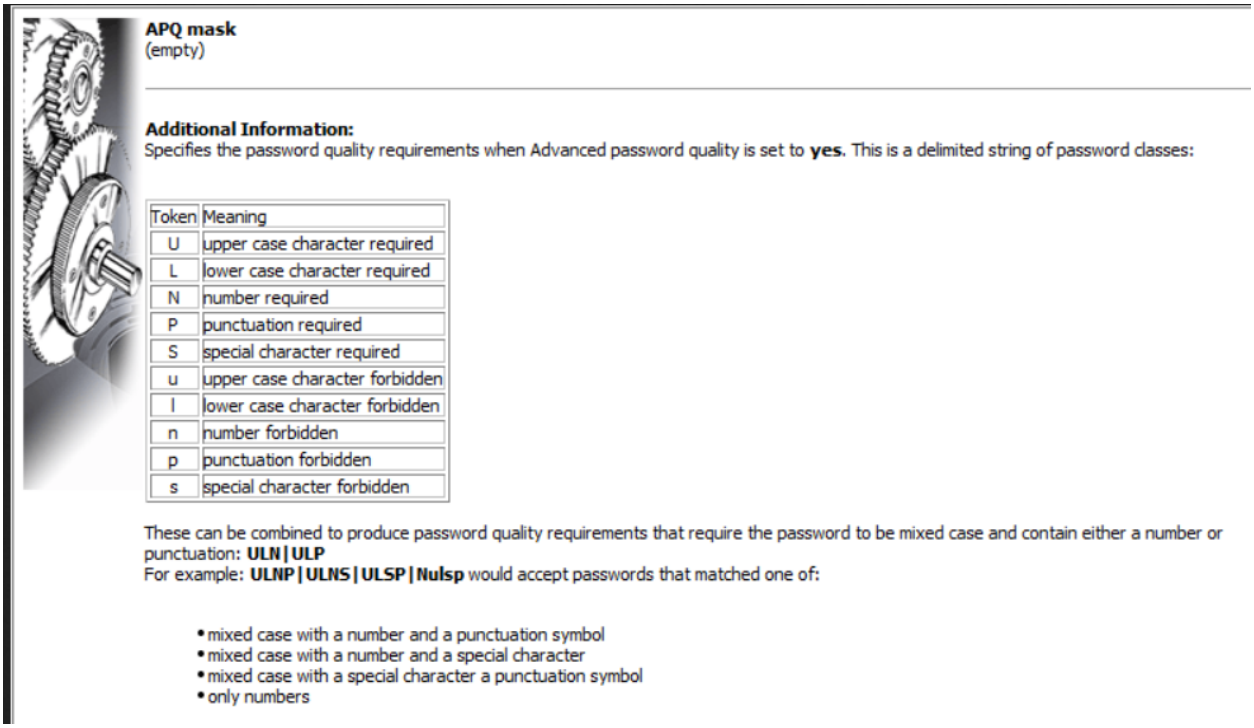
If any of the first three optional parameters are omitted, default text will be used as described in the following table (and as illustrated in the sample screenshots below):

dialog element	**default value**	**overriding command line parameter**
window title	“Create password file”	-t
prompt text	“Output path: *pwd_file*”	-p
box label	“Enter password to be encrypted:”	-l

Note that the arguments *t_str*, *p_str*, and *l_str* all support HTML formatting commands; see the final screenshot below to get an idea of what is possible.

PASSWORD STRENGTH

The argument for the password strength specification, *s_str*, must be a series of words separated by vertical bars ('|', representing a disjunctive 'OR'). Each word must consist of one or more characters chosen from the following table of tokens labeling their respective character classes (as defined in Bagala); white space is not allowed. The table also indicates the semantics associated with each token.



APQ mask
(empty)

Additional Information:
Specifies the password quality requirements when Advanced password quality is set to **yes**. This is a delimited string of password classes:

Token	Meaning
U	upper case character required
L	lower case character required
N	number required
P	punctuation required
S	special character required
u	upper case character forbidden
l	lower case character forbidden
n	number forbidden
p	punctuation forbidden
s	special character forbidden

These can be combined to produce password quality requirements that require the password to be mixed case and contain either a number or punctuation: **ULN|ULP**
For example: **ULNP|ULNS|ULSP|Nulsp** would accept passwords that matched one of:

- mixed case with a number and a punctuation symbol
- mixed case with a number and a special character
- mixed case with a special character a punctuation symbol
- only numbers

If a compound *s_str* argument is used, the entire string must be double quoted; for example, the command line option: -s "ULN|ULP" means the password must contain at least one upper case letter, at least one lower case letter, and either a decimal digit or a punctuation symbol.

When used with the -s option, **cpf4cmu** behaves like **cspid_cli --get-new-pin** in the sense that the dialog displays a password confirmation field and requires it to be completed and match the password field. Without the -s option, **cpf4cmu** behaves like **cspid_cli --get-pin** and the confirmation field is not displayed.

EXIT STATUS

The following values may be returned:

-1 user cancelled or closed the password entry dialog without clicking OK.

0 successful completion of all requested operations.

