
*Full credit is given to the above companies including the OS that this PDF file was generated!*

## *Red Hat Enterprise Linux Release 9.2 Manual Pages on 'tpm2_gettime.1' command*

***$ man tpm2_gettime.1***

tpm2_gettime(1) General Commands Manual tpm2_gettime(1)

NAME

tpm2_gettime(1) - Get the current time and clock from the TPM in a signed form.

SYNOPSIS

tpm2_gettime [OPTIONS] [ARGUMENT]

DESCRIPTION

tpm2_gettime(1) - Provides a signed copy of the current time and clock from the TPM. It returns both a signature, and the data in the stan? dard TPM attestation form, a TPMS_ATTEST structure.

It outputs to stdout, in YAML format, the TPMS_TIME_INFO structure from the TPM. The structure contains the current setting of Time, Clock, resetCount, and restartCount. The structure is output as YAML defined as:

```
time: 13673142     # 64 bit value of time since last _TPM_Init or TPM2_Startup
                   # in ms.
clock_info:
  clock: 13673142  # 64 bit value of time TPM has been powered on in ms.
  reset_count: 0   # 32 bit value of the number of TPM Resets since the last
                   # TPM2_Clear.
  restart_count: 0 # 32 bit value of the number of times that TPM2_Shutdown or
                   # _TPM_Hash_Start have occurred since the last TPM Reset or
                   # TPM2_Clear.
```

```
    safe: yes       # boolean yes|no value that no value of Clock greater than
                    # the current value of Clock has been previously reported by
                    # the TPM.
```

OPTIONS

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

Context object pointing to the the key used for signing. Either a file or a handle number. See section ?Context Object Format?.

? -p, --auth_AUTH_:

Optional authorization value to use the key specified by -c. Autho? rization values should follow the ?authorization formatting stan? dards?, see section ?Authorization Formatting?.

? -P, --endorse-auth_AUTH_:

Optional authorization value for the endorsement hierarchy. Autho? rization values should follow the ?authorization formatting stan? dards?, see section ?Authorization Formatting?.

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

The hash algorithm used to digest the message. Algorithms should follow the ?formatting standards?, see section ?Algorithm Speci? fiers?. Also, see section ?Supported Hash Algorithms? for a list of supported hash algorithms.

? -s, --scheme=ALGORITHM:

The signing scheme used to sign the message. Optional. Signing schemes should follow the ?formatting standards?, see section ?Algo? rithm Specifiers?. Also, see section ?Supported Signing Schemes? for a list of supported signature schemes. If specified, the signature scheme must match the key type. If left unspecified, a default sig? nature scheme for the key type will be used.

? -q, --qualification=FILE_OR_HEX_STR:

Optional, the policy qualifier data that the signer can choose to in? clude in the signature. Can be either a hex string or path.

? -o, --signature=FILE:

The signature file, records the signature structure.

? -f, --format=FORMAT:

Format selection for the signature output file. See section ?Signa?
ture Format Specifiers?.

? ?attestation=FILE:

The attestation data of the type TPMS_ATTEST signed with signing key.

? --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.

? ARGUMENT the command line argument specifies the file data for sign.

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.

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.

TCTI Configuration

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. # Signature Format Specifiers

Format selection for the signature output file. tss (the default) will

output a binary blob according to the TPM 2.0 specification and any po?

tential compiler padding. The option plain will output the plain sig?

nature data as defined by the used cryptographic algorithm.

EXAMPLES

Create a key and get attested TPM time

```
tpm2_createprimary -C e -c primary.ctx
tpm2_create -G rsa -u rsa.pub -r rsa.priv -C primary.ctx
tpm2_load -C primary.ctx -u rsa.pub -r rsa.priv -c rsa.ctx
tpm2_gettime -c rsa.ctx -o attest.sig --attestation attest.data
```

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)