Rocky Enterprise Linux 9.2 Manual Pages on command 'tpm2_createprimary.1'

$ man tpm2_createprimary.1

tpm2_createprimary(1) General Commands Manual tpm2_createprimary(1)

NAME

tpm2_createprimary(1) - Create a primary key.

SYNOPSIS

tpm2_createprimary [OPTIONS]

DESCRIPTION

tpm2_createprimary(1) - This command is used to create a primary object under one of the hierarchies: Owner, Platform, Endorsement, NULL. The command will create and load a Primary Object. The sensitive and pub? lic portions are not returned. A context file for the created object?s handle is saved as a file for future interactions with the created pri? mary.

OPTIONS

? -C, --hierarchy=OBJECT:

The hierarchy under which the object is created. This will also dic? tate which authorization secret (if any) must be supplied. Defaults to TPM_RH_OWNER, when no value specified. Supported options are:

? o for TPM_RH_OWNER

? p for TPM_RH_PLATFORM

? e for TPM_RH_ENDORSEMENT

? n for TPM_RH_NULL

? <num> where a raw number can be used.

? -P, --hierarchy-auth=AUTH:

The authorization value for the hierarchy specified with -C.

? -p, --key-auth=AUTH:

The authorization value for the primary object created.

? -g, --hash-algorithm=ALGORITHM:

The hash algorithm to use for generating the objects name. Defaults to sha256 if not specified.

? -G, --key-algorithm=ALGORITHM:

The algorithm type for the generated primary key. Defaults to rsa2048:null:aes128cfb.

? -c, --key-context=FILE:

The file path to save the object context of the generated primary ob?ject.

? -L, --policy=FILE:

An optional file input that contains the policy digest for policy based authorization of the object.

? -a, --attributes=ATTRIBUTES:

The object attributes, optional. Defaults to: TPMA_OBJECT_RESTRICT?ED|TPMA_OBJECT_DECRYPT|TPMA_OBJECT_FIXEDTPM| TPMA_OBJECT_FIXEDPAR?ENT|TPMA_OBJECT_SENSITIVEDATAORIGIN| TPMA_OBJECT_USERWITHAUTH

? -u, --unique-data=FILE OR STDIN:

An optional file input that contains the unique field of TPMT_PUBLIC in little-endian format. Primary key creator may place information that causes the primary key generation scheme internal to the TPM to generate statistically unique values. The TPM v2.0 specification calls this field unique and overloads it so that it contains one val?ue when the application provides this structure as input and another value when the applications receives this structure as output (like public portion of the rsa key).

If the data is specified as a file, the user is responsible for en?
suring that this buffer is formatted per TPMU_PUBLIC_ID union.
The unique data can also be retrieved from stdin buffer by specifying
?-? as the ?unique-data option value and the tool will parse the key
type and associate the input data with the unique data buffer associ?
ated with the key type. NOTE:

1. The maximum allowed bytes is dependent on key type and the TPM im?
plementation. Eg. While TSS allows a value upto 512 for
MAX_RSA_KEY_BYTES, however the ibmSwTPM implementation supports a
value upto 256 bytes.

2. The unique input data specified on stdin for ECC is split for
specifying the X coordinate and Y coordinate buffers.

? --creation-data=FILE:
An optional file output that saves the creation data for certifica?
tion.

? --template-data=FILE:
An optional file output that saves the key template data (TPM2B_PUB?
LIC) to be used in tpm2_policytemplate.

? -t, --creation-ticket=FILE:
An optional file output that saves the creation ticket for certifica?
tion.

? -d, --creation-hash=FILE:
An optional file output that saves the creation hash for certifica?
tion.

? -q, --outside-info=FILE_OR_HEX:
An optional file or hex string to add unique data to the creation da?
ta. Note that it does not contribute in creating statistically
unique object.

? -l, --pcr-list=PCR:
The list of PCR banks and selected PCRs? ids for each bank to be in?
cluded in the creation data for certification.

? --cphash=FILE
File path to record the hash of the command parameters. This is com?

monly termed as cpHash. NOTE: When this option is selected, The tool will not actually execute the command, it simply returns a cpHash.

? -f, --format:

Format selection for the public key output file. `tss' (the default) will output a binary blob according to the TPM 2.0 Specification. `pem' will output an OpenSSL compatible PEM encoded public key. `der' will output an OpenSSL compatible DER encoded public key. `tpmt' will output a binary blob of the TPMT_PUBLIC struct referenced by TPM 2.0 specs.

