wolfTPM User Manual

June, 2018
Version 1.3

Table of Contents

TPM2_ReadPublic
TPM2_ActivateCredential
TPM2_MakeCredential
TPM2_ObjectChangeAuth
TPM2_Duplicate
TPM2_Rewrap
TPM2_Import
TPM2_RSA_Encrypt
TPM2_RSA_Decrypt
TPM2_ECDH_KeyGen
TPM2_ECDH_ZGen
TPM2_ECC_Parameters
TPM2_ZGen_2Phase
TPM2_EncryptDecrypt
TPM2_EncryptDecrypt2
TPM2_Hash
TPM2_HMAC
TPM2_HMAC_Start
TPM2_HashSequenceStart
TPM2_SequenceUpdate
TPM2_SequenceComplete
TPM2_EventSequenceComplete
TPM2_Certify
TPM2_CertifyCreation
TPM2_Quote
TPM2_GetSessionAuditDigest
TPM2_GetCommandAuditDigest
TPM2_GetTime
TPM2_Commit
TPM2_EC_Ephemeral
TPM2_VerifySignature
TPM2_Sign
TPM2_SetCommandCodeAuditStatus
TPM2_PCR_Event
TPM2_PCR_Allocate
TPM2_PCR_SetAuthPolicy
TPM2_PCR_SetAuthValue
TPM2_PCR_Reset
TPM2_PolicySigned

TPM2_PolicySecret
TPM2_PolicyTicket
TPM2_PolicyOR
TPM2_PolicyPCR
TPM2_PolicyLocality
TPM2_PolicyNV
TPM2_PolicyCounterTimer
TPM2_PolicyCommandCode
TPM2_PolicyPhysicalPresence
TPM2_PolicyCpHash
TPM2_PolicyNameHash
TPM2_PolicyDuplicationSelect
TPM2_PolicyAuthorize
TPM2_PolicyAuthorizeNV
TPM2_PolicyAuthValue
TPM2_PolicyPassword
TPM2_PolicyGetDigest
TPM2_PolicyNvWritten
TPM2_PolicyTemplate
TPM2_PolicyAuthorizeNV
_TPM_Hash_Start
_TPM_Hash_Data
_TPM_Hash_End
TPM2_HierarchyControl
TPM2_SetPrimaryPolicy
TPM2_ChangePPS
TPM2_ChangeEPS
TPM2_Clear
TPM2_ClearControl
TPM2_HierarchyChangeAuth
TPM2_DictionaryAttackLockReset
TPM2_DictionaryAttackParameters
TPM2_PP_Commands
TPM2_SetAlgorithmSet
TPM2_FieldUpgradeStart
TPM2_FieldUpgradeData
TPM2_FirmwareRead
TPM2_ContextSave
TPM2_ContextLoad

TPM2_EvictControl
TPM2_ReadClock
TPM2_ClockSet
TPM2_ClockRateAdjust
TPM2_TestParms
TPM2_NV_DefineSpace
TPM2_NV_UndefineSpace
TPM2_NV_UndefineSpaceSpecial
TPM2_NV_ReadPublic
TPM2_NV_Write
TPM2_NV_Increment
TPM2_NV_Extend
TPM2_NV_SetBits
TPM2_NV_WriteLock
TPM2_NV_GlobalWriteLock
TPM2_NV_Read
TPM2_NV_ReadLock
TPM2_NV_ChangeAuth
TPM2_NV_Certify
TPM2_Cleanup
TPM2_SetSessionAuth
TPM2_GetActiveCtx
TPM2_GetHashDigestSize
TPM2_GetNonce
TPM2_SetupPCRSel
TPM2_GetRCString
TPM2_GetAlgName
TPM2_GetCurveSize
TPM2_PrintBin
wolfTPM2_GetTpmDevId
wolfTPM2_SetAuth
wolfTPM2_StartSession

1. Intro

---

wolfTPM is a portable TPM 2.0 project, designed for embedded use. It is highly portable, due to having been written in native C, having a single IO callback for SPI hardware interface, no external dependencies, and its compacted code with low resource usage.

1.1 Protocol overview

Trusted Platform Module (TPM, also known as ISO/IEC 11889) is an international standard for a secure cryptoprocessor, a dedicated microcontroller designed to secure hardware through integrated cryptographic keys. Computer programs can use a TPM to authenticate hardware devices, since each TPM chip has a unique and secret RSA key burned in as it is produced.

According to Wikipedia, a TPM provides the following[1]:

- A random number generator
- Facilities for the secure generation of cryptographic keys for limited uses.
- Remote attestation: Creates a nearly unforgeable hash key summary of the hardware and software configuration. The software in charge of hashing the configuration data determines the extent of the summary. This allows a third party to verify that the software has not been changed.
- Binding: Encrypts data using the TPM bind key, a unique RSA key descended from a storage key.
- Sealing: Similar to binding, but in addition, specifies the TPM state for the data to be decrypted (unsealed).

In addition, TPM can also be used for various applications such as platform integrity, disk encryption, password protection, and software license protection.

1.2 Hierarchies

Platform: `TPM_RH_PLATFORM`
Owner: `PM_RH_OWNER`
Endorsement: `TPM_RH_ENDORSEMENT`

Each hierarchy has their own manufacture generated seed.
The arguments used on TPM2_Create or TPM2_CreatePrimary create a template, which is fed into a KDF to produce the same key based hierarchy used. The key generated is the same each

[1] Wikipedia contributors. (2018, May 30). Trusted Platform Module. In *Wikipedia, The Free Encyclopedia*. Retrieved 22:46, June 20, 2018, from https://en.wikipedia.org/w/index.php?title=Trusted_Platform_Module&oldid=843601428

time; even after reboot. The generation of a new RSA 2048 bit key takes about 15 seconds. Typically these are created and then stored in NV using TPM2_EvictControl. Each TPM generates their own keys uniquely based on the seed.
There is also an Ephemeral hierarchy (TPM_RH_NULL), which can be used to create ephemeral keys.

1.3 Platform Configuration Registers (PCRs)

Platform Configuration Registers (PCRs) are one of the essential features of a TPM. Their prime use case is to provide a method to cryptographically record (measure) software state: both the software running on a platform and configuration data used by that software.[2]

wolfTPM contains hash digests for SHA-1 and SHA-256 with an index 0-23. These hash digests can be extended to prove the integrity of a boot sequence (secure boot).

2. Building wolfTPM

To build the wolfTPM library, it's required to first build and install the wolfSSL library. This can be downloaded from the download page, or through a "git clone" command, shown below:
```
$ git clone https://github.com/wolfssl/wolfssl
```

Once the wolfSSL library has been downloaded, it needs to be built with the following options being passed to the configure script:
```
$ ./configure --enable-certgen --enable-certreq --enable-certext
--enable-pkcs7 --enable-cryptodev
```
Then the wolfSSL library just needs to be built and installed however the user prefers.

The next step is to download and install the wolfTPM library. At the time this documentation was written, the wolfTPM library does not have a stable release yet and needs to be cloned from GitHub. The following commands show how to clone and install wolfTPM:
```
$ git clone https://github.com/wolfssl/wolftpm
$ cd wolftpm
$ ./autogen.sh
$ ./configure
$ make
```

[2] Arthur W., Challener D., Goldman K. (2015) Platform Configuration Registers. In: A Practical Guide to TPM 2.0. Apress, Berkeley, CA

3. Getting Started

The wolfTPM library has TPM 2.0 wrapper tests, native tests, and a sample benchmark application that come ready-to-use after a successful installation of wolfTPM. Below are some instructions on how to run the sample applications yourself.

To interface with the hardware platform that is running these applications, please see the function "`TPM2_IoCb`" inside of `tpm_demo.c`.

3.1 Examples

The two example applications are focused on testing the wrapper APIs and native interfaces for the Raspberry Pi 3, or the STM32 with the CubeMX HAL.

By default, when running the examples wolfTPM will use the native `spi_dev` interface on the Raspberry Pi, and defaults to `/dev/spidev0.1`. If these examples are being tested with the Infineon patches on the Infineon OPTIGA SLB9670, then they will override the kernel interface with their `spi_tis_dev`, causing an error.

If desired, the wolfTPM library can be built with debug enabled for a more detailed example application output. To do this, simply run the "configure" script like this:
`$ ./configure --enable-debug`

If everything has been set up correctly, the example applications should have the following output:

```
$ ./examples/wrap/wrap_test
TPM2 Demo for Wrapper APIs
RSA Encrypt Test Passed
ECC Sign/Verify Test Passed
ECC DH Generation Passed
NV Test on index 0x1800200 with 1024 bytes passed

$ ./examples/native/native_test
TPM2 Demo using Native API's
TPM2: Caps 0x30000697, Did 0x001b, Vid 0x15d1, Rid 0x10
TPM2_Startup pass
TPM2_SelfTest pass
TPM2_GetTestResult: Size 10, Rc 0x0
TPM2_IncrementalSelfTest: Rc 0x0, Alg 0x1 (Todo 0)
```

```
TPM2_GetCapability: Property FamilyIndicator 0x322e3000
TPM2_GetCapability: Property PCR Count 24
TPM2_GetRandom: Got 32 bytes
TPM2_StirRandom: success
TPM2_PCR_Read: Index 0, Digest Sz 32, Update Counter 21
TPM2_PCR_Read: Index 1, Digest Sz 32, Update Counter 21
TPM2_PCR_Read: Index 2, Digest Sz 32, Update Counter 21
TPM2_PCR_Read: Index 3, Digest Sz 32, Update Counter 21
TPM2_PCR_Read: Index 4, Digest Sz 32, Update Counter 21
TPM2_PCR_Read: Index 5, Digest Sz 32, Update Counter 21
TPM2_PCR_Read: Index 6, Digest Sz 32, Update Counter 21
TPM2_PCR_Read: Index 7, Digest Sz 32, Update Counter 21
TPM2_PCR_Read: Index 8, Digest Sz 32, Update Counter 21
TPM2_PCR_Read: Index 9, Digest Sz 32, Update Counter 21
TPM2_PCR_Read: Index 10, Digest Sz 32, Update Counter 21
TPM2_PCR_Read: Index 11, Digest Sz 32, Update Counter 21
TPM2_PCR_Read: Index 12, Digest Sz 32, Update Counter 21
TPM2_PCR_Read: Index 13, Digest Sz 32, Update Counter 21
TPM2_PCR_Read: Index 14, Digest Sz 32, Update Counter 21
TPM2_PCR_Read: Index 15, Digest Sz 32, Update Counter 21
TPM2_PCR_Read: Index 16, Digest Sz 32, Update Counter 21
TPM2_PCR_Read: Index 17, Digest Sz 32, Update Counter 21
TPM2_PCR_Read: Index 18, Digest Sz 32, Update Counter 21
TPM2_PCR_Read: Index 19, Digest Sz 32, Update Counter 21
TPM2_PCR_Read: Index 20, Digest Sz 32, Update Counter 21
TPM2_PCR_Read: Index 21, Digest Sz 32, Update Counter 21
TPM2_PCR_Read: Index 22, Digest Sz 32, Update Counter 21
TPM2_PCR_Read: Index 23, Digest Sz 32, Update Counter 21
TPM2_PCR_Extend success
TPM2_PCR_Read: Index 0, Digest Sz 32, Update Counter 22
TPM2_StartAuthSession: sessionHandle 0x3000000
TPM2_PolicyGetDigest: size 32
TPM2_PCR_Read: Index 0, Digest Sz 20, Update Counter 22
wc_Hash of PCR[0]: size 32
TPM2_PolicyPCR failed 0x1c4: TPM_RC_AUTHSIZE
TPM2_PolicyPCR: Updated
TPM2_PolicyRestart: Done
TPM2_HashSequenceStart: sequenceHandle 0x80000000
Hash SHA256 test success
TPM2_CreatePrimary: Endorsement 0x80000000 (314 bytes)
TPM2_CreatePrimary: Storage 0x80000001 (282 bytes)
TPM2_LoadExternal: 0x80000002
TPM2_MakeCredential: credentialBlob 68, secret 256
```

```
TPM2_ReadPublic Handle 0x80000002: pub 314, name 34, qualifiedName 34
Create HMAC-SHA256 Key success, public 48, Private 141
TPM2_Load New HMAC Key Handle 0x80000002
TPM2_PolicyCommandCode: success
TPM2_ObjectChangeAuth failed 0x9a2: TPM_RC_BAD_AUTH
TPM2_ObjectChangeAuth: private 2
TPM2_ECC_Parameters: CurveID 3, sz 256, p 32, a 32, b 32, gX 32, gY
32, n 32, h 1
TPM2_Create: New ECDSA Key: pub 88, priv 126
TPM2_Load ECDSA Key Handle 0x80000002
TPM2_Sign: ECC S 32, R 32
TPM2_VerifySignature: Tag 32802
TPM2_Create: New ECDH Key: pub 88, priv 126
TPM2_Load ECDH Key Handle 0x80000002
TPM2_ECDH_KeyGen: zPt 68, pubPt 68
TPM2_Create: New RSA Key: pub 278, priv 222
TPM2_Load RSA Key Handle 0x80000002
TPM2_RSA_Encrypt: 256
TPM2_RSA_Decrypt: 32
RSA Encrypt/Decrypt test passed
TPM2_NV_ReadPublic failed 0x18b: TPM_RC_HANDLE
TPM2_NV_ReadPublic failed 0x18b: TPM_RC_HANDLE
TPM2_NV_ReadPublic failed 0x18b: TPM_RC_HANDLE
TPM2_NV_ReadPublic failed 0x18b: TPM_RC_HANDLE
TPM2_NV_ReadPublic failed 0x18b: TPM_RC_HANDLE
TPM2_NV_ReadPublic failed 0x18b: TPM_RC_HANDLE
TPM2_NV_ReadPublic failed 0x18b: TPM_RC_HANDLE
TPM2_NV_ReadPublic failed 0x18b: TPM_RC_HANDLE
TPM2_NV_ReadPublic failed 0x18b: TPM_RC_HANDLE
TPM2_NV_ReadPublic failed 0x18b: TPM_RC_HANDLE
TPM2_NV_DefineSpace failed 0x184: TPM_RC_VALUE
```

