NAME

p11tool - GnuTLS PKCS #11 tool

SYNOPSIS

p11tool [-**flags**] [-**flag** [*value*]] [--**option-name**[[=|]*value*]] [url]

Operands and options may be intermixed. They will be reordered.

DESCRIPTION

Program that allows operations on PKCS #11 smart cards and security modules.

To use PKCS #11 tokens with GnuTLS the p11-kit configuration files need to be setup. That is create a .module file in /usr/local/share/p11-kit/modules with the contents ’module: /path/to/pkcs11.so’. Alternatively the configuration file /usr/local/etc/gnutls/pkcs11.conf has to exist and contain a number of lines of the form ’load=/usr/lib/opensc-pkcs11.so’.

You can provide the PIN to be used for the PKCS #11 operations with the environment variables GNUTLS_PIN and GNUTLS_SO_PIN.

OPTIONS

Tokens

--list-tokens

List all available tokens.

--list-token-urls

List the URLs available tokens.

This is a more compact version of --list-tokens.

--list-mechanisms

List all available mechanisms in a token.

--initialize

Initializes a PKCS #11 token.

--initialize-pin

Initializes/Resets a PKCS #11 token user PIN.

--initialize-so-pin
Initializes/Resets a PKCS #11 token security officer PIN.

This initializes the security officer's PIN. When used non-interactively use the GNUTLS_NEW_SO_PIN environment variables to initialize SO's PIN.

--set-pin=*str*
Specify the PIN to use on token operations.

Alternatively the GNUTLS_PIN environment variable may be used.

--set-so-pin=*str*
Specify the Security Officer's PIN to use on token initialization.

Alternatively the GNUTLS_SO_PIN environment variable may be used.

Object listing

--list-all
List all available objects in a token.

All objects available in the token will be listed. That includes objects which are potentially unaccessible using this tool.

--list-all-certs
List all available certificates in a token.

That option will also provide more information on the certificates, for example, expand the attached extensions in a trust token (like p11-kit-trust).

--list-certs
List all certificates that have an associated private key.

That option will only display certificates which have a private key associated with them (share the same ID).

--list-all-privkeys
List all available private keys in a token.

Lists all the private keys in a token that match the specified URL.

--list-privkeys
This is an alias for the --*list-all-privkeys* option.

--list-keys
This is an alias for the --*list-all-privkeys* option.

--list-all-trusted
List all available certificates marked as trusted.

--export
Export the object specified by the URL. This option must not appear in combination with any of the following options: export-stapled, export-chain, export-pubkey.

--export-stapled
Export the certificate object specified by the URL. This option must not appear in combination with any of the following options: export, export-chain, export-pubkey.

Exports the certificate specified by the URL while including any attached extensions to it. Since attached extensions are a p11-kit extension, this option is only available on p11-kit registered trust modules.

--export-chain
Export the certificate specified by the URL and its chain of trust. This option must not appear in combination with any of the following options: export-stapled, export, export-pubkey.

Exports the certificate specified by the URL and generates its chain of trust based on the stored certificates in the module.

--export-pubkey
Export the public key for a private key. This option must not appear in combination with any of the following options: export-stapled, export, export-chain.

Exports the public key for the specified private key

--info
List information on an available object in a token.

--trusted

This is an alias for the --*mark-trusted* option.

--distrusted

This is an alias for the --*mark-distrusted* option.

Key generation

--generate-privkey=*str*

Generate private-public key pair of given type.

Generates a private-public key pair in the specified token. Acceptable types are RSA, ECDSA, Ed25519, and DSA. Should be combined with --sec-param or --bits.

--generate-rsa

Generate an RSA private-public key pair.

Generates an RSA private-public key pair on the specified token. Should be combined with --sec-param or --bits.

NOTE: THIS OPTION IS DEPRECATED

--generate-dsa

Generate a DSA private-public key pair.

Generates a DSA private-public key pair on the specified token. Should be combined with --sec-param or --bits.

NOTE: THIS OPTION IS DEPRECATED

--generate-ecc

Generate an ECDSA private-public key pair.