Public key format.

? -o, --output=FILE:

The output file path, recording the public portion of the object.

References

Context Object Format

The type of a context object, whether it is a handle or file name, is determined according to the following logic in-order:

? If the argument is a file path, then the file is loaded as a restored TPM transient object.

? If the argument is a prefix match on one of:

? owner: the owner hierarchy

? platform: the platform hierarchy

? endorsement: the endorsement hierarchy

? lockout: the lockout control persistent object

? If the argument argument can be loaded as a number it will be treat as a handle, e.g. 0x81010013 and used directly._OBJECT_.

Authorization Formatting

Authorization for use of an object in TPM2.0 can come in 3 different forms: 1. Password 2. HMAC 3. Sessions

NOTE: ?Authorizations default to the EMPTY PASSWORD when not speci? fied?.

Passwords

Passwords are interpreted in the following forms below using prefix identifiers.

Note: By default passwords are assumed to be in the string form when they do not have a prefix.

String

A string password, specified by prefix ?str:? or it?s absence (raw string without prefix) is not interpreted, and is directly used for au? thorization.

Examples

foobar

str:foobar

Hex-string

A hex-string password, specified by prefix ?hex:? is converted from a hexidecimal form into a byte array form, thus allowing passwords with non-printable and/or terminal un-friendly characters.

Example

hex:0x1122334455667788

File

A file based password, specified be prefix ?file:? should be the path of a file containing the password to be read by the tool or a ?-? to use stdin. Storing passwords in files prevents information leakage, passwords passed as options can be read from the process list or common shell history features.

Examples

```
# to use stdin and be prompted
file:-
# to use a file from a path
file:path/to/password/file
# to echo a password via stdin:
echo foobar | tpm2_tool -p file:-
# to use a bash here-string via stdin:
tpm2_tool -p file:- <<< foobar
```

Sessions

When using a policy session to authorize the use of an object, prefix the option argument with the session keyword. Then indicate a path to

a session file that was created with tpm2_startauthsession(1). Option?
ally, if the session requires an auth value to be sent with the session
handle (eg policy password), then append a + and a string as described
in the Passwords section.

Examples

To use a session context file called session.ctx.

session:session.ctx

To use a session context file called session.ctx AND send the authvalue
mypassword.

session:session.ctx+mypassword

To use a session context file called session.ctx AND send the HEX auth?
value 0x11223344.

session:session.ctx+hex:11223344

PCR Authorizations

You can satisfy a PCR policy using the ?pcr:? prefix and the PCR mini?
language. The PCR minilanguage is as follows:
<pcr-spec>=<raw-pcr-file>

The PCR spec is documented in in the section ?PCR bank specifiers?.

The raw-pcr-file is an optional argument that contains the output of
the raw PCR contents as returned by tpm2_pcrread(1).

PCR bank specifiers (pcr.md)

Examples

To satisfy a PCR policy of sha256 on banks 0, 1, 2 and 3 use a specifi?
er of:

pcr:sha256:0,1,2,3

specifying AUTH.

Algorithm Specifiers

Options that take algorithms support ?nice-names?.

There are two major algorithm specification string classes, simple and
complex. Only certain algorithms will be accepted by the TPM, based on
usage and conditions.

Simple specifiers

These are strings with no additional specification data. When creating

objects, non-specified portions of an object are assumed to defaults.

You can find the list of known ?Simple Specifiers Below?.

Asymmetric

? rsa

? ecc

Symmetric

? aes

? camellia

Hashing Algorithms

? sha1

? sha256

? sha384

? sha512

? sm3_256

? sha3_256

? sha3_384

? sha3_512

Keyed Hash

? hmac

? xor

Signing Schemes

? rsassa

? rsapss

? ecdsa

? ecdaa

? ecschnorr

Asymmetric Encryption Schemes

? oaep

? rsaes

? ecdh

Modes

? ctr

? ofb

? cbc

? cfb

? ecb

Misc

? null

Complex Specifiers

Objects, when specified for creation by the TPM, have numerous algo? rithms to populate in the public data. Things like type, scheme and asymmetric details, key size, etc. Below is the general format for specifying this data: <type>:<scheme>:<symmetric-details>

Type Specifiers

This portion of the complex algorithm specifier is required. The re? maining scheme and symmetric details will default based on the type specified and the type of the object being created.