3.2 Benchmarks

The wolfTPM benchmark application requires the same setup as the example applications. Below is a sample output from running the benchmark application on an Infineon OPTIGA SLB9670:

```
$ ./examples/bench/bench
TPM2 Benchmark using Wrapper API's
RSA    2048 Public        65 ops took 1.005 sec, avg 15.466 ms, 64.657
ops/sec
RSA     2048 Private       3 ops took 1.343 sec, avg 447.759 ms,
```

```
2.233 ops/sec
RSA     2048 Pub  OAEP     12 ops took 1.040 sec, avg 86.657 ms,
11.540 ops/sec
RSA     2048 Priv OAEP      2 ops took 1.032 sec, avg 515.885 ms,
1.938 ops/sec
ECDSA    256 sign          14 ops took 1.037 sec, avg 74.101 ms,
13.495 ops/sec
ECDSA    256 verify         8 ops took 1.027 sec, avg 128.417 ms,
7.787 ops/sec
ECDHE    256 agree          8 ops took 1.040 sec, avg 130.003 ms,
7.692 ops/sec
```

4. wolfTPM Library Design

4.1 Library Headers

wolfTPM header files are located in the following locations:

wolfTPM:	wolftpm/
wolfSSL:	wolfssl/
wolfCrypt:	wolfssl/wolfcrypt

The general header file that should be included from wolfTPM is shown below:

```
#include <wolftpm/tpm2.h>
```

4.2 Example Design

Every example application that is included with wolfTPM includes the tpm_io.h header file, located in wolfTPM/examples. The tpm_io.c file sets up the example HAL IO callback necessary for testing and running the example applications with a Linux Kernel, STM32 CubeMX HAL or Atmel/Microchip ASF. The reference is easily modified, such that custom IO callbacks or different callbacks may be added or removed as desired.

5. wolfTPM API Reference

TPM2_Init

Synopsis:

#include <wolftpm/tpm2.h>

TPM_RC TPM2_Init(TPM2_CTX* ctx, TPM2HalIoCb ioCb, void* userCtx);

Description:

Initializes a TPM

Return Values:

If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:

ctx - a pointer to a TPM2_CTX struct
ioCb - a TPM2HalIoCb (HAL IO) callback struct
userCtx - a pointer to the user's context that will be stored as a member of the **ctx** struct

Example:

```
int rc;
TPM2_CTX tpm2Ctx;

rc = TPM2_Init(&tpm2Ctx, TPM2_IoCb, userCtx);
if (rc != TPM_RC_SUCCESS) {
      /* TPM2_Init failed */
}
```

See Also:

TPM2_Startup
TPM2_GetRCString
TPM2_FlushContext

TPM2_Startup

Synopsis:

#include <wolftpm/tpm2.h>

TPM_RC TPM2_Startup(Startup_In* in);

Description:

TPM2_Startup is always preceded by TPM_Init, which is the physical indication that TPM initialization is necessary because of a system-wide reset. TPM2_Startup is only valid after TPM_Init. Additional TPM2_Startup commands are not allowed after it has completed successfully. If a TPM requires TPM2_Startup and another command is received, or if the TPM receives TPM2_Startup when it is not required, the TPM shall return TPM_RC_INITIALIZE.

Return Values:

If successful the call will return **TPM_RC_SUCCESS**.
BAD_FUNC_ARG is an error that may be returned.
TPM_RC_FAILURE is an error that may be returned.
TPM_RC_INITIALIZE may be returned if TPM2_Startup is called unnecessarily.
TPM_RC_VALUE may be returned if the TPM receives a startup without a prior shutdown.

Parameters:

in - a pointer to a Shutdown_In struct.

Example

```
XMEMSET(&cmdIn.startup, 0, sizeof(cmdIn.startup));
cmdIn.startup.startupType = TPM_SU_CLEAR;
rc = TPM2_Startup(&cmdIn.startup);
if (rc != TPM_RC_SUCCESS && rc != TPM_RC_INITIALIZE) {
    /* failed to startup TPM2 */
}
```

See Also:

TPM2_Cleanup
TPM2_FlushContext
TPM2_Shutdown
TPM2_GetRCString

TPM2_Shutdown

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_Shutdown(Shutdown_In* in);

Description:
This command is used to prepare the TPM for a power cycle.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
BAD_FUNC_ARG is an error that may be returned.
TPM_RC_FAILURE is an error that may be returned.

Parameters:
in - a pointer to a Startup_In struct containing a shutdownType (TPM_SU) member. Indicates how the subsequent TPM2_Startup will be processed.

Example:
```
cmdIn.shutdown.shutdownType = TPM_SU_CLEAR;
rc = TPM2_Shutdown(&cmdIn.shutdown);
if (rc != TPM_RC_SUCCESS) {
    /* failed to startup TPM2 */
}
```

See Also:
TPM2_Cleanup
TPM2_FlushContext
TPM2_Startup
TPM2_GetRCString

TPM2_GetCapability

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_GetCapability(GetCapability_In* in, GetCapability_Out* out);

Description:
The function writes the capability type into the out parameter.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
BAD_FUNC_ARG is an error that may be returned.
-1 on unknown capability type.

Parameters:
in - a pointer to a GetCapability_In struct.
out - a pointer to a GetCapability_Out struct.

Example:
```
XMEMSET(&cmdIn.cap, 0, sizeof(cmdIn.cap));
cmdIn.cap.capability = TPM_CAP_TPM_PROPERTIES;
cmdIn.cap.property = TPM_PT_FAMILY_INDICATOR;
cmdIn.cap.propertyCount = 1;
rc = TPM2_GetCapability(&cmdIn.cap, &cmdOut.cap);
if (rc != TPM_RC_SUCCESS) {
    /* Could not get capability */
}
```

See Also:
TPM2_GetRCString

TPM2_SelfTest

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_SelfTest(SelfTest_In* in);

Description:
This command causes the TPM to perform a test of its capabilities.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.

TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:

in - a pointer to a SelfTest_In struct. If the fullTest member of **in** is set to YES, all functions will be tested. If NO, the TPM will only test those functions that have not been previously tested.

Example:

```
XMEMSET(&cmdIn.selfTest, 0, sizeof(cmdIn.selfTest));
cmdIn.selfTest.fullTest = YES;
rc = TPM2_SelfTest(&cmdIn.selfTest);
if (rc != TPM_RC_SUCCESS) {
    /* self test failed */
}
```

See Also:

TPM2_GetTestResult
TPM2_GetRCString
TPM2_IncrementalSelfTest

TPM2_IncrementalSelfTest

Synopsis:

#include <wolftpm/tpm2.h>

TPM_RC TPM2_IncrementalSelfTest(IncrementalSelfTest_In* in,
IncrementalSelfTest_Out* out);

Description:

This command causes the TPM to perform a test of the selected algorithms.

Return Values:

If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:

in - a pointer to an IncrementalSelfTest_In struct
out - a pointer to an IncrementalSelfTest_Out struct

Example:

```
XMEMSET(&cmdIn.incSelfTest, 0, sizeof(cmdIn.incSelfTest));
cmdIn.incSelfTest.toTest.count = 1;
cmdIn.incSelfTest.toTest.algorithms[0] = TPM_ALG_RSA;
rc = TPM2_IncrementalSelfTest(&cmdIn.incSelfTest, &cmdOut.incSelfTest);
printf("TPM2_IncrementalSelfTest: Rc 0x%x, Alg 0x%x (Todo %d)\n",
        rc, cmdIn.incSelfTest.toTest.algorithms[0],
        (int)cmdOut.incSelfTest.toDoList.count);
```

See Also:

TPM2_GetTestResult
TPM2_GetRCString

TPM2_GetTestResult

Synopsis:

#include <wolftpm/tpm2.h>

TPM_RC TPM2_GetTestResult(GetTestResult_Out* out);

Description:

This command returns manufacturer-specific information regarding the results of a self-test and an indication of the test status.

Return Values:

If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:

out - a pointer to an GetTestResult_Out struct

Example:

```
rc = TPM2_SelfTest(&cmdIn.selfTest);
...
rc = TPM2_GetTestResult(&cmdOut.tr);
```

```
if (rc != TPM_RC_SUCCESS) {
      /* self test failed */
}
```

See Also:
TPM2_GetRCString

TPM2_GetRandom

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_GetRandom(GetRandom_In* in, GetRandom_Out* out);

Description:

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to an GetRandom_In struct
out - a pointer to an GetRandom_Out struct

Example:
```
XMEMSET(&cmdIn.getRand, 0, sizeof(cmdIn.getRand));
cmdIn.getRand.bytesRequested = WC_SHA256_DIGEST_SIZE;
rc = TPM2_GetRandom(&cmdIn.getRand, &cmdOut.getRand);
if (rc != TPM_RC_SUCCESS) {
      /* TPM2_GetRandom failed */
}
```

See Also:
TPM2_GetRCString

TPM2_StirRandom

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_StirRandom(StirRandom_In* in);

Description:
This command adds "additional information" to the RNG state

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a StirRandom_In struct.

Example:
```
XMEMSET(&cmdIn.getRand, 0, sizeof(cmdIn.getRand));
cmdIn.getRand.bytesRequested = WC_SHA256_DIGEST_SIZE;
rc = TPM2_GetRandom(&cmdIn.getRand, &cmdOut.getRand);
if (rc != TPM_RC_SUCCESS) {
      /* TPM2_GetRandom failed */
}
```

See Also:
TPM2_GetRCString

TPM2_PCR_Read

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_PCR_Read(PCR_Read_In* in, PCR_Read_Out* out);

Description:
This command writes the values of all PCR specified in the *pcrSelectionIn* member of the in parameter to the *pcrSelectionOut* member of the out parameter.

The TPM will process the list of TPMS_PCR_SELECTION in *pcrSelectionIn* in order. Within each TPMS_PCR_SELECTION, the TPM will process the bits in the *pcrSelect* array in ascending PCR order (see ISO/IEC 11889-2 for definition of the PCR order). If a bit is set, and the indicated PCR is present, then the TPM will add the digest of the PCR to the list of values to be written to *pcrValues*.

Return Values:

If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:

in - a pointer to an PCR_Read_In struct, which contains a *pcrSelectionIn* (TPML_PCR_SELECTION) member.
out - a pointer to an PCR_Read_Out struct, which contains *pcrValues* (TPML_DIGEST) and *pcrSelectionOut* (TPML_PCR_SELECTION) members.

Example:

```
for (i=0; i<pcrCount; i++) {
    pcrIndex = i;
    XMEMSET(&cmdIn.pcrRead, 0, sizeof(cmdIn.pcrRead));
    TPM2_SetupPCRSel(&cmdIn.pcrRead.pcrSelectionIn,
        TPM_ALG_SHA256, pcrIndex);
    rc = TPM2_PCR_Read(&cmdIn.pcrRead, &cmdOut.pcrRead);
    if (rc != TPM_RC_SUCCESS) {
        /* Could not read PCR */
    }
}
```

See Also:

TPM2_GetRCString
TPM2_SetupPCRSel

TPM2_PCR_Extend

Synopsis:

#include <wolftpm/tpm2.h>

TPM_RC TPM2_PCR_Extend(PCR_Extend_In* in);

Description:

This command is used to cause an update to a PCR. The *digests* member of **in** contains one or more tagged digest values identified by an algorithm ID. For each digest, the PCR associated with *pcrHandle* is extended into the bank identified by the tag.