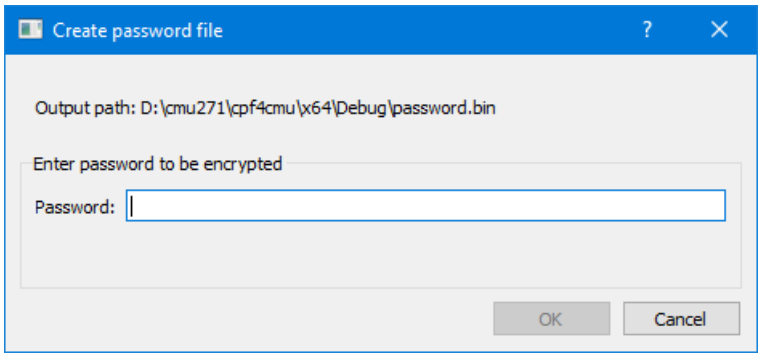
>0 an error occurred or a warning was issued -- not all requested operations completed successfully. See the list below for brief descriptions of the possible error codes:

```
enum {
        RET_CANCELED = -1,     // user canceled dialog
        RET_OK,                // used throughout to signal success
        ERR_CMD_LINE,          // error parsing command line
        ERR_MASK,              // error parsing or applying password strength mask
        ERR_WRITE,             // error writing output file
        ERR_ENCODE,            // error encrypting password with Windows DPAPI
};
```

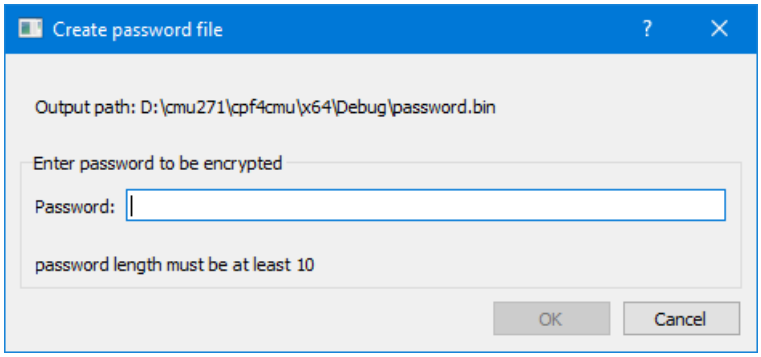
NOTE: In some situations, such as file permissions or write errors, cpf4cmu may also return the standard Windows error code associated with that condition.

SCREENSHOTS

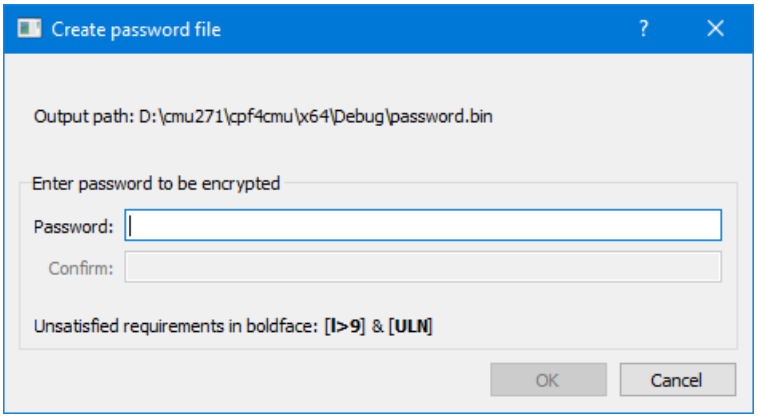
The password entry dialog will appear as:



when neither '-m' nor '-s' options are specified; as:



when '-m 10' is used without a '-s' parameter; or as:

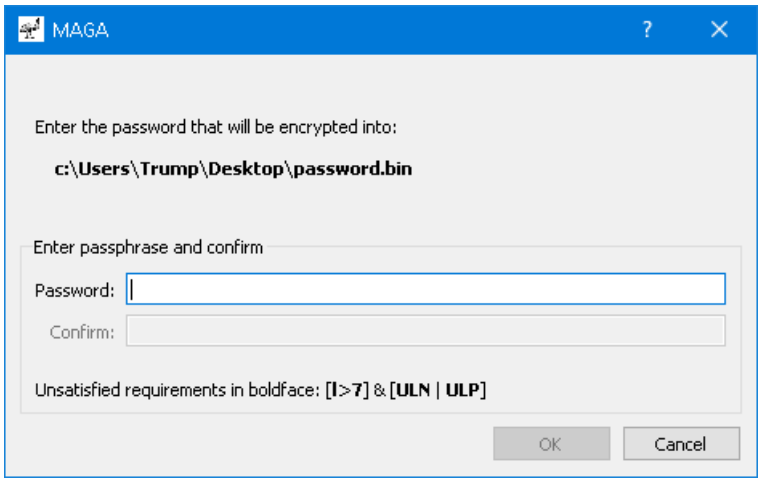


if '-m 10 -s ULN' is used.

To illustrate what is possible, we show how to make a password entry dialog with a custom window title and box label, a custom prompt containing the output path, and a compound password strength requirement:

```
cpf4cmu -t "MAGA" -p "Enter the password that will be encrypted into:          \
<br /><br /><b>    c:\Users\Trump\Desktop\password.bin"    \
-l "Enter passphrase and confirm” -m 8 -s "ULN|ULP"                           \
"c:\Users\Donald Trump\Desktop\password.bin"
```

This results in the following dialog:



Given the complexity of the real-time update functionality of the status line (“Unsatisfied requirements…”), we did not find it feasible to make that dialog element customizable in this release.