? aes - Default AES: aes128

? aes128<mode> - 128 bit AES with optional mode (ctr|ofb|cbc|cfb|ecb). If mode is not specified, defaults to null.

? aes192<mode> - Same as aes128<mode>, except for a 192 bit key size.

? aes256<mode> - Same as aes128<mode>, except for a 256 bit key size.

? ecc - Elliptical Curve, defaults to ecc256.

? ecc192 - 192 bit ECC

? ecc224 - 224 bit ECC

? ecc256 - 256 bit ECC

? ecc384 - 384 bit ECC

? ecc521 - 521 bit ECC

? rsa - Default RSA: rsa2048

? rsa1024 - RSA with 1024 bit keysize.

? rsa2048 - RSA with 2048 bit keysize.

? rsa4096 - RSA with 4096 bit keysize.

Scheme Specifiers

Next, is an optional field, it can be skipped.

Schemes are usually Signing Schemes or Asymmetric Encryption Schemes.

Most signing schemes take a hash algorithm directly following the sign?

ing scheme. If the hash algorithm is missing, it defaults to sha256.

Some take no arguments, and some take multiple arguments.

Hash Optional Scheme Specifiers

These scheme specifiers are followed by a dash and a valid hash algo?

rithm, For example: oaep-sha256.

? oaep

? ecdh

? rsassa

? rsapss

? ecdsa

? ecschnorr

Multiple Option Scheme Specifiers

This scheme specifier is followed by a count (max size UINT16) then

followed by a dash(-) and a valid hash algorithm. * ecdaa For example,

ecdaa4-sha256. If no count is specified, it defaults to 4.

No Option Scheme Specifiers

This scheme specifier takes NO arguments. * rsaes

Symmetric Details Specifiers

This field is optional, and defaults based on the type of object being

created and it?s attributes. Generally, any valid Symmetric specifier

from the Type Specifiers list should work. If not specified, an asym?

metric objects symmetric details defaults to aes128cfb.

Examples

Create an rsa2048 key with an rsaes asymmetric encryption scheme

tpm2_create -C parent.ctx -G rsa2048:rsaes -u key.pub -r key.priv

Create an ecc256 key with an ecdaa signing scheme with a count of 4 and

sha384 hash

/tpm2_create -C parent.ctx -G ecc256:ecdaa4-sha384 -u key.pub -r

key.priv cryptographic algorithms ALGORITHM.

Object Attributes

Object Attributes are used to control various properties of created ob?

jects. When specified as an option, either the raw bitfield mask or

?nice-names? may be used. The values can be found in Table 31 Part 2

of the TPM2.0 specification, which can be found here:

<https://trustedcomputinggroup.org/wp-content/uploads/TPM-Rev-2.0-Part-2-Structures-01.38.pdf>

Nice names are calculated by taking the name field of table 31 and re? moving the prefix TPMA_OBJECT_ and lowercasing the result. Thus, TP? MA_OBJECT_FIXEDTPM becomes fixedtpm. Nice names can be joined using the bitwise or ?|? symbol.

For instance, to set The fields TPMA_OBJECT_FIXEDTPM, TPMA_OBJECT_NODA, and TPMA_OBJECT_SIGN_ENCRYPT, the argument would be: fixedtpm|noda|sign specifying the object attributes ATTRIBUTES.

COMMON OPTIONS

This collection of options are common to many programs and provide in? formation that many users may expect.

? -h, --help=[man|no-man]: Display the tools manpage. By default, it attempts to invoke the manpager for the tool, however, on failure will output a short tool summary. This is the same behavior if the ?man? option argument is specified, however if explicit ?man? is re? quested, the tool will provide errors from man on stderr. If the ?no-man? option if specified, or the manpager fails, the short op? tions will be output to stdout.

To successfully use the manpages feature requires the manpages to be installed or on MANPATH, See man(1) for more details.

? -v, --version: Display version information for this tool, supported tctis and exit.

? -V, --verbose: Increase the information that the tool prints to the console during its execution. When using this option the file and line number are printed.

? -Q, --quiet: Silence normal tool output to stdout.

? -Z, --enable-errata: Enable the application of errata fixups. Useful if an errata fixup needs to be applied to commands sent to the TPM.

Defining the environment TPM2TOOLS_ENABLE_ERRATA is equivalent. in? formation many users may expect.