Return Values:

If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:

in - a pointer to an PCR_Extend_In struct that contains digests (TPML_DIGEST_VALUES) and pcrHandle (TPMI_DH_PCR) members.

Example:

```
pcrIndex = 0;
XMEMSET(&cmdIn.pcrExtend, 0, sizeof(cmdIn.pcrExtend));
cmdIn.pcrExtend.pcrHandle = pcrIndex;
cmdIn.pcrExtend.digests.count = 1;
cmdIn.pcrExtend.digests.digests[0].hashAlg = TPM_ALG_SHA256;
for (i=0; i<WC_SHA256_DIGEST_SIZE; i++) {
    cmdIn.pcrExtend.digests.digests[0].digest.H[i] = i;
}
rc = TPM2_PCR_Extend(&cmdIn.pcrExtend);
if (rc != TPM_RC_SUCCESS) {
    /* could not extend pcr */
}
```

See Also:

TPM2_GetRCString
TPM2_SetupPCRSel

TPM2_Create

Synopsis:

#include <wolftpm/tpm2.h>

TPM_RC TPM2_Create(Create_In* in, Create_Out* out);

Description:

This command is used to create an object that can be loaded into a TPM using **TPM2_Load**. If the command completes successfully, the TPM will create the new object and write the object's creation data, its public area, and its encrypted sensitive area to the **out** parameter's *creationData*, *outPublic*, and *outPrivate* members respectively. Preservation of the written data is the responsibility of the caller. The object will need to be loaded (**TPM2_Load**) before it may be used.

Return Values:

If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:

in - a pointer to a Create_In struct, which contains all the information required to set up a new object
out - a pointer to a Create_Out struct which contains the new object's data

Example:

```
/* Create an HMAC-SHA256 Key */
XMEMSET(&cmdIn.create, 0, sizeof(cmdIn.create));
cmdIn.create.parentHandle = storage.handle;
cmdIn.create.inSensitive.sensitive.userAuth.size = sizeof(usageAuth)-1;
XMEMCPY(cmdIn.create.inSensitive.sensitive.userAuth.buffer, usageAuth,
    cmdIn.create.inSensitive.sensitive.userAuth.size);
cmdIn.create.inSensitive.sensitive.data.size = sizeof(userKey)-1;
XMEMCPY(cmdIn.create.inSensitive.sensitive.data.buffer, userKey,
    cmdIn.create.inSensitive.sensitive.data.size);
cmdIn.create.inPublic.publicArea.type = TPM_ALG_KEYEDHASH;
cmdIn.create.inPublic.publicArea.nameAlg = TPM_ALG_SHA256;
cmdIn.create.inPublic.publicArea.objectAttributes = (
    TPMA_OBJECT_userWithAuth | TPMA_OBJECT_sign | TPMA_OBJECT_noDA);
cmdIn.create.inPublic.publicArea.parameters.keyedHashDetail.scheme.scheme =
TPM_ALG_HMAC;
cmdIn.create.inPublic.publicArea.parameters.keyedHashDetail.scheme.details.hmac
.hashAlg = TPM_ALG_SHA
rc = TPM2_Create(&cmdIn.create, &cmdOut.create);
if (rc != TPM_RC_SUCCESS) {
    printf("TPM2_Create HMAC failed 0x%x: %s\n", rc, TPM2_GetRCString(rc));
    goto exit;
}
```

See Also:
TPM2_GetRCString
TPM2_SetupPCRSel

TPM2_CreateLoaded

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_CreateLoaded(CreateLoaded_In* in, CreateLoaded_Out* out);

Description:

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a CreateLoaded_In struct
out - a pointer to a CreateLoaded_Out struct

Example:

See Also:

TPM2_CreatePrimary

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_CreatePrimary(CreatePrimary_In* in, CreatePrimary_Out* out);

Description:

This command is used to create a Primary Object under one of the Primary Seeds or a Temporary Object under TPM_RH_NULL. The command uses a TPM2B_PUBLIC as a template for the object to be created.

Return Values:

If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:

in - a pointer to a CreatePrimary_In struct, which contains the *inPublic* (TPM2B_PUBLIC) template for the object to be created
out - a pointer to a CreatePrimary_Out struct

Example:

```
/* Create Primary (Endorsement) */
XMEMSET(&cmdIn.createPri, 0, sizeof(cmdIn.createPri));
...
rc = TPM2_CreatePrimary(&cmdIn.createPri, &cmdOut.createPri);
if (rc != TPM_RC_SUCCESS) {
    /* could not create primary */
}
```

See Also:

TPM2_GetRCString

TPM2_Load

Synopsis:

#include <wolftpm/tpm2.h>

TPM_RC TPM2_Load(Load_In* in, Load_Out* out);

Description:

This command is used to load objects into the TPM.

Return Values:

If successful the call will return **TPM_RC_SUCCESS**.

TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a Load_In struct
out - a pointer to a Load_Out struct

Example:
```
XMEMSET(&cmdIn.load, 0, sizeof(cmdIn.load));
cmdIn.load.parentHandle = storage.handle;
cmdIn.load.inPrivate = hmacKey.priv;
cmdIn.load.inPublic = hmacKey.pub;
rc = TPM2_Load(&cmdIn.load, &cmdOut.load);
if (rc != TPM_RC_SUCCESS) {
    /* Could not load TPM */
}
```

See Also:
TPM2_GetRcString

TPM2_FlushContext

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_FlushContext(FlushContext_In* in);

Description:
This command causes all context associated with a loaded object or session to be removed from TPM memory. This command may not be used to remove a persistent object from the TPM.

A session does not have to be loaded in TPM memory to have its context flushed. The saved session context associated with the indicated handle is invalidated.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a FlushContext_In struct

Example:
```
XMEMSET(&cmdIn.readPub, 0, sizeof(cmdIn.readPub));
cmdIn.readPub.objectHandle = handle;
rc = TPM2_ReadPublic(&cmdIn.readPub, &cmdOut.readPub);
if (rc != TPM_RC_SUCCESS) {
    /* could not read public key */
}

cmdIn.flushCtx.flushHandle = handle;
TPM2_FlushContext(&cmdIn.flushCtx);
```

See Also:
TPM2_GetRcString
TPM2_ReadPublic

TPM2_Unseal

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_Unseal(Unseal_In* in, Unseal_Out* out);

Description:
This command writes the data in a loaded Sealed Data Object to the **out** parameter.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to an Unseal_In struct
out - a pointer to an Unseal_Out struct

Example:

See Also:
TPM2_GetRcString
TPM2_ReadPublic

TPM2_StartAuthSession

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_StartAuthSession(StartAuthSession_In* in, StartAuthSession_Out* out);

Description:
This command is used to start an authorization session using alternative methods of establishing the session key. The session key is then used to derive values used for authorization and for encrypting parameters.

This command allows injection of a secret into the TPM using either asymmetric or symmetric encryption. The type of key determines how the value in *encryptedSalt* (a member of **in**) is encrypted. The decrypted secret value is used to compute the session key.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a StartAuthSession_In struct which contains the necessary data to start the authorization session
out - a pointer to a StartAuthSession_Out struct containing the new *sessionHandle* (TPMI_SH_AUTH_SESSION) and an initial *nonceTPM* (TPM2B_NONCE) as members

Example:
```
XMEMSET(&cmdIn.authSes, 0, sizeof(cmdIn.authSes));
cmdIn.authSes.tpmKey = TPM_RH_NULL;
cmdIn.authSes.bind = TPM_RH_NULL;
cmdIn.authSes.sessionType = TPM_SE_POLICY;
cmdIn.authSes.symmetric.algorithm = TPM_ALG_NULL;
```

```
cmdIn.authSes.authHash = TPM_ALG_SHA256;
cmdIn.authSes.nonceCaller.size = WC_SHA256_DIGEST_SIZE;
rc = TPM2_GetNonce(cmdIn.authSes.nonceCaller.buffer,
                   cmdIn.authSes.nonceCaller.size);
if (rc < 0) {
      /* failed to generate block */
}
rc = TPM2_StartAuthSession(&cmdIn.authSes, &cmdOut.authSes);
if (rc != TPM_RC_SUCCESS) {
    /* failed to start auth session */
}
```

See Also:
TPM2_GetRcString
TPM2_GetNonce

TPM2_PolicyRestart

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_PolicyRestart(PolicyRestart_In* in);

Description:
This command allows a policy authorization session to be returned to its initial state.

This command is used after the TPM returns TPM_RC_PCR_CHANGED. That response code indicates that a policy will fail because the PCR have changed after **TPM2_PolicyPCR** was executed. Restarting the session allows the authorizations to be replayed because the session restarts with the same *nonceTPM*. If the PCR are valid for the policy, the policy may then succeed.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a PolicyRestart_In struct

Example:

```
XMEMSET(&cmdIn.policyRestart, 0, sizeof(cmdIn.policyRestart));
cmdIn.policyRestart.sessionHandle = sessionHandle;
rc = TPM2_PolicyRestart(&cmdIn.policyRestart);
if (rc != TPM_RC_SUCCESS) {
    /* Policy restart failed */
}
```

See Also:

TPM2_GetRcString

TPM2_LoadExternal

Synopsis:

#include <wolftpm/tpm2.h>

TPM_RC TPM2_LoadExternal(LoadExternal_In* in, LoadExternal_Out* out);

Description:

This command is used to load an object that is not a Protected Object into the TPM. The command allows loading of a public area or both a public and sensitive area.

- note: Typical use for loading a public area is to allow the TPM to validate an asymmetric signature. Typical use for loading both a public and sensitive area is so the TPM can be used as a crypto accelerator.

Return Values:

If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:

in - a pointer to a LoadExternal_In struct
out - a pointer to a LoadExternal_Out struct

Example:

```
XMEMSET(&cmdIn.loadExt, 0, sizeof(cmdIn.loadExt));
cmdIn.loadExt.inPublic = endorse.pub;
cmdIn.loadExt.hierarchy = TPM_RH_NULL;
rc = TPM2_LoadExternal(&cmdIn.loadExt, &cmdOut.loadExt);
```

```
if (rc != TPM_RC_SUCCESS) {
    /* failed to load external */
}
```

See Also:
TPM2_GetRcString

TPM2_ReadPublic

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_ReadPublic(ReadPublic_In* in, ReadPublic_Out* out);

Description:
This command allows access to the public area of a loaded object.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a ReadPublic_In struct
out - a pointer to a ReadPublic_Out struct

Example:
```
XMEMSET(&cmdIn.readPub, 0, sizeof(cmdIn.readPub));
cmdIn.readPub.objectHandle = handle;
rc = TPM2_ReadPublic(&cmdIn.readPub, &cmdOut.readPub);
if (rc != TPM_RC_SUCCESS) {
      /* failed to read public key */
}
```

See Also:
TPM2_GetRcString
TPM2_FlushContext

TPM2_ActivateCredential

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_ActivateCredential(ActivateCredential_In* in, ActivateCredential_Out* out);

Description:
This command enables the association of a credential with an object in a way that ensures that the TPM has validated the parameters of the credentialed object.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a ActivateCredential_In struct
out - a pointer to a ActivateCredential_Out struct

Example:

See Also:
TPM2_GetRcString

TPM2_MakeCredential

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_MakeCredential(MakeCredential_In* in, MakeCredential_Out* out);

Description:
This command allows the TPM to perform the actions required of a CA in creating a TPM2B_ID_OBJECT containing an activation credential.

Return Values:

If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:

in - a pointer to a MakeCredential_In struct
out - a pointer to a MakeCredential_Out struct

Example:

```
XMEMSET(&cmdIn.makeCred, 0, sizeof(cmdIn.makeCred));
cmdIn.makeCred.handle = handle;
cmdIn.makeCred.credential.size = WC_SHA256_DIGEST_SIZE;
XMEMSET(cmdIn.makeCred.credential.buffer, 0x11,
    cmdIn.makeCred.credential.size);
cmdIn.makeCred.objectName = endorse.name;
rc = TPM2_MakeCredential(&cmdIn.makeCred, &cmdOut.makeCred);
if (rc != TPM_RC_SUCCESS) {
      /* failed to make credential */
}
```

See Also:

TPM2_GetRcString

TPM2_ObjectChangeAuth

Synopsis:

#include <wolftpm/tpm2.h>

TPM_RC TPM2_ObjectChangeAuth(ObjectChangeAuth_In* in,
                             ObjectChangeAuth_Out* out);

Description:

This command is used to change the authorization secret for a TPM-resident object.