Generates an ECDSA private-public key pair on the specified token. Should be combined with --curve, --sec-param or --bits.

NOTE: THIS OPTION IS DEPRECATED

--bits=*num*

Specify the number of bits for the key generate. This option takes an integer number as its argument.

For applications which have no key-size restrictions the --sec-param option is recommended, as the sec-param levels will adapt to the acceptable security levels with the new versions of gnutls.

--curve=*str*
Specify the curve used for EC key generation.

Supported values are secp192r1, secp224r1, secp256r1, secp384r1 and secp521r1.

--sec-param=*security parameter*
Specify the security level.

This is alternative to the bits option. Available options are [low, legacy, medium, high, ultra].

Writing objects

--set-id=*str*
Set the CKA_ID (in hex) for the specified by the URL object. This option must not appear in combination with any of the following options: write.

Modifies or sets the CKA_ID in the specified by the URL object. The ID should be specified in hexadecimal format without a ’0x’ prefix.

--set-label=*str*
Set the CKA_LABEL for the specified by the URL object. This option must not appear in combination with any of the following options: write, set-id.

Modifies or sets the CKA_LABEL in the specified by the URL object

--write
Writes the loaded objects to a PKCS #11 token.

It can be used to write private, public keys, certificates or secret keys to a token. Must be combined with one of --load-privkey, --load-pubkey, --load-certificate option.

When writing a certificate object, its CKA_ID is set to the same CKA_ID of the corresponding public key, if it exists on the token; otherwise it will be derived from the X.509 Subject Key Identifier of the certificate. If this behavior is undesired, write the public key to the token beforehand.

--delete
Deletes the objects matching the given PKCS #11 URL.

--label=*str*
Sets a label for the write operation.

--id=*str*
Sets an ID for the write operation.

Sets the CKA_ID to be set by the write operation. The ID should be specified in hexadecimal format without a ’0x’ prefix.

--mark-wrap, **--no-mark-wrap**
Marks the generated key to be a wrapping key. The *no-mark-wrap* form will disable the option.

Marks the generated key with the CKA_WRAP flag.

--mark-trusted, **--no-mark-trusted**
Marks the object to be written as trusted. This option must not appear in combination with any of the following options: mark-distrusted. The *no-mark-trusted* form will disable the option.

Marks the object to be generated/written with the CKA_TRUST flag.

--mark-distrusted
When retrieving objects, it requires the objects to be distrusted (blacklisted). This option must not appear in combination with any of the following options: mark-trusted.

Ensures that the objects retrieved have the CKA_X_TRUST flag. This is p11-kit trust module extension, thus this flag is only valid with p11-kit registered trust modules.

--mark-decrypt, **--no-mark-decrypt**
Marks the object to be written for decryption. The *no-mark-decrypt* form will disable the option.

Marks the object to be generated/written with the CKA_DECRYPT flag set to true.

--mark-sign, **--no-mark-sign**
Marks the object to be written for signature generation. The *no-mark-sign* form will disable the option.

Marks the object to be generated/written with the CKA_SIGN flag set to true.

--mark-ca, **--no-mark-ca**

Marks the object to be written as a CA. The *no-mark-ca* form will disable the option.

Marks the object to be generated/written with the CKA_CERTIFICATE_CATEGORY as CA.

--mark-private, **--no-mark-private**

Marks the object to be written as private. The *no-mark-private* form will disable the option.

Marks the object to be generated/written with the CKA_PRIVATE flag. The written object will require a PIN to be used.

--ca

This is an alias for the --*mark-ca* option.

--private

This is an alias for the --*mark-private* option.

--mark-always-authenticate, **--no-mark-always-authenticate**

Marks the object to be written as always authenticate. The *no-mark-always-authenticate* form will disable the option.

Marks the object to be generated/written with the CKA_ALWAYS_AUTHENTICATE flag. The written object will Mark the object as requiring authentication (pin entry) before every operation.

--secret-key=*str*

Provide a hex encoded secret key.

This secret key will be written to the module if --write is specified.

--load-privkey=*file*

Private key file to use.

--load-pubkey=*file*

Public key file to use.

