



*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\_commit.1' command**

**\$ man tpm2\_commit.1**

tpm2\_commit(1)      General Commands Manual      tpm2\_commit(1)

### NAME

tpm2\_commit(1) - Performs the first part of an ECC anonymous signing operation.

### SYNOPSIS

tpm2\_commit [OPTIONS]

### DESCRIPTION

tpm2\_commit(1) - Performs the first part of an ECC anonymous signing operation. The TPM will perform the point multiplications on the provided points and return intermediate signing values. The signing key is an ECC key. The key cannot be a sign+decrypt key and must have an anonymous signing scheme. TPM\_ALG\_ECDSA is the only supported anonymous scheme.

### OPTIONS

? ARGUMENT=FILE:

Specify the input data used to derive the x coordinate of the base point.

? --basepoint-y=FILE:

Specify the y coordinate of the basepoint.

? --eccpoint-P=FILE:

Specify a point on the curve used by sign handle.

? --eccpoint-K=FILE:

Output ECC point K ? [ds](x2, y2).

? --eccpoint-L=FILE:

Output ECC point L ? [r](x2, y2).

? -u, --public=FILE:

Output ECC point E ? [r]P1.

? -t, --counter=FILE

Specify file path to save the least-significant 16 bits of commit count.

? -p, --auth=AUTH:

The authorization value for the created object.

? -c, --context=FILE:

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

## References

### 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 algorithms 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 remaining 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 its attributes. Generally, any valid Symmetric specifier from the Type Specifiers list should work. If not specified, an asymmetric 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 information 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 requested, the tool will provide errors from man on stderr. If the ?no-man? option is specified, or the manpager fails, the short options 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. Information 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 variable.

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 simulator.

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

? none - Do not initialize 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-option-config> results in the default being used for that portion respectively.

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 indicate 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 library 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=?device:/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_TCTI=?mssim:host=localhost,port=2321?`

? `abrmd`: For the `abrmd` TCTI, the configuration string format is a series 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.example.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

```
tpm2_createprimary -C o -c prim.ctx -Q
```

```
tpm2_create -C prim.ctx -c key.ctx -u key.pub -r key.priv -G ecc256:ecdaa
```

```
tpm2_commit -c key.ctx -t count.er \
```

```
--eccpoint-K K.bin --eccpoint-L L.bin -u E.bin
```

## 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>)

tpm2-tools

tpm2\_commit(1)