Return Values:

If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:

in - a pointer to a ObjectChangeAuth_In struct
out - a pointer to a ObjectChangeAuth_Out struct

Example:

```
XMEMSET(&cmdIn.objChgAuth, 0, sizeof(cmdIn.objChgAuth));
cmdIn.objChgAuth.objectHandle = hmacKey.handle;
cmdIn.objChgAuth.parentHandle = storage.handle;
cmdIn.objChgAuth.newAuth.size = WC_SHA256_DIGEST_SIZE;
rc = TPM2_GetNonce(cmdIn.objChgAuth.newAuth.buffer,
                   cmdIn.objChgAuth.newAuth.size);
if (rc < 0) {
      /* wc_RNG_GenerateBlock failed */
}
rc = TPM2_ObjectChangeAuth(&cmdIn.objChgAuth, &cmdOut.objChgAuth);
if (rc != TPM_RC_SUCCESS) {
      /* TPM2_ObjectChangeAuth failed */
}
```

See Also:

TPM2_GetRcString
TPM2_GetNonce
TPM2_FlushContext

TPM2_Duplicate

Synopsis:

#include <wolftpm/tpm2.h>

TPM_RC TPM2_Duplicate(Duplicate_In* in, Duplicate_Out* out);

Description:

This command duplicates a loaded object so that it may be used in a different hierarchy. The new parent key for the duplicate may be on the same or different TPM or TPM_RH_NULL. Only the public area of *newParentHandle* is required to be loaded.

Return Values:

If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.

BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a Duplicate_In struct
out - a pointer to a Duplicate_Out struct

Example:

See Also:

TPM2_Rewrap

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_Rewrap(Rewrap_In* in, Rewrap_Out* out);

Description:
This command allows the TPM to serve in the role as a Duplication Authority. If proper authorization for use of the *oldParent* is provided, then an HMAC key and a symmetric key are recovered from *inSymSeed* and used to integrity check and decrypt *inDuplicate*. A new protection seed value is generated according to the methods appropriate for *newParent* and the blob is re-encrypted and a new integrity value is computed. The re-encrypted blob is returned in *outDuplicate* and the symmetric key returned in *outSymKey*.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a Rewrap_In struct
out - a pointer to a Rewrap_Out struct

Example:

See Also:

TPM2_Import

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_Import(Import_In* in, Import_Out* out);

Description:
This command allows an object to be encrypted using the symmetric encryption values of a Storage Key. After encryption, the object may be loaded and used in the new hierarchy. The imported object (*duplicate*) may be singly encrypted, multiply encrypted, or unencrypted.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a Import_In struct
out - a pointer to a Import_Out struct

Example:

See Also:

TPM2_RSA_Encrypt

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_RSA_Encrypt(RSA_Encrypt_In* in, RSA_Encrypt_Out* out);

Description:
This command performs RSA encryption using the padding scheme indicated in the **in** parameter's *inScheme*.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a RSA_Encrypt_In struct
out - a pointer to a RSA_Encrypt_Out struct

Example:
```
XMEMSET(&cmdIn.rsaEnc, 0, sizeof(cmdIn.rsaEnc));
cmdIn.rsaEnc.keyHandle = rsaKey.handle;
cmdIn.rsaEnc.message = message;
cmdIn.rsaEnc.inScheme.scheme = TPM_ALG_OAEP;
cmdIn.rsaEnc.inScheme.details.oaep.hashAlg = TPM_ALG_SHA256;
cmdIn.rsaEnc.label.size = sizeof(label); /* Null term required */
XMEMCPY(cmdIn.rsaEnc.label.buffer, label, cmdIn.rsaEnc.label.size);
rc = TPM2_RSA_Encrypt(&cmdIn.rsaEnc, &cmdOut.rsaEnc);
if (rc != TPM_RC_SUCCESS) {
    /* TPM2_RSA_Encrypt failed */
}
```

See Also:
TPM2_RSA_Encrypt
TPM2_GetRCString
TPM2_FlushContext

TPM2_RSA_Decrypt

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_RSA_Decrypt(RSA_Decrypt_In* in, RSA_Decrypt_Out* out);

Description:
This command performs RSA decryption

Return Values:

If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:

in - a pointer to a RSA_Decrypt_In struct
out - a pointer to a RSA_Decrypt_Out struct

Example:

```
XMEMSET(&cmdIn.rsaDec, 0, sizeof(cmdIn.rsaDec));
cmdIn.rsaDec.keyHandle = rsaKey.handle;
cmdIn.rsaDec.cipherText = cmdOut.rsaEnc.outData;
cmdIn.rsaDec.inScheme.scheme = TPM_ALG_OAEP;
cmdIn.rsaDec.inScheme.details.oaep.hashAlg = TPM_ALG_SHA256;
cmdIn.rsaDec.label.size = sizeof(label); /* Null term required */
XMEMCPY(cmdIn.rsaDec.label.buffer, label, cmdIn.rsaEnc.label.size);
rc = TPM2_RSA_Decrypt(&cmdIn.rsaDec, &cmdOut.rsaDec);
if (rc != TPM_RC_SUCCESS) {
      /* TPM2_RSA_Decrypt failed */
}
```

See Also:

TPM2_RSA_Encrypt
TPM2_GetRCString
TPM2_FlushContext

TPM2_ECDH_KeyGen

Synopsis:

#include <wolftpm/tpm2.h>

TPM_RC TPM2_ECDH_KeyGen(ECDH_KeyGen_In* in, ECDH_KeyGen_Out* out);

Description:

This command uses the TPM to generate an ephemeral key pair.

Return Values:

If successful the call will return **TPM_RC_SUCCESS**.

TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a ECDH_KeyGen_In struct
out - a pointer to a ECDH_KeyGen_Out struct

Example:
```
XMEMSET(&cmdIn.ecdh, 0, sizeof(cmdIn.ecdh));
cmdIn.ecdh.keyHandle = eccKey.handle;
rc = TPM2_ECDH_KeyGen(&cmdIn.ecdh, &cmdOut.ecdh);
if (rc != TPM_RC_SUCCESS) {
      /* TPM2_ECDH_KeyGen failed */
}
```

See Also:
TPM2_GetRCString
TPM2_FlushContext

TPM2_ECDH_ZGen

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_ECDH_ZGen(ECDH_ZGen_In* in, ECDH_ZGen_Out* out);

Description:
This command uses the TPM to recover the Z value from a public point and a private key.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a ECDH_ZGen_In struct
out - a pointer to a ECDH_ZGen_Out struct

Example:

```
XMEMSET(&cmdIn.ecdhZ, 0, sizeof(cmdIn.ecdhZ));
cmdIn.ecdhZ.keyHandle = eccKey.handle;
cmdIn.ecdhZ.inPoint = cmdOut.ecdh.pubPoint;
rc = TPM2_ECDH_ZGen(&cmdIn.ecdhZ, &cmdOut.ecdhZ);
if (rc != TPM_RC_SUCCESS) {
      /* TPM2_ECDH_ZGen failed */
}
```

See Also:

TPM2_GetRCString
TPM2_FlushContext

TPM2_ECC_Parameters

Synopsis:

#include <wolftpm/tpm2.h>

TPM_RC TPM2_ECC_Parameters(ECC_Parameters_In* in,
ECC_Parameters_Out* out);

Description:

Used for retrieving the parameters of an ECC curve (curve ID, key size, etc.)

Return Values:

If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:

in - a pointer to a ECC_Parameters_In struct
out - a pointer to a ECC_Parameters_Out struct

Example:

```
XMEMSET(&cmdIn.eccParam, 0, sizeof(cmdIn.eccParam));
cmdIn.eccParam.curveID = TPM_ECC_NIST_P256;
rc = TPM2_ECC_Parameters(&cmdIn.eccParam, &cmdOut.eccParam);
if (rc != TPM_RC_SUCCESS) {
      /* TPM2_ECC_Parameters failed */
}
```

See Also:
TPM2_GetRCString
TPM2_FlushContext

TPM2_ZGen_2Phase

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_ZGen_2Phase(ZGen_2Phase_In* in, ZGen_2Phase_Out* out);

Description:

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a ZGen_2Phase_In struct
out - a pointer to a ZGen_2Phase_Out struct

Example:

See Also:

TPM2_EncryptDecrypt

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_EncryptDecrypt(EncryptDecrypt_In* in, EncryptDecrypt_Out* out);

Description:

This command performs symmetric encryption or decryption.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a EncryptDecrypt_In struct
out - a pointer to a EncryptDecrypt_Out struct

Example:

See Also:

TPM2_EncryptDecrypt2

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_EncryptDecrypt(EncryptDecrypt2_In* in, EncryptDecrypt2_Out* out);

Description:

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a EncryptDecrypt2_In struct
out - a pointer to a EncryptDecrypt2_Out struct

Example:

See Also:

TPM2_Hash

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_Hash(Hash_In* in, Hash_Out* out);

Description:
This command performs a hash operation on a data buffer and returns the results.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a Hash_In struct
out - a pointer to a Hash_Out struct

Example:

See Also:

TPM2_HMAC

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_HMAC(HMAC_In* in, HMAC_Out* out);

Description:
This command performs an HMAC on the supplied data using the indicated hashing algorithm.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a Hash_In struct
out - a pointer to a Hash_Out struct

Example:

See Also:
TPM2_HMAC_Start

TPM2_HMAC_Start

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_HMAC_Start(HMAC_Start_In* in, HMAC_Start_Out* out);

Description:
This command starts an HMAC sequence.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a HMAC_Start_In struct
out - a pointer to a HMAC_Start_Out struct

Example:

See Also:

TPM2_HashSequenceStart

Synopsis:

#include <wolftpm/tpm2.h>

TPM_RC TPM2_HashSequenceStart(HashSequenceStart_In* in,
HashSequenceStart_Out* out);

Description:

This command starts a hash or an Event Sequence.

Return Values:

If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:

in - a pointer to a HashSequenceStart_In struct
out - a pointer to a HashSequenceStart_Out struct

Example:

```
XMEMSET(&cmdIn.hashSeqStart, 0, sizeof(cmdIn.hashSeqStart));
cmdIn.hashSeqStart.auth.size = sizeof(usageAuth)-1;
XMEMCPY(cmdIn.hashSeqStart.auth.buffer, usageAuth,
    cmdIn.hashSeqStart.auth.size);
cmdIn.hashSeqStart.hashAlg = TPM_ALG_SHA256;
rc = TPM2_HashSequenceStart(&cmdIn.hashSeqStart, &cmdOut.hashSeqStart);
if (rc != TPM_RC_SUCCESS) {
    /* TPM2_HashSequenceStart failed */
}
```

See Also:

TPM2_HMAC
TPM2_SequenceUpdate
TPM2_SequenceComplete

TPM2_SequenceUpdate

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_SequenceUpdate(SequenceUpdate_In* in);

Description:
This command is used to add data to a hash or an HMAC sequence.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a SequenceUpdate_In struct

Example:
```
XMEMSET(&cmdIn.seqUpdate, 0, sizeof(cmdIn.seqUpdate));
cmdIn.seqUpdate.sequenceHandle = handle;
cmdIn.seqUpdate.buffer.size = XSTRLEN(hashTestData);
XMEMCPY(cmdIn.seqUpdate.buffer.buffer, hashTestData,
    cmdIn.seqUpdate.buffer.size);
rc = TPM2_SequenceUpdate(&cmdIn.seqUpdate);
if (rc != TPM_RC_SUCCESS) {
      /* TPM2_SequenceUpdate failed*/
}
```

See Also:
TPM2_HashSequenceStart
TPM2_SequenceComplete

TPM2_SequenceComplete

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_SequenceComplete(SequenceComplete_In* in,

SequenceComplete_Out* out);

Description:

This command adds the last part of data, if any, to a hash/HMAC sequence and writes the result to the **out** parameter.