--load-certificate=*file*

Certificate file to use.

Other options

-d *num*, **--debug**=*num*

Enable debugging. This option takes an integer number as its argument. The value of *num* is constrained to being:

in the range 0 through 9999

Specifies the debug level.

--outfile=*str*

Output file.

--login, **--no-login**

Force (user) login to token. The *no-login* form will disable the option.

--so-login, **--no-so-login**

Force security officer login to token. The *no-so-login* form will disable the option.

Forces login to the token as security officer (admin).

--admin-login

This is an alias for the *--so-login* option.

--test-sign

Tests the signature operation of the provided object.

It can be used to test the correct operation of the signature operation. If both a private and a public key are available this operation will sign and verify the signed data.

--sign-params=*str*

Sign with a specific signature algorithm.

This option can be combined with --test-sign, to sign with a specific signature algorithm variant. The only option supported is ’RSA-PSS’, and should be specified in order to use RSA-PSS signature on RSA keys.

--hash=*str*

Hash algorithm to use for signing.

This option can be combined with test-sign. Available hash functions are SHA1, RMD160, SHA256, SHA384, SHA512, SHA3-224, SHA3-256, SHA3-384, SHA3-512.

--generate-random=*num*
Generate random data. This option takes an integer number as its argument.

Asks the token to generate a number of bytes of random bytes.

-8, **--pkcs8**
Use PKCS #8 format for private keys.

--inder, **--no-inder**
Use DER/RAW format for input. The *no-inder* form will disable the option.

Use DER/RAW format for input certificates and private keys.

--inraw
This is an alias for the **--***inder* option.

--outder, **--no-outder**
Use DER format for output certificates, private keys, and DH parameters. The *no-outder* form will disable the option.

The output will be in DER or RAW format.

--outraw
This is an alias for the **--***outder* option.

--provider=*file*
Specify the PKCS #11 provider library.

This will override the default options in /usr/local/etc/gnutls/pkcs11.conf

--provider-opts=*str*
Specify parameters for the PKCS #11 provider library.

This is a PKCS#11 internal option used by few modules.
Mainly for testing PKCS#11 modules.

NOTE: THIS OPTION IS DEPRECATED

--detailed-url, **--no-detailed-url**
Print detailed URLs. The *no-detailed-url* form will disable the option.

--only-urls
Print a compact listing using only the URLs.

--batch
Disable all interaction with the tool.

In batch mode there will be no prompts, all parameters need to be specified on command line.

-v *arg*, **--version**=*arg*
Output version of program and exit. The default mode is ‘v’, a simple version. The ‘c’ mode will print copyright information and ‘n’ will print the full copyright notice.

-h, **--help**
Display usage information and exit.

-!, **--more-help**
Pass the extended usage information through a pager.

EXAMPLES

To view all tokens in your system use:

```
 $ p11tool --list-tokens
```

To view all objects in a token use:

```
 $ p11tool --login --list-all "pkcs11:TOKEN-URL"
```

To store a private key and a certificate in a token run:

```
 $ p11tool --login --write "pkcs11:URL" --load-privkey key.pem           --label "Mykey"
 $ p11tool --login --write "pkcs11:URL" --load-certificate cert.pem           --label "Mykey"
```

Note that some tokens require the same label to be used for the certificate and its corresponding private key.

To generate an RSA private key inside the token use:

```
$ p11tool --login --generate-privkey rsa --bits 1024 --label "MyNewKey"           --outfile MyNewKey.pub "pkcs11
```

The bits parameter in the above example is explicitly set because some tokens only support limited choices in the bit length. The output file is the corresponding public key. This key can be used to general a certificate request with certtool.

```
certtool --generate-request --load-privkey "pkcs11:KEY-URL"    --load-pubkey MyNewKey.pub --outfile request.
```

EXIT STATUS

One of the following exit values will be returned:

0 (EXIT_SUCCESS)
: Successful program execution.

1 (EXIT_FAILURE)
: The operation failed or the command syntax was not valid.

SEE ALSO

certtool (1)

AUTHORS

COPYRIGHT

BUGS

Please send bug reports to: bugs@gnutls.org