The TCTI or ?Transmission Interface? is the communication mechanism with the TPM. TCTIs can be changed for communication with TPMs across different mediums.

To control the TCTI, the tools respect:

1. The command line option -T or --tcti

2. The environment variable: TPM2TOOLS_TCTI.

Note: The command line option always overrides the environment vari? able.

The current known TCTIs are:

? tabrmd - The resource manager, called tabrmd (https://github.com/tpm2-software/tpm2-abrmd). Note that tabrmd and abrmd as a tcti name are synonymous.

? mssim - Typically used for communicating to the TPM software simula? tor.

? device - Used when talking directly to a TPM device file.

? none - Do not initalize a connection with the TPM. Some tools allow for off-tpm options and thus support not using a TCTI. Tools that do not support it will error when attempted to be used without a TCTI connection. Does not support ANY options and MUST BE presented as the exact text of ?none?.

The arguments to either the command line option or the environment variable are in the form:

<tcti-name>:<tcti-option-config>

Specifying an empty string for either the <tcti-name> or <tcti-op? tion-config> results in the default being used for that portion respec? tively.

TCTI Defaults

When a TCTI is not specified, the default TCTI is searched for using dlopen(3) semantics. The tools will search for tabrmd, device and mssim TCTIs IN THAT ORDER and USE THE FIRST ONE FOUND. You can query what TCTI will be chosen as the default by using the -v option to print the version information. The ?default-tcti? key-value pair will indi? cate which of the aforementioned TCTIs is the default.

Custom TCTIs

Any TCTI that implements the dynamic TCTI interface can be loaded. The tools internally use dlopen(3), and the raw tcti-name value is used for the lookup. Thus, this could be a path to the shared library, or a li? brary name as understood by dlopen(3) semantics.

TCTI OPTIONS

This collection of options are used to configure the various known TCTI modules available:

? device: For the device TCTI, the TPM character device file for use by the device TCTI can be specified. The default is /dev/tpm0.

Example: -T device:/dev/tpm0 or export TPM2TOOLS_TCTI=?de? vice:/dev/tpm0?

? mssim: For the mssim TCTI, the domain name or IP address and port number used by the simulator can be specified. The default are 127.0.0.1 and 2321.

Example: -T mssim:host=localhost,port=2321 or export TPM2TOOLS_TC? TI=?mssim:host=localhost,port=2321?

? abrmd: For the abrmd TCTI, the configuration string format is a se? ries of simple key value pairs separated by a `,' character. Each key and value string are separated by a `=' character.

? TCTI abrmd supports two keys:

1. `bus_name' : The name of the tabrmd service on the bus (a string).
2. `bus_type' : The type of the dbus instance (a string) limited to `session' and `system'.

Specify the tabrmd tcti name and a config string of bus_name=com.ex? ample.FooBar:

\--tcti=tabrmd:bus_name=com.example.FooBar

Specify the default (abrmd) tcti and a config string of bus_type=ses? sion:

\--tcti:bus_type=session

NOTE: abrmd and tabrmd are synonymous. the various known TCTI mod? ules.

EXAMPLES

Create an ECC primary object

tpm2_createprimary -C o -g sha256 -G ecc -c context.out

Create a primary object that follows the guidance of TCG Provisioning guide

See : https://trustedcomputinggroup.org/wp-content/up?

loads/TCG-TPM-v2.0-Provisioning-Guidance-Published-v1r1.pdf

Where unique.dat contains the binary-formatted data: 0x00 0x01 (0x00 *

256)

```bash tpm2_createprimary -C o -G rsa2048:aes128cfb -g sha256 -c

prim.ctx

-a `restricted|decrypt|fixedtpm|fixedparent|sensitivedataorigin|user?

withauth|

noda' -u unique.dat

Create a primary object and output the public key in pem format

tpm2_createprimary -c primary.ctx --format=pem --output=public.pem

Returns

Tools can return any of the following codes:

? 0 - Success.

? 1 - General non-specific error.

? 2 - Options handling error.

? 3 - Authentication error.

? 4 - TCTI related error.

? 5 - Non supported scheme. Applicable to tpm2_testparams.

BUGS

Github Issues (https://github.com/tpm2-software/tpm2-tools/issues)

HELP

See the Mailing List (https://lists.01.org/mailman/listinfo/tpm2)
```