Return Values:

If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:

in - a pointer to a SequenceComplete_In struct
out - a pointer to a SequenceComplete_Out struct

Example:

```
XMEMSET(&cmdIn.seqComp, 0, sizeof(cmdIn.seqComp));
cmdIn.seqComp.sequenceHandle = handle;
cmdIn.seqComp.hierarchy = TPM_RH_NULL;
rc = TPM2_SequenceComplete(&cmdIn.seqComp, &cmdOut.seqComp);
if (rc != TPM_RC_SUCCESS) {
/*
}
```

If `TPM2_SequenceComplete` was used after successful calls to `TPM2_HashSequenceStart` and `TPM2_SequenceUpdate`, then it is possible to test if the test has passed using the following `if` branch (as an example):

```
if (cmdOut.seqComp.result.size != WC_SHA256_DIGEST_SIZE &&
    XMEMCMP(cmdOut.seqComp.result.buffer, hashTestDig,
                                          WC_SHA256_DIGEST_SIZE) != 0) {
      /* Hash SHA256 test failed due to result differing from expectation */
}
```

See Also:

TPM2_HashSequenceStart
TPM2_SequenceUpdate

TPM2_EventSequenceComplete

Synopsis:

#include <wolftpm/tpm2.h>

TPM_RC TPM2_SequenceComplete(EventSequenceComplete_In* in,
EventSequenceComplete_Out* out);

Description:
This command adds the las part of data, if any, to an Event Sequence and writes the result to the TPML_DIGEST_VALUES *results* member of the **out** parameter.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a EventSequenceComplete_In struct
out - a pointer to a EventSequenceComplete_In struct

Example:

See Also:
TPM2_HashSequenceStart
TPM2_SequenceUpdate
TPM2_SequenceComplete

TPM2_Certify

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_Certify(Certify_In* in, Certify_Out* out);

Description:
The purpose of this command is to prove that an object with a specific name is loaded into the TPM.

Return Values:

If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a Certify_In struct
out - a pointer to a Certify_Out struct

Example:

See Also:
TPM2_CertifyCreation

TPM2_CertifyCreation

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_Certify(CertifyCreation_In* in, CertifyCreation_Out* out);

Description:
This command is used to prove the association between an object and its creation data.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a CertifyCreation_In struct
out - a pointer to a CertifyCreation_Out struct

Example:

See Also:
TPM2_Certify

TPM2_Quote

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_Certify(Quote_In* in, Quote_Out* out);

Description:
This command quotes PCR values.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a Quote_In struct
out - a pointer to a Quote_Out struct

Example:

See Also:

TPM2_GetSessionAuditDigest

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_GetSessionAuditDigest(GetSessionAuditDigest_In* in,
GetSessionAuditDigest_Out* out);

Description:
This command writes the digital signature of the audit session digest to the **out** parameter.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a GetSessionAuditDigest_In struct
out - a pointer to a GetSessionAuditDigest_Out struct

Example:

See Also:
TPM2_GetCommandAuditDigest
TPM2_SetCommandCodeAuditStatus

TPM2_GetCommandAuditDigest

Synopsis:
#include <wolftpm/tpm2.h>

```
TPM_RC TPM2_GetCommandAuditDigest(GetCommandAuditDigest_In* in,
                                  GetCommandAuditDigest_Out* out);
```

Description:
This command writes the current value of the command audit digest, a digest of the commands being audited, and the audit hash algorithm to the **out** parameter and signs these values with the key referenced by the **in** parameter's TPMT_SIG_SCHEME *inScheme* member.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a GetSessionAuditDigest_In struct

out - a pointer to a GetSessionAuditDigest_Out struct

Example:

See Also:
TPM2_GetSessionAuditDigest
TPM2_SetCommandCodeAuditStatus

TPM2_GetTime

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_GetTime(GetTime_In* in, GetTime_Out* out);

Description:
Writes the current time values of Time and Clock to the **out** parameter.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a GetTime_In struct
out - a pointer to a GetTime_Out struct

Example:

See Also:

TPM2_Commit

Synopsis:

#include <wolftpm/tpm2.h>

TPM_RC TPM2_Commit(Commit_In* in, Commit_Out* out);

Description:
Performs the first part of an ECC anonymous signing operation.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a Commit_In struct
out - a pointer to a Commit_Out struct

Example:

See Also:

TPM2_EC_Ephemeral

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_EC_Ephemeral(EC_Ephemeral_In* in, EC_Ephemeral_Out* out);

Description:
Creates an ephemeral key for use in a two-phase key exchange protocol.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:

in - a pointer to a EC_Ephemeral_In struct
out - a pointer to a EC_Ephemeral_Out struct

Example:

See Also:

TPM2_VerifySignature

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_VerifySignature(VerifySignature_In* in, VerifySignature_Out* out);

Description:
Uses loaded keys to validate a signature on a message with the message digest passed to the TPM.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a VerifySignature_In struct
out - a pointer to a VerifySignature_Out struct

Example:

```
XMEMSET(&cmdIn.verifySign, 0, sizeof(cmdIn.verifySign));
cmdIn.verifySign.keyHandle = eccKey.handle;
cmdIn.verifySign.digest.size = message.size;
XMEMCPY(cmdIn.verifySign.digest.buffer, message.buffer, message.size);
cmdIn.verifySign.signature = cmdOut.sign.signature;
rc = TPM2_VerifySignature(&cmdIn.verifySign, &cmdOut.verifySign);
if (rc != TPM_RC_SUCCESS) {
      /* TPM2_VerifySignature failed */
}
```

See Also:
TPM2_GetRCString
TPM2_FlushContext

TPM2_Sign

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_Sign(Sign_In* in, Sign_Out* out);

Description:
This command causes the TPM to sign an externally provided hash with the specified asymmetric signing key.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a Sign_In struct
out - a pointer to a Sign_Out struct

Example:
```
XMEMSET(&cmdIn.sign, 0, sizeof(cmdIn.sign));
cmdIn.sign.keyHandle = eccKey.handle;
cmdIn.sign.digest.size = message.size;
XMEMCPY(cmdIn.sign.digest.buffer, message.buffer, message.size);
cmdIn.sign.inScheme.scheme = TPM_ALG_ECDSA;
cmdIn.sign.inScheme.details.ecdsa.hashAlg = TPM_ALG_SHA256;
cmdIn.sign.validation.tag = TPM_ST_HASHCHECK;
cmdIn.sign.validation.hierarchy = TPM_RH_NULL;
rc = TPM2_Sign(&cmdIn.sign, &cmdOut.sign);
if (rc != TPM_RC_SUCCESS) {
      /* TPM2_Sign failed */
}
```

See Also:
TPM2_GetRCString

TPM2_FlushContext

TPM2_SetCommandCodeAuditStatus

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_SetCommandCodeAuditStatus(SetCommandCodeAuditStatus_In* in);

Description:
This command may be used by the privacy administrator or platform to change the audit status of a command or to set the hash algorithm used for the audit digest, but not both simultaneously.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a SetCommandCodeAuditStatus_In struct

Example:

See Also:
TPM2_GetCommandAuditDigest
TPM2_GetSessionAuditDigest
TPM2_GetRCString
TPM2_FlushContext

TPM2_PCR_Event

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_PCR_Event(PCR_Event_In* in, PCR_Event_Out* out);

Description:
This command is used to cause an update to the indicated PCR.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a PCR_Event_In struct
out - a pointer to a PCR_Event_Out struct

Example:

See Also:
TPM2_PCR_Allocate
TPM2_PCR_SetAuthPolicy
TPM2_PCR_SetAuthValue
TPM2_PCR_Reset
TPM2_GetRCString
TPM2_FlushContext

TPM2_PCR_Allocate

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_PCR_Allocate(PCR_Allocate_In* in, PCR_Allocate_Out* out);

Description:
This command is used to set the desired PCR allocation of PCR and algorithms, and requires Platform Authorization.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.

BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a PCR_Allocate_In struct
out - a pointer to a PCR_Allocate_Out struct

Example:

See Also:
TPM2_PCR_Event
TPM2_PCR_SetAuthPolicy
TPM2_PCR_SetAuthValue
TPM2_PCR_Reset
TPM2_GetRCString
TPM2_FlushContext

TPM2_PCR_SetAuthPolicy

Synopsis:
```
#include <wolftpm/tpm2.h>

TPM_RC TPM2_PCR_SetAuthPolicy(PCR_SetAuthPolicy_In* in,
                              PCR_SetAuthPolicy_Out* out);
```

Description:
This command is used to associate a policy with a PCR or group of PCR. The policy determines the conditions under which a PCR may be extended or reset.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a PCR_SetAuthPolicy_In struct
out - a pointer to a PCR_SetAuthPolicy_Out struct

Example:

See Also:
TPM2_PCR_Event
TPM2_PCR_Allocate
TPM2_PCR_SetAuthValue
TPM2_PCR_Reset
TPM2_GetRCString
TPM2_FlushContext

TPM2_PCR_SetAuthValue

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_PCR_SetAuthValue(PCR_SetAuthValue_In* in);

Description:
This command is used to associate a policy with a PCR or group of PCR. The policy determines the conditions under which a PCR may be extended or reset.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a PCR_SetAuthValue_In struct

Example:

See Also:
TPM2_PCR_Event
TPM2_PCR_Allocate
TPM2_PCR_SetAuthPolicy
TPM2_PCR_Reset

TPM2_GetRCString
TPM2_FlushContext

TPM2_PCR_Reset

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_PCR_Reset(PCR_Reset_In* in);

Description:
If the attribute of a PCR allows the PCR to be reset and proper authorization is provided, then this command may be used to set the PCR to zero. The PCR attributes may restrict the localities that can perform the operation.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a PCR_Reset_In struct

Example:

See Also:
TPM2_PCR_Event
TPM2_PCR_Allocate
TPM2_PCR_SetAuthValue
TPM2_PCR_Reset
TPM2_GetRCString
TPM2_FlushContext

TPM2_PolicySigned

Synopsis:

#include <wolftpm/tpm2.h>

TPM_RC TPM2_PolicySigned(PolicySigned_In* in, PolicySigned_Out* out);

Description:
Includes a signed authorization in a policy. It ties the policy to a signing key by including the name of the signing key in the **out** parameter.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a PolicySigned_In struct
out - a pointer to a PolicySigned_Out struct

Example:

See Also:
TPM2_PolicySecret
TPM2_PolicyTicket
TPM2_GetRCString
TPM2_FlushContext

TPM2_PolicySecret

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_PolicySecret(PolicySecret_In* in, PolicySecret_Out* out);

Description:
This command includes a secret-based authorization to a policy.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.

TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a PolicySecret_In struct
out - a pointer to a PolicySecret_Out struct

Example:

See Also:
TPM2_PolicySigned
TPM2_PolicyTicket
TPM2_PolicyOR
TPM2_GetRCString
TPM2_FlushContext

TPM2_PolicyTicket

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_PolicySecret(PolicySecret_In* in, PolicySecret_Out* out);

Description:
This command is similar to **TPM2_PolicySigned** except that it takes a ticket instead of a signed authorization. The ticket represents a validated authorization that had an expiration time associated with it.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a PolicySecret_In struct
out - a pointer to a PolicySecret_Out struct

Example:

See Also:
TPM2_PolicySigned
TPM2_PolicySecret
TPM2_PolicyOR
TPM2_GetRCString
TPM2_FlushContext

TPM2_PolicyOR

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_PolicyOR(PolicyOR_In* in);

Description:
Allows options in authorizations without requiring that the TPM evaluate all of the options. If a policy may be satisfied by different sets of conditions, the TPM need only evaluate one set that satisfies the policy. This command will indicate that one of the required sets of conditions has been satisfied.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a PolicyOR_In struct

Example:

See Also:
TPM2_PolicySigned
TPM2_PolicySecret
TPM2_PolicyTicket

TPM2_GetRCString
TPM2_FlushContext

TPM2_PolicyPCR

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_PolicyPCR(PolicyPCR_In* in);

Description:
This command is used to cause conditional gating of a policy based on PCR. This command together with **TPM2_PolicyOR** allows one group of authorizations to occur when PCR are in one state, and a different set of authorizations to occur when the PCR are in a different state.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a PolicyPCR_In struct

Example:
```
pcrIndex = 0;
XMEMSET(&cmdIn.policyPCR, 0, sizeof(cmdIn.policyPCR));
cmdIn.policyPCR.policySession = sessionHandle;
cmdIn.policyPCR.pcrDigest.size = hash_len;
XMEMCPY(cmdIn.policyPCR.pcrDigest.buffer, hash, hash_len);
TPM2_SetupPCRSel(&cmdIn.policyPCR.pcrs, TPM_ALG_SHA1, pcrIndex);
rc = TPM2_PolicyPCR(&cmdIn.policyPCR);
if (rc != TPM_RC_SUCCESS) {
    /* TPM2_PolicyPCR failed */
}
```

See Also:
TPM2_PolicySigned
TPM2_PolicySecret
TPM2_PolicyTicket

TPM2_SetupPCRSel
TPM2_GetRCString
TPM2_FlushContext

TPM2_PolicyLocality

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_PolicyLocality(PolicyLocality_In* in);

Description:
Indicates that the authorization will be limited to a specific locality.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a PolicyLocality_In struct

Example:

See Also:
TPM2_GetRCString
TPM2_FlushContext

TPM2_PolicyNV

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_PolicyNV(PolicyNV_In* in);

Description:

This command is used to cause conditional gating of a policy based on the contents of an NV index. The NV index is validated during this command, not when the session is used for authorization.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a PolicyNV_In struct

Example:

See Also:
TPM2_GetRCString
TPM2_FlushContext

TPM2_PolicyCounterTimer

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_PolicyCounterTimer(PolicyCounterTimer_In* in);

Description:
This command is used to cause conditional gating of a policy based on the contents of the **in** parameter.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a PolicyCounterTimer_In struct

Example:

See Also:
TPM2_GetRCString
TPM2_FlushContext

TPM2_PolicyCommandCode

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_PolicyCommandCode(PolicyCommandCode_In* in);

Description:
Indicates that the authorization will be limited to a specific command code.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a PolicyCommandCode_In struct

Example:
```
XMEMSET(&cmdIn.policyCC, 0, sizeof(cmdIn.policyCC));
cmdIn.policyCC.policySession = sessionHandle;
cmdIn.policyCC.code = TPM_CC_ObjectChangeAuth;
rc = TPM2_PolicyCommandCode(&cmdIn.policyCC);
if (rc != TPM_RC_SUCCESS) {
      /* TPM2_PolicyCommandCode failed */
}
```

See Also:
TPM2_GetRCString
TPM2_FlushContext

TPM2_PolicyPhysicalPresence

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_PolicyPhysicalPresence(PolicyPhysicalPresence_In* in);

Description:
Indicates that the physical presence will need to be asserted at the time the authorization is performed.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a PolicyPhysicalPresence_In struct

Example:

See Also:
TPM2_PolicyAuthValue
TPM2_PolicyPassword
TPM2_GetRCString
TPM2_FlushContext

TPM2_PolicyCpHash

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_PolicyCpHash(PolicyCpHash_in* in);

Description:
This command is used to allow a policy to be bound to a specific command and

command parameters.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:

in - a pointer to a PolicyCpHash_In struct

Example:

See Also:
TPM2_GetRCString
TPM2_FlushContext

TPM2_PolicyNameHash

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_PolicyNameHash(PolicyNameHash_in* in);

Description:
Allows a policy to be bound to a specific set of TPM entities without being bound to the parameters of the command. This is most useful for commands when referenced PCR requires a policy.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a PolicyNameHash_In struct

Example:

See Also:
TPM2_GetRCString
TPM2_FlushContext

TPM2_PolicyDuplicationSelect

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_PolicyNameHash(PolicyNameHash_in* in);

Description:
This command allows qualification of duplication to allow duplication to a selected new parent. If this command is not used in conjunction with **TPM2_PolicyAuthorize**, then only the new parent is selected.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a PolicyNameHash_In struct

Example:

See Also:
TPM2_GetRCString
TPM2_FlushContext

TPM2_PolicyAuthorize

Synopsis:

#include <wolftpm/tpm2.h>

TPM_RC TPM2_PolicyAuthorize(PolicyAuthorize_in* in);

Description:
This command allows policies to change. If policies were static, it would be difficult to add users to a policy.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a PolicyAuthorize_In struct

Example:

See Also:
TPM2_PolicyAuthorizeNV
TPM2_GetRCString
TPM2_FlushContext

TPM2_PolicyAuthorizeNV

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_PolicyAuthorizeNV(PolicyAuthorizeNV_in* in);

Description:
???

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a PolicyAuthorizeNV_In struct

Example:

See Also:
TPM2_PolicyAuthorize
TPM2_GetRCString
TPM2_FlushContext

TPM2_PolicyAuthValue

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_PolicyAuthValue(PolicyAuthValue_in* in);

Description:
Allows a policy to be bound to the authorization value of the authorized object.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a PolicyAuthValue_In struct

Example:

See Also:
TPM2_PolicyPhysicalPresence
TPM2_PolicyPassword
TPM2_GetRCString
TPM2_FlushContext

TPM2_PolicyPassword

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_PolicyPassword(PolicyPassword_in* in);

Description:
Allows a policy to be bound to the authorization value of the authorized object.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a PolicyPassword_In struct

Example:

See Also:
TPM2_PolicyPhysicalPresence
TPM2_PolicyAuthValue
TPM2_GetRCString
TPM2_FlushContext

TPM2_PolicyGetDigest

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_PolicyGetDigest(PolicyGetDigest_in* in, PolicyGetDigest_Out* out);

Description:
This command writes the current *policyDigest* of the session to the **out** parameter's

TPMI_SH_POLICY *policyDigest* member.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a PolicyGetDigest_In struct
out - a pointer to a PolicyGetDigest_Out struct

Example:
```
XMEMSET(&cmdIn.policyGetDigest, 0, sizeof(cmdIn.policyGetDigest));
cmdIn.policyGetDigest.policySession = sessionHandle;
rc = TPM2_PolicyGetDigest(&cmdIn.policyGetDigest, &cmdOut.policyGetDigest);
if (rc != TPM_RC_SUCCESS) {
      /* TPM2_PolicyGetDigest failed */
}
```

See Also:
TPM2_PrintBin
TPM2_GetRCString
TPM2_FlushContext

TPM2_PolicyNvWritten

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_PolicyNvWritten(PolicyNvWritten_In* in);

Description:
This command allows a policy to be bound to the TPMA_NV_WRITTEN attributes. This is a deferred assertion. Values are stored in the policy session context and checked when the policy is used for authorization.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.

BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a PolicyNvWritten_In struct

Example:

See Also:
TPM2_GetRCString
TPM2_FlushContext

TPM2_PolicyTemplate

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_PolicyTemplate(PolicyTemplate_In* in);

Description:

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a PolicyTemplate_In struct

Example:

See Also:
TPM2_GetRCString
TPM2_FlushContext

TPM2_PolicyAuthorizeNV

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_PolicyAuthorizeNV(PolicyAuthorizeNV_In* in);

Description:

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a PolicyAuthorizeNV_In struct

Example:

See Also:
TPM2_GetRCString
TPM2_FlushContext

_TPM_Hash_Start

Synopsis:
#include <wolftpm/tpm2.h>

void _TPM_Hash_Start(void);

Description:
This command is typically triggered by hardware events, and it doesn't return an error code if no slots are available. Instead, it kicks an object out and provides no indication

of which object was evicted.[3]

Return Values:
This function returns void

Parameters:
This function takes no arguments as its parameters.

Example:

See Also:
_TPM_Hash_Data
_TPM_Hash_End

_TPM_Hash_Data

Synopsis:
#include <wolftpm/tpm2.h>

void _TPM_Hash_Data(UINT32 dataSize, BYTE* data);

Description:
This function is a special hardware-triggered command. The indication from the TPM interface indicates arrival of one or more octets of data that are to be included in the Core Root of Trust for Management (CRTM) sequence context created by the **_TPM_Hash_Start** indication. The context holds data for each hash algorithm for each PCR bank implemented on the TPM.

Return Values:
This function returns void

Parameters:
dataSize - an unsigned 32-bit integer that contains the size of the data pointed to by the **data** argument.
data - a pointer to a BYTE containing data

[3] Arthur, Will, and David Challener. "Management of Objects, Sessions, and Sequences." *A Practical Guide to TPM 2.0: Using the Trusted Platform Module in the New Age of Security*, Apress, 2015, p. 294.

Example:

See Also:
_TPM_Hash_Start
_TPM_Hash_End

_TPM_Hash_End

Synopsis:
#include <wolftpm/tpm2.h>

void _TPM_Hash_End(void);

Description:
This indication from the TPM interface indicates the end of the H-CRTM measurement. This indication is discarded and no other action is performed if the TPM does not contain a H-CRTM Event Sequence context.

Return Values:
This function returns void

Parameters:
This function takes no arguments as its parameters

Example:

See Also:
_TPM_Hash_Start
_TPM_Hash_Data

TPM2_HierarchyControl

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_HierarchyControl(HierarchyControl_In* in);

Description:
This command enables and disables use of a hierarchy and its associated NV storage. The command allows *phEnable*, *phEnableNV*, *shEnable*, and *ehEnable* to be changed when the proper authorization is provided.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a HierarchyControl_In struct

Example:

See Also:
TPM2_GetRCString
TPM2_FlushContext

TPM2_SetPrimaryPolicy

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_SetPrimaryPolicy(SetPrimaryPolicy_In* in);

Description:
This command allows setting of the authorization policy for the lockout (*lockoutPolicy*), the platform hierarchy (*platformPolicy*), the storage hierarchy (*ownerPolicy*), and the endorsement hierarchy (*endorsementPolicy*).

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.

BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a SetPrimaryPolicy_In struct

Example:

See Also:
TPM2_GetRCString
TPM2_FlushContext

TPM2_ChangePPS

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_ChangePPS(ChangePPS_In* in);

Description:
This function calls TPM2_ChangeSeed with the **in** parameter, and the ChangePPS command code.
This replaces the current PPS with a value from the RNG and sets *platformPolicy* to the default initialization value (the Empty Buffer).

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a ChangePPS_In struct

Example:

See Also:
TPM2_ChangeEPS

TPM2_ChangeSeed
TPM2_GetRCString
TPM2_FlushContext

TPM2_ChangeEPS

Synopsis:

```
#include <wolftpm/tpm2.h>

TPM_RC TPM2_ChangeEPS(ChangeEPS_In* in);
```

Description:

This function calls TPM2_ChangeSeed with the **in** parameter, and the ChangeEPS command code.
This replaces the current EPS with a value from the RNG and sets the Endorsement hierarchy controls to their default initialization values. It will flush any resident objects in the EPS hierarchy and not allow objects in the hierarchy associated with the previous EPS to be loaded.

ehEnable default value: SET
endorsementAuth default value: Empty Buffer
endorsementPolicy default value: Empty Buffer

Return Values:

If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:

in - a pointer to a ChangeEPS_In struct

Example:

See Also:

TPM2_ChangePPS
TPM2_ChangeSeed
TPM2_GetRCString

TPM2_FlushContext

TPM2_Clear

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_Clear(Clear_In* in);

Description:
Removes all TPM context associated with a specific Owner.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a Clear_In struct

Example:
```
cmdIn.clear.authHandle = TPM_RH_PLATFORM;
rc = TPM2_Clear(&cmdIn.clear);
if (rc != TPM_RC_SUCCESS) {
      /* TPM2_Clear failed */
}
```

See Also:
TPM2_GetRCString
TPM2_FlushContext

TPM2_ClearControl

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_ClearControl(ClearControl_In* in);

Description:
Disables and enables the execution of **TPM2_Clear**.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a ClearControl_In struct

Example:

See Also:
TPM2_GetRCString
TPM2_FlushContext

TPM2_HierarchyChangeAuth

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_HierarchyChangeAuth(HierarchyChangeAuth_In* in);

Description:
Allows the authorization secret for a hierarchy or lockout to be changed using the current authorization value as the command authorization.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a HierarchyChangeAuth_In struct

Example:

See Also:
TPM2_GetRCString
TPM2_FlushContext

TPM2_DictionaryAttackLockReset

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_DictionaryAttackLockReset(DictionaryAttackLockReset_In* in);

Description:
This command cancels the effect of a TPM lockout due to a number of successive authorization failures. If this command is properly authorized, the lockout counter is set to zero.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a DictionaryAttackLockReset_In struct

Example:

See Also:
TPM2_DictionaryAttackParameters
TPM2_GetRCString
TPM2_FlushContext

TPM2_DictionaryAttackParameters

Synopsis:

#include <wolftpm/tpm2.h>

TPM_RC TPM2_DictionaryAttackParameters(DictionaryAttackParameters_In* in);

Description:
This command changes the lockout parameters, and requires Lockout Authorization.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a DictionaryAttackParameters_In struct

Example:

See Also:
TPM2_DictionaryAttackLockReset
TPM2_GetRCString
TPM2_FlushContext

TPM2_PP_Commands

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_PP_Commands(PP_Commands_In* in);

Description:
This command is used to determine which commands require assertion of Physical Presence (PP) in addition to *platformAuth*/*platformPolicy*.[4]

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.

[4] Trusted Platform Module Library Family Part 3: Commands. p. 306

BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a PP_Commands_In struct

Example:

See Also:
TPM2_GetRCString
TPM2_FlushContext

TPM2_SetAlgorithmSet

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_SetAlgorithmSet(SetAlgorithmSet_In* in);

Description:
This command allows the platform to change the set of algorithms that are used by the TPM. The *algorithmSet* setting is a vendor-dependent value.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a SetAlgorithmSet_In struct

Example:

See Also:
TPM2_GetRCString
TPM2_FlushContext

TPM2_FieldUpgradeStart

Synopsis:

```
#include <wolftpm/tpm2.h>

TPM_RC TPM2_FieldUpgradeStart(FieldUpgradeStart_In* in);
```

Description:

This command uses *platformPolicy* and a TPM Vendor Authorization Key to authorize a Field Upgrade Manifest.
If the signature checks succeed, the authorization is valid and the TPM will accept **TPM2_FieldUpgradeData**.

Return Values:

If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:

in - a pointer to a FieldUpgradeStart_In struct

Example:

See Also:

TPM2_FIeldUpgradeData
TPM2_GetRCString
TPM2_FlushContext

TPM2_FieldUpgradeData

Synopsis:

```
#include <wolftpm/tpm2.h>

TPM_RC TPM2_FieldUpgradeData(FieldUpgradeData_In* in,
                             FieldUpgradeData_Out* out);
```

Description:
This command will take the actual field upgrade image to be installed on the TPM. The exact format of *fuData* is vendor-specific.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a FieldUpgradeData_In struct
out - a pointer to a FieldUpgradeData_Out struct

Example:

See Also:
TPM2_FieldUpgradeStart
TPM2_GetRCString
TPM2_FlushContext

TPM2_FirmwareRead

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_FirmwareRead(FirmwareRead_In* in, FirmwareRead_Out* out);

Description:
This command is used to read a copy of the current firmware installed in the TPM.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a FirmwareRead_In struct

out - a pointer to a FirmwareRead_Out struct

Example:

See Also:
TPM2_GetRCString
TPM2_FlushContext

TPM2_ContextSave

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_ContextSave(ContextSave_In* in, ContextSave_Out* out);

Description:
This command saves a session context, object context, or sequence object context outside the TPM.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a ContextSave_In struct
out - a pointer to a ContextSave_Out struct

Example:

See Also:
TPM2_ContextLoad
TPM2_GetRCString
TPM2_FlushContext

TPM2_ContextLoad

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_ContextLoad(ContextLoad_In* in, ContextLoad_Out* out);

Description:
This command is used to reload a context that has been saved by **TPM2_ContextSave.**

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a ContextLoad_In struct
out - a pointer to a ContextLoad_Out struct

Example:

See Also:
TPM2_ContextSave
TPM2_GetRCString
TPM2_FlushContext

TPM2_EvictControl

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_EvictControl(EvictControl_In* in);

Description:
This command allows certain Transient Objects to be made persistent or a persistent object to be evicted.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a EvictControl_In struct

Example:
```
cmdIn.evict.auth = endorse.handle;
cmdIn.evict.objectHandle = storage.handle;
cmdIn.evict.persistentHandle;
rc = TPM2_EvictControl(&cmdIn.evict);
if (rc != TPM_RC_SUCCESS) {
      /* TPM2_EvictControl failed */
}
```

See Also:
TPM2_GetRCString
TPM2_FlushContext

TPM2_ReadClock

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_ReadClock(ReadClock_Out* out);

Description:
This command reads the given TPMS_TIME_INFO structure.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
out - a pointer to a ReadClock_Out struct that will provide the current data of *time*,

clock, *resetCount*, and *restartCount*.

Example:

See Also:
TPM2_ClockSet
TPM2_ClockRateAdjust
TPM2_GetRCString
TPM2_FlushContext

TPM2_ClockSet

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_ClockSet(ClockSet_In* in);

Description:
This command is used to advance the value of the TPM's Clock. The command will fail if the new time is less than the current value of Clock or if the new time is greater than

$$\texttt{FF FF 00 00 00 00 00 00}_{16}$$

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a ClockSet_In struct which contains an unsigned 64-bit integer member titled *newTime*, which is the actual new time to to set the TPM clock to.

Example:

See Also:
TPM2_ReadClock
TPM2_ClockRateAdjust
TPM2_GetRCString
TPM2_FlushContext

TPM2_ClockRateAdjust

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_ClockRateAdjust(ClockRateAdjust_In* in);

Description:
This command adjusts the rate of advance of *Clock* and *Time* to provide a better approximation to real time.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a ClockRateAdjust_In struct which contains a TPM_CLOCK_ADJUST (unsigned 8-bit integer) *rateAdjust* member

Example:

See Also:
TPM2_ReadClock
TPM2_ClockSet
TPM2_GetRCString
TPM2_FlushContext

TPM2_TestParms

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_TestParms(TestParms_In* in);

Description:

This command is used to check to see if specific combinations of algorithm parameters are supported.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to a TestParms_In struct

Example:

See Also:
TPM2_GetRCString
TPM2_FlushContext

TPM2_NV_DefineSpace

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_NV_DefineSpace(NV_DefineSpace_In* in);

Description:
This command defines the attributes of an NV Index and causes the TPM to reserve space to hold the data associated with the NV Index.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to an NV_DefineSpace_In struct

Example:
```
nvIndex = TPM_20_OWNER_NV_SPACE + 0x003FFFFF; /* Last owner Index */
```

```
XMEMSET(&cmdIn.nvDefine, 0, sizeof(cmdIn.nvDefine));
cmdIn.nvDefine.authHandle = storage.handle;
cmdIn.nvDefine.auth.size = sizeof(usageAuth)-1;
XMEMCPY(cmdIn.nvDefine.auth.buffer, usageAuth, cmdIn.nvDefine.auth.size);
cmdIn.nvDefine.publicInfo.nvPublic.nvIndex = nvIndex;
cmdIn.nvDefine.publicInfo.nvPublic.nameAlg = TPM_ALG_SHA256;
cmdIn.nvDefine.publicInfo.nvPublic.attributes = (
    TPMA_NV_OWNERWRITE | TPMA_NV_OWNERREAD | TPMA_NV_NO_DA | TPMA_NV_ORDERLY);
cmdIn.nvDefine.publicInfo.nvPublic.dataSize = WC_SHA256_DIGEST_SIZE;
rc = TPM2_NV_DefineSpace(&cmdIn.nvDefine);
if (rc != TPM_RC_SUCCESS) {
      /* TPM2_NV_DefineSpace failed */
}
```

See Also:
TPM2_NV_UndefineSpace
TPM2_NV_UndefineSpaceSpecial
TPM2_NV_ReadPublic
TPM2_NV_Write
TPM2_GetRCString
TPM2_FlushContext

TPM2_NV_UndefineSpace

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_NV_UndefineSpace(NV_UndefineSpace_In* in);

Description:
This command removes an Index from the TPM.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to an NV_UndefineSpace_In struct which contains a TPMI_RH_NV_INDEX (unsigned 32-bit integer) nvIndex member that refers to the

index to remove from the TPM.

Example:

```
XMEMSET(&cmdIn.nvUndefine, 0, sizeof(cmdIn.nvUndefine));
cmdIn.nvUndefine.authHandle = storage.handle;
cmdIn.nvUndefine.nvIndex = nvIndex;
rc = TPM2_NV_UndefineSpace(&cmdIn.nvUndefine);
if (rc != TPM_RC_SUCCESS) {
      /* TPM2_NV_UndefineSpace failed*/
}
```

See Also:

TPM2_NV_DefineSpace
TPM2_NV_UndefineSpaceSpecial
TPM2_NV_ReadPublic
TPM2_NV_Write
TPM2_GetRCString
TPM2_FlushContext

TPM2_NV_UndefineSpaceSpecial

Synopsis:

#include <wolftpm/tpm2.h>

TPM_RC TPM2_NV_UndefineSpaceSpecial(NV_UndefineSpaceSpecial_In* in);

Description:

This command allows removal of a platform-created NV Index that has TPMA_NV_POLICY_DELETE set.

Return Values:

If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:

in - a pointer to an NV_UndefineSpaceSpecial_In struct which contains a TPMI_RH_NV_INDEX (unsigned 32-bit integer) nvIndex member that refers to the index to remove from the TPM.

Example:

See Also:
TPM2_NV_DefineSpace
TPM2_NV_UndefineSpace
TPM2_NV_ReadPublic
TPM2_NV_Write
TPM2_GetRCString
TPM2_FlushContext

TPM2_NV_ReadPublic

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_NV_ReadPublic(NV_ReadPublic_In* in, NV_ReadPublic_Out* out);

Description:
This command is used to read the public area and Name of an NV Index. The public area of an Index is not privacy-sensitive and no authorization is required to read this data.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to an NV_ReadPublic_In struct

Example:
```
XMEMSET(&cmdIn.nvReadPub, 0, sizeof(cmdIn.nvReadPub));
cmdIn.nvReadPub.nvIndex = nvIndex;
rc = TPM2_NV_ReadPublic(&cmdIn.nvReadPub, &cmdOut.nvReadPub);
if (rc != TPM_RC_SUCCESS) {
      /* TPM2_NV_ReadPublic failed */
}
```

See Also:
TPM2_NV_DefineSpace
TPM2_NV_UndefineSpace
TPM2_NV_UndefineSpaceSpecial
TPM2_NV_Write
TPM2_GetRCString
TPM2_FlushContext

TPM2_NV_Write

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_NV_Write(NV_Write_In* in);

Description:
This command writes a value to an area in NV memory that was previously defined by **TPM2_NV_DefineSpace**.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to an NV_Write_In struct which contains the nvIndex and data to write to the space defined by TPM2_NV_DefineSpace

Example:
```
XMEMSET(&in, 0, sizeof(in));
in.authHandle = authHandle;
in.nvIndex = nvIndex;
in.offset = offset+pos;
in.data.size = towrite;
if (dataBuf) {
    XMEMCPY(in.data.buffer, &dataBuf[pos], towrite);
}
rc = TPM2_NV_Write(&in);
if (rc != TPM_RC_SUCCESS) {
      /* TPM2_NV_Write failed */
```

```
}
```

See Also:
TPM2_NV_DefineSpace
TPM2_NV_UndefineSpace
TPM2_NV_UndefineSpaceSpecial
TPM2_NV_ReadPublic
TPM2_GetRCString
TPM2_FlushContext

TPM2_NV_Increment

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_NV_Increment(NV_Increment_In* in);

Description:
This command is used to increment the value in an NV Index that has the TPM_NT_COUNTER attribute.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to an NV_Increment_In struct

Example:

See Also:
TPM2_NV_DefineSpace
TPM2_NV_UndefineSpace
TPM2_NV_UndefineSpaceSpecial
TPM2_NV_ReadPublic
TPM2_NV_Write
TPM2_GetRCString

TPM2_FlushContext

TPM2_NV_Extend

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_NV_Extend(NV_Extend_In* in);

Description:
This command extends a value to an area in NV memory that was previously defined by **TPM2_NV_DefineSpace**.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to an NV_Extend_In struct with the *data* (TPM2B_MAX_NV_BUFFER) that will be extended, and the *nvIndex* (TPMI_RH_NV_INDEX) that it will be extended to.

Example:

See Also:
TPM2_GetRCString
TPM2_FlushContext

TPM2_NV_SetBits

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_NV_SetBits(NV_SetBits_In* in);

Description:
This command is used to set bits in an NV Index that was created as a bit field. Any number of bits from 0 to 64 may be set. The contents of data are ORed with the current contents of the NV Index.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to an NV_SetBits_In struct containing the *bits* (UINT64) to set and which *nvIndex* (TPMI_RH_NV_INDEX) to set the bits within.

Example:

See Also:
TPM2_NV_Define
TPM2_NV_Undefine
TPM2_GetRCString
TPM2_FlushContext

TPM2_NV_WriteLock

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_NV_WriteLock(NV_WriteLock_In* in);

Description:
If the TPMA_NV_WRITEDEFINE or TPMA_NV_WRITE_STCLEAR attributes of an NV location are set, then this command may be used to inhibit further writes of the NV Index.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.

BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to an NV_WriteLock_In struct containing the nvIndex (TPMI_RH_NV_INDEX) to place a write-lock on.

Example:

See Also:
TPM2_NV_WriteLock
TPM2_GetRCString
TPM2_FlushContext

TPM2_NV_GlobalWriteLock

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_NV_GlobalWriteLock(NV_GlobalWriteLock_In* in);

Description:
The command will set TPMA_NV_WRITELOCKED for all indexes that have their TPMA_NV_GLOBALLOCK attribute set.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to an NV_GlobalWriteLock_In struct containing an authentication handle.

Example:

See Also:
TPM2_NV_WriteLock
TPM2_GetRCString

TPM2_FlushContext

TPM2_NV_Read

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_NV_Read(NV_Read_In* in, NV_Read_Out* out);

Description:
This command reads a value from an area in NV memory previously defined by **TPM2_NV_DefineSpace**.
Proper authorizations are required for this command as determined by TPMA_NV_PPREAD, TPMA_NV_OWNERREAD, TPMA_NV_AUTHREAD, and the *authPolicy* of the *nvIndex*.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to an NV_Read_In struct
out - a pointer to an NV_Read_Out struct

Example:

See Also:
TPM2_NV_ReadPublic
TPM2_NV_ReadLock
TPM2_GetRCString
TPM2_FlushContext

TPM2_NV_ReadLock

Synopsis:

#include <wolftpm/tpm2.h>

TPM_RC TPM2_NV_ReadLock(NV_ReadLock_In* in);

Description:
If TPMA_NV_READ_STCLEAR is SET in an Index, then this command may be used to prevent further reads of the NV Index until the next **TPM2_Startup** (TPM_SU_CLEAR). Proper authorizations are required for this command as determined by TPMA_NV_PPREAD, TPMA_NV_OWNERREAD, TPMA_NV_AUTHREAD, and the *authPolicy* of the *nvIndex*.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to an NV_ReadLock_In struct containing an *authHandle* (TPM2_RH_NV_AUTH), and *nvIndex* (TPM2_RH_NV_INDEX)

Example:

See Also:
TPM2_NV_Read
TPM2_NV_ReadPublic
TPM2_GetRCString
TPM2_FlushContext

TPM2_NV_ChangeAuth

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_NV_ChangeAuth(NV_ChangeAuth_In* in);

Description:
This command allows the authorization secret for an NV Index to be changed. If

successful, the authorization secret (newAuth member of the **in** parameter) of the NV Index associated with nvIndex is changed.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to an NV_ChangeAuth_In struct containing the nvIndex and the new authorization secret.

Example:

See Also:
TPM2_GetRCString
TPM2_FlushContext

TPM2_NV_Certify

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_NV_Certify(NV_Certify_In* in, NV_Certify_Out* out);

Description:
The purpose of this command is to certify the contents of an NV Index or portion of an NV Index.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
in - a pointer to an NV_Certify_In struct
out - a pointer to an NV_Certify_Out struct

Example:

See Also:
TPM2_GetRCString
TPM2_FlushContext

TPM2_Cleanup

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_Cleanup(TPM2_CTX* ctx);

Description:
This function clears the global lock of the current context, and releases the locks on other resources that may have been used.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
ctx - a pointer to a TPM2_CTX struct to free/clean up the members of

Example:
```
/* Shutdown */
cmdIn.shutdown.shutdownType = TPM_SU_CLEAR;
rc = TPM2_Shutdown(&cmdIn.shutdown);
if (rc != TPM_RC_SUCCESS) {
      /* TPM2_Shutdown failed */
}

TPM2_Cleanup(&tpm2Ctx);
```

See Also:
TPM2_Shutdown
TPM2_GetRCString

TPM2_FlushContext
wolfCrypt_Cleanup

TPM2_SetSessionAuth

Synopsis:
#include <wolftpm/tpm2.h>

TPM_RC TPM2_SetSessionAuth(TPMS_AUTH_COMMAND *cmd);

Description:
This command assigns the current context's *authCmd* (TPMS_AUTH_COMMAND) to the given *cmd*.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
TPM_RC_FAILURE is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
cmd - a pointer to a TPMS_AUTH_COMMAND struct that will replace the context's current *authCmd*.

Example:
```
TPMS_AUTH_COMMAND session[MAX_SESSION_NUM];
XMEMSET(session, 0, sizeof(session));
session[0].sessionHandle = TPM_RS_PW;
TPM2_SetSessionAuth(session);
```

See Also:
TPM2_GetRCString
TPM2_FlushContext

TPM2_GetActiveCtx

Synopsis:
#include <wolftpm/tpm2.h>

TPM2_CTX* TPM2_GetActiveCtx(void);

Description:
Returns a pointer the current active TPM2 context variable

Return Values:
A **TPM2_CTX** struct that points to the current context.

Parameters:
This function takes no arguments as its parameters.

Example:
```
TPM2_CTX* ctx = TPM2_GetActiveCtx();
```

See Also:
TPM2_GetRCString
TPM2_FlushContext

TPM2_GetHashDigestSize

Synopsis:
#include <wolftpm/tpm2.h>

int TPM2_GetHashDigestSize(TPMI_ALG_HASH hashAlg);

Description:
Is able to get the digest sizes of SHA-1, SHA-256, SHA-384, and SHA-512.

Return Values:
The size of the given hashAlg parameter as an **int**.
If the given hashAlg is not a member of the list mentioned in the description, this function will return **0**.

Parameters:
hashAlg - an enumerated value that can be TPM_ALG_SHA1, TPM_ALG_SHA256, TPM_ALG_SHA384, or TPM_ALG_SHA512.

Example:

```
/* in is a pointer to a PCR_Extend_In struct */
UINT16 hashAlg = in->digests.digests[i].hashAlg;
int digestSz = TPM2_GetHashDigestSize(hashAlg);
```

See Also:
TPM2_GetRCString
TPM2_FlushContext

TPM2_GetNonce

Synopsis:
#include <wolftpm/tpm2.h>

int TPM2_GetNonce(byte* nonceBuf, int nonceSz);

Description:
This command copies *nonceSz* bytes of pseudorandom input into the *nonceBuf*, using the current context's RNG.

Return Values:
If successful the call will return **TPM_RC_SUCCESS**.
RNG_FAILURE_E is an error that may be returned.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
nonceBuf - a pointer to a byte array that will contain the nonce.
nonceSz - an integer that contains the size of *nonceBuf*

Example:
```
/* Start Auth Session */
XMEMSET(&cmdIn.authSes, 0, sizeof(cmdIn.authSes));
cmdIn.authSes.tpmKey = TPM_RH_NULL;
cmdIn.authSes.bind = TPM_RH_NULL;
cmdIn.authSes.sessionType = TPM_SE_POLICY;
cmdIn.authSes.symmetric.algorithm = TPM_ALG_NULL;
cmdIn.authSes.authHash = TPM_ALG_SHA256;
cmdIn.authSes.nonceCaller.size = WC_SHA256_DIGEST_SIZE;
rc = TPM2_GetNonce(cmdIn.authSes.nonceCaller.buffer,
                   cmdIn.authSes.nonceCaller.size);
if (rc < 0) {
    /* TPM2_GetNonce failed */
```

```
}
```

See Also:
TPM2_GetRCString
TPM2_FlushContext

TPM2_SetupPCRSel

Synopsis:
#include <wolftpm/tpm2.h>

void TPM2_SetupPCRSel(TPML_PCR_SELECTION* pcr,
TPM_ALG_ID alg,
int pcrIndex);

Description:

Return Values:
This function returns void

Parameters:
pcr - a pointer to a TPML_PCR_SELECTION struct
alg - an unsigned 16-bit integer that represents the algorithm's ID
pcrIndex - an int that contains the index of the PCR

Example:

```
pcrIndex = 0;
XMEMSET(&cmdIn.policyPCR, 0, sizeof(cmdIn.policyPCR));
cmdIn.policyPCR.policySession = sessionHandle;
cmdIn.policyPCR.pcrDigest.size = hash_len;
XMEMCPY(cmdIn.policyPCR.pcrDigest.buffer, hash, hash_len);
TPM2_SetupPCRSel(&cmdIn.policyPCR.pcrs, TPM_ALG_SHA1, pcrIndex);
```

See Also:
TPM2_GetRCString
TPM2_FlushContext

TPM2_GetRCString

Synopsis:
#include <wolftpm/tpm2.h>

const char* TPM2_GetRCString(int rc);

Description:
This function takes the return code from a function and returns a string containing an error message that explains what occurred.

Return Values:
This function returns a pointer to a character array that explains what occurred.

Parameters:
rc - the given return code that was obtained from a function

Example:
```
rc = TPM2_Init(&tpm2Ctx, TPM2_IoCb, userCtx);
if (rc != TPM_RC_SUCCESS) {
    printf("TPM2_Init failed 0x%x: %s\n", rc, TPM2_GetRCString(rc));
    goto exit;
}
```

See Also:
TPM2_Init
TPM2_Startup
TPM2_Cleanup
TPM2_FlushContext

TPM2_GetAlgName

Synopsis:
#include <wolftpm/tpm2.h>

const char* TPM2_GetAlgName(TPM_ALG_ID alg);

Description:
This function converts the ID of an algorithm into a character array.

Return Values:
This function returns a character array that contains the name of the given algorithm ID.

Parameters:
alg - the given algorithm ID from the TPM_ALG_ID enum

Example:

See Also:
TPM2_Init
TPM2_Startup
TPM2_Cleanup
TPM2_FlushContext

TPM2_GetCurveSize

Synopsis:
#include <wolftpm/tpm2.h>

int TPM2_GetCurveSize(TPM_ECC_CURVE curveID);

Description:
This functions gets the corresponding size of some curve type given a curve ID.

Return Values:
This function returns the size of the corresponding curve ID as an integer.

Parameters:
curveID - the given curve ID from the TPM_ECC_CURVE enum

Example:
```
int curveSize = TPM2_GetCurveSize(
      key->pub.publicArea.parameters.eccDetail.curveID);
if (curveSize <= 0 || *sigSz < (curveSize * 2)) {
      return BAD_FUNC_ARG;
```

```
}
```

See Also:
TPM2_Init
TPM2_Startup
TPM2_Cleanup
TPM2_FlushContext

TPM2_PrintBin

Synopsis:
#include <wolftpm/tpm2.h>

void TPM2_PrintBin(const byte* buffer, word32 length);

Description:
This function prints out the binary encoding of the given buffer with size length
- *note: this function requires debug to be enabled*

Return Values:
This function returns void

Parameters:
buffer - a generic buffer of data to print the binary of
length - the length of the **buffer** parameter

Example:
```
byte hash[WC_SHA256_DIGEST_SIZE];
int hash_len = WC_SHA256_DIGEST_SIZE;
rc = wc_Hash(WC_HASH_TYPE_SHA256, pcr, pcr_len, hash, hash_len);
if (rc < 0) {
      /* wc_Hash failed */
}
TPM2_PrintBin(hash, hash_len);
```

See Also:
TPM2_Init
TPM2_Startup
TPM2_Cleanup
TPM2_FlushContext

---

wolfTPM2_GetTpmDevId

Synopsis:
#include <wolftpm/tpm2_wrap.h>

int wolfTPM2_GetTpmDevId(WOLFTPM2_DEV* dev);

Description:
This function attempts to return the *dev*'s

Return Values:
BAD_FUNC_ARG is an error that may be returned.

Parameters:
dev - a pointer to a WOLFTPM2_DEV struct

Example:
```
int rc;
WOLFTPM2_DEV dev;
void* userCtx;
rc = wolfTPM2_Init(&dev, TPM2_IoCb, userCtx);
...
rc = TPM2_GetTpmDevId(&dev);
```

See Also:

wolfTPM2_SetAuth

Synopsis:
#include <wolftpm/tpm2_wrap.h>

int wolfTPM2_SetAuth(WOLFTPM2_DEV* dev,
                     int index,
                     TPM_HANDLE sessionHandle,
                     const byte* auth,

```
                                    int authSz);
```

Description:
This function simplifies the call to TPM2_SetSessionAuth

Return Values:
If successful, this function will return **0**.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
dev - a pointer to a WOLFTPM2_DEV struct
index - the number of the session to set (must be less than MAX_SESSION_NUM)
sessionHandle - the default session handle
auth - the default session auth
authSz - size of the default session auth

Example:

See Also:

wolfTPM2_StartSession

Synopsis:
```
#include <wolftpm/tpm2_wrap.h>

int wolfTPM2_StartSession(WOLFTPM2_DEV* dev,
                          WOLFTPM2_SESSION* session,
                          WOLFTPM2_KEY* tpmKey,
                          WOLFTPM2_HANDLE* bind,
                          TPM_SE sesType,
                          int useEncrypDecrypt);
```

Description:

Return Values:

If successful, this function will return **0**.
BAD_FUNC_ARG is an error that may be returned.

Parameters:
dev - a pointer to a WOLFTPM2_DEV struct
index - the number of the session to set (must be less than MAX_SESSION_NUM)
sessionHandle - the default session handle
auth - the default session auth
authSz - size of the default session auth

Example:

See Also: