TCG TSS 2.0 System Level API (SAPI) Specification

Version 1.1

Revision 29

August 6, 2019

Contact: admin@trustedcomputinggroup.org

PUBLISHED

TCG

Disclaimers, Notices, and License Terms

Copyright Licenses:

- Trusted Computing Group (TCG) grants to the user of the source code in this specification (the “Source Code”) a worldwide, irrevocable, nonexclusive, royalty free, copyright license to reproduce, create derivative works, distribute, display and perform the Source Code and derivative works thereof, and to grant others the rights granted herein.

- The TCG grants to the user of the other parts of the specification (other than the Source Code) the rights to reproduce, distribute, display, and perform the specification solely for the purpose of developing products based on such documents.

Source Code Distribution Conditions:

- Redistributions of Source Code must retain the above copyright licenses, this list of conditions and the following disclaimers.

- Redistributions in binary form must reproduce the above copyright licenses, this list of conditions and the following disclaimers in the documentation and/or other materials provided with the distribution.

Disclaimers:

- THE COPYRIGHT LICENSES SET FORTH ABOVE DO NOT REPRESENT ANY FORM OF LICENSE OR WAIVER, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, WITH RESPECT TO PATENT RIGHTS HELD BY TCG MEMBERS (OR OTHER THIRD PARTIES) THAT MAY BE NECESSARY TO IMPLEMENT THIS SPECIFICATION OR OTHERWISE. Contact TCG Administration (admin@trustedcomputinggroup.org) for information on specification licensing rights available through TCG membership agreements.

- THIS SPECIFICATION IS PROVIDED "AS IS" WITH NO EXPRESS OR IMPLIED WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, ACCURACY, COMPLETENESS, OR NONINFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE.

- Without limitation, TCG and its members and licensors disclaim all liability, including liability for infringement of any proprietary rights, relating to use of information in this specification and to the implementation of this specification, and TCG disclaims all liability for cost of procurement of substitute goods or services, lost profits, loss of use, loss of data or any incidental, consequential, direct, indirect, or special damages, whether under contract, tort, warranty or otherwise, arising in any way out of use or reliance upon this specification or any information herein.

Corrections and Comments

Please send comments and corrections to techquestionsadmin@trustedcomputinggroup.org.

Normative-Informative Language

“SHALL,” “SHALL NOT”, “SHOULD”, ”SHOULD NOT”, “RECOMMENDED”, “MAY” and “OPTIONAL” in this document are normative statements. They are to be interpreted as described in [RFC-2119].

Acknowledgements

TCG and the TSS Work Group would like to thank the following people for their work on this specification.

- Will Arthur Raytheon
- Brenda Baggaley OnBoard Security
- David Challener Johns Hopkins University, APL
- Michael Cox OnBoard Security
- Andreas Fuchs Fraunhofer SIT
- Ken Goldman IBM
- Jürgen Repp Fraunhofer SIT
- William Roberts Intel
- Tadeusz Struk Intel
- Philip Tricca Intel
- Lee Wilson OnBoard Security

Table of Contents

1 General Information on the TCG TSS 2.0 Specification Library

1.1 Scope of this Specification

This TCG TSS 2.0 SAPI Specification provides the lowest level programmatic interface for users creating applications for TPM 2.0. It provides the smallest footprint and uses the least resources of the three user-facing APIs of the TSS 2.0 stack (note: The three user-facing TSS 2.0 APIs are SAPI, ESAPI and FAPI). The intended audience for this specification includes software developers and designers implementing applications for TPM 2.0 and TCG TSS 2.0.

1.2 Acronyms

For definitions of the acronyms used in the TSS 2.0 specifications please see the TCG TSS 2.0 Overview and Common Structures Specification [22}.

1.3 TCG Software Stack 2.0 (TSS 2.0) Specification Library Structure

At the time of writing, the documents that are part of the specification of the TSS 2.0 are:

[1] TCG TSS 2.0 Overview and Common Structures Specification

[2] TCG TSS 2.0 TPM Command Transmission Interface (TCTI) API Specification

[3] TCG TSS 2.0 Marshaling/Unmarshaling API Specification

[4] TCG TSS 2.0 System API (SAPI) Specification

[5] TCG TSS 2.0 Enhanced System API (ESAPI) Specification

[6] TCG TSS 2.0 Feature API (FAPI) Specification

[7] TCG TSS 2.0 TAB and Resource Manager Specification

[8] TCG TSS 2.0 TAB Response Code API Specification

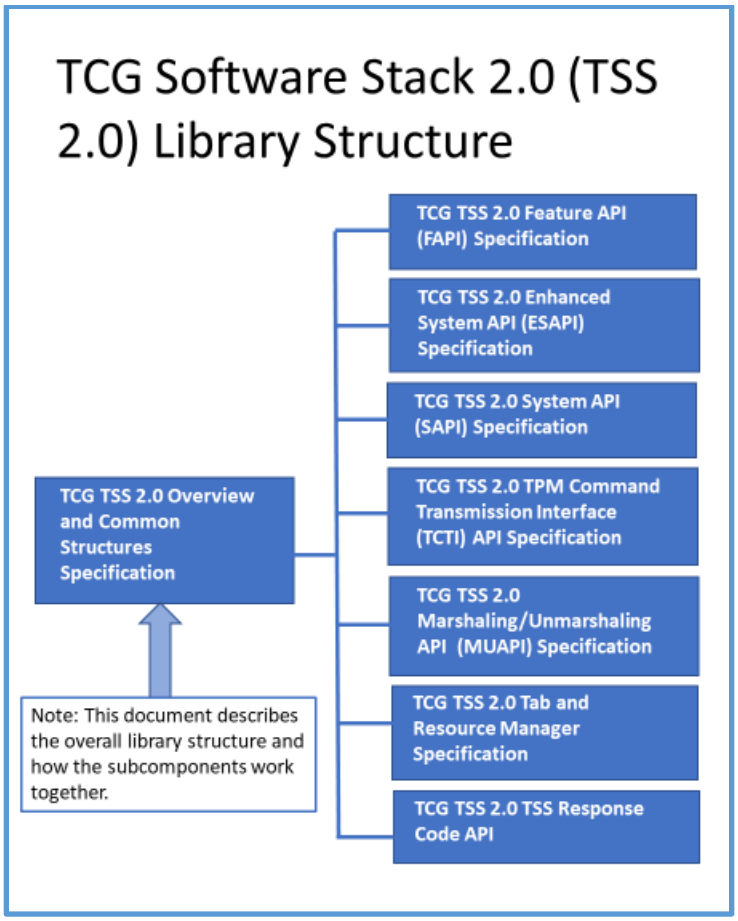

Figure 1: TSS 2.0 Specification Library

1.4 References

All references for the TSS 2.0 specifications are provided in the TCG TSS 2.0 Overview and Common Structures Specification [22].

2 TSS 2.0 SAPI Introduction

2.1 Architecture

Use of the SAPI requires expert knowledge of the underlying TPM 2.0 commands and architecture. The purpose of the SAPI is to enable applications to perform TPM2.0 specification Part 3 commands using all possible variations of inputs to those commands and receiving all possible variations of outputs. The System API may use the MU API to marshal inputs from C structures to command byte streams and unmarshal responses from response byte stream format to C structures. It uses the TCTI to communicate with the TPM.

2.2 SAPI High Level Design Requirements:

The SAPI has the following overarching (high-level) design requirements:

- Multiple TPMs shall be supported both on the same platform and on remote platforms.
- A single application can talk to multiple TPMs.
- The APIs shall neither inhibit the use of multi-thread synchronization of accesses to the TPM nor require it.
- The SAPI architecture should be able to support all operating systems as well as embedded, non-OS systems.
- SAPI implementations should support ANSI C compilers.
- SAPI implementations shall not be required to maintain state in the global or thread local scope. All state shall be maintained in their respective context structures.
- The SAPI context structure is opaque and provides functions for:
    - Getting and setting all necessary fields.
    - Returning the size of the context structures.
    - Isolating the application from changes in the size and structure of the context structures.
- The SAPI shall return system API-specific error codes in a way that differentiates it from other TSS layers' error codes.
- All SAPI data structures shall be allocated by the caller and not by the SAPI implementations.

2.2.1 Threading Model

The SAPI shall follow the same design model as a non-blocking and thread-aware library. This allows the implementation and usage of the presented APIs even in highly embedded devices and/or main loop driven applications that may not provide threading at all or with specific thread isolation mechanisms.

This means that unless otherwise stated, every function is a non-blocking operation. This specification does not give exact deadlines for function executions, but they are expected to require the typical number of cycles required by non-blocking operations.

For multi-threaded applications, this means that any SAPI context may only be called from one thread at a given time. The application may implement a means to synchronize concurrent access to the same SAPI context if desired. It is recommended to instantiate one context per thread that needs to communicate with the TPM.

If a multi-threaded application is calling in to a single SAPI context from multiple threads then the application must implement its own mutual exclusion mechanism. This approach is not recommended because it requires synchronization between threads which may be error prone.

Instead, the recommended method is to initialize multiple SAPI-contexts, one for each thread that needs access to the TPM. The concurrent access of different threads to distinct contexts is safe and may be supported by a system that includes a TAB.

3 SAPI

The System API is the layer of the overall TSS architecture that provides access to all the functionality of the TPM.

3.1 Overall functionality

SAPI is designed to be used wherever low level calls to the TPM functions are made: firmware, BIOS, applications, OS, etc. It sits at the application interface layer in a manner similar to the TSP in a TSS 1.2, but also comprises much of the functionality incorporated into the TCS in the TSS 1.2, such as the TPM parameter block generator.

The System API, as a low level interface, is targeted towards expert applications. Its purpose is to provide access to the full range of the TPM 2.0 capabilities and to do this in a way that makes the caller's job as easy as possible. It is designed to be used in a wide range of computing devices, from very low end, i.e. highly embedded, systems to high end servers.

3.2 Design requirements

The requirements for the SAPI are:

1. The SAPI SHALL provide access to all capabilities of all TPM 2.0 functions as defined in part 3 of the TPM 2.0 specification.
2. The SAPI implementation SHALL NOT require crypto and session management functionality. Its main purpose is to provide access to TPM 2.0 commands: it only supports those plus a minimal set of required management functions. All other functionality is out of scope for the SAPI.
3. The SAPI SHALL NOT unnecessarily limit any system architectures that may want to use the System API and its implementation. Some specifics:
    a. The SAPI SHALL NOT preclude systems which don't need HMACs or cpHash pre-calculations nor SHALL it burden such systems with unnecessary overhead related to HMACs or cpHash calculations.
    b. The SAPI SHALL be configurable with respect to the handling of return data required by applications. For instance, many commands return values that the caller may not want or need and some of these are code intensive to return in unmarshalled form. NULL buffer pointers passed in for particular parameters can be used by the SAPI to signal the implementation that these return values aren't needed.
4. An implementation of the SAPI SHALL support either asynchronous or synchronous calls for all TPM 2.0 commands.
5. An implementation of the SAPI MAY support asynchronous and synchronous calls to TPM 2.0 commands.
6. For each TPM 2.0 command, an implementation of the SAPI MAY support a command-specific Tss2_Sys_<COMMAND_NAME[1]>_Prepare function call that provides calling applications with the information needed to calculate a cpHash for the command. This cpHash is used by the application

[1] <COMMAND_NAME> is derived from the name of the TPM function from library specification part3 without the leading "TPM2_" (e.g. TPM2_GenRandom produces Tss2_Sys_GenRandom_Prepare)

for calculating HMAC authorizations and pre-calculating policy hashes which will be used in creating objects that will use policy authorizations.

7. The API SHALL NOT require the SAPI implementation to allocate memory for any input or output data structures. It is the calling application's responsibility to allocate any memory needed.
8. All SAPI data SHALL be in native-endian format. This means that the SAPI implementation will do any endian conversion required for both inputs and outputs.
9. The SAPI implementation SHALL perform formatting, marshalling, and unmarshalling tasks so that the caller needs as little knowledge of the inner workings of TPM 2.0 as possible. Marshalling of input and unmarshalling of output data is performed by the SAPI.
10. The SAPI implementation SHALL return all "unhandled" error codes from lower layers in the TSS stack to the SAPI caller without alteration.

NOTE: An example of a handled error is the case where the SAPI calls tss2_tcti_receive and TSS2_RC_TCTI_TRY_AGAIN is returned. In this case the SAPI will call tss2_tcti_receive again.

NOTE: An example of an unhandled error is the TPM returning TPM_RC_BAD_VALUE in response to a command and the SAPI propagating the error to the caller.

3.3 Design rules

In order to best achieve the System API requirements, the following design rules for the System API were developed:

1. SAPI functions that execute TPM 2.0 commands typically execute one TPM 2.0 command.
   NOTE: one exception to this is if underlying layers such as the resource manager do TPM 2.0 commands in order to fulfil their role.
2. As much as possible, the System API will mimic the TPM 2.0, Part 3 command schematics and Part 1 command and response layout diagrams. Function input and output parameters are ordered in the way they appear in the Part 3 command schematics and variable names match as much as possible.
   NOTE: This will help programmers understand the code and easily correlate it to the specification.
3. Since memory for input and output parameters is provided by the caller, some design rules result:
   a. All output parameters will require a pointer to be passed to the SAPI.
   b. In order to minimize the stack memory requirements, inputs that are not simple data types or bit fields will be passed in as pointers.
   c. Buffers for the input command byte stream, output response byte stream, cpBuffer, and rpBuffer are allocated by the caller as part of the context structure to minimize use of function stack space.
4. The System API implementation will do as much work for the caller as possible. Some examples of this:
   a. The commandSize field for all commands is calculated dynamically by the SAPI implementation.
   b. Output parameters will be unmarshalled into C structures before being returned to the caller so that the caller can read fields out of them in a straightforward manner.
5. The only callbacks used by the SAPI code are to TCTI functions. No callbacks are used to call from the System API implementation to "helper" functions to calculate cpHash or HMAC or to perform any other crypto or session management tasks. Instead a layer above the SAPI explicitly makes those calls. This provides the most flexibility possible to the ESAPI or whatever other

layer is directly above the SAPI. For instance, the caller may want to use different HMAC helper functions depending on what actions are being performed.

3.4 Architecture

1. Each SAPI function that corresponds to a Part 3 command takes the following inputs:
    a. A pointer to a SAPI context structure that is used to maintain any state required. This structure is allocated by the caller.
    b. A group of command and/or response parameters to the TPM:
        i. Inputs:
            1. Input parameters are in “TSS System API” form to make the caller’s job easier. This means that C structures will be used as inputs.
            2. All inputs to the TPM that are data structures are input as pointers to those data structures in the SAPI.
        ii. Outputs:
            1. A group of pointers to buffers, one for each possible output data item or structure. If any output pointer is NULL, the output is not required by the caller and the implementation will not do any work to provide that output to the caller.
    c. All commands that aren’t restricted to the TPM_ST_NO_SESSIONS command. These commands will be capable of handling between 0 and 3 sessions or authorizations on input and output. Sessions and authorizations are input through a data structure that:
        i. Identifies the number of sessions or authorizations that are in use
        ii. Contains all the data for each session/authorization.
2. The SAPI implementation marshals the input parameters before sending them to the TPM. This includes the following:
    a. Endian conversion if necessary.
    b. Population of tag, command size (computed by the implementation after marshalling is complete), command code, handles, authorization block size, marshalled authorizations, and command parameters.
3. To send the data to the next lower layer, the SAPI implementation SHOULD invoke the tss2_tcti_transmit macro, passing it the tctiContext member of the SAPI context structure.
4. To receive the response, the SAPI implementation SHOULD invoke the tss2_tcti_receive macro, passing it the tctiContext member of the SAPI context structure.
5. The System API implementation checks for errors in the transmission to or reception of the data from the TPM. Handling of these errors is the responsibility of the caller.
6. If an error has occurred, the System API implementation returns this error code to the application. No more processing of the returned data occurs in this case.
7. If the TPM command completed successfully, for all outputs that received a non-null output pointer, the System API implementation:
    a. Unmarshals the output into a C structure, which includes converting the endianness when necessary.
    b. Copies the data to the structure pointed to by the output parameter pointer.
8. The SAPI implementation unmarshals the response authorization areas into C structures.
9. TSS SAPI assumes that all required initialization is done before any SAPI functions are called. Specifically:

a. Underlying TAB instantiations, resource manager(s), and device driver interfaces are initialized.
b. Applications that call into the SAPI know which TPMs are available and tell the SAPI how to communicate to the particular TPMs that are used by means of the TCTI context structures.
c. Figure 3 below is a drawing of this. Purple outlined blocks and lines are done at initialization time. The definition of initialization time is implementation specific and out of scope for this specification.
d. The sequence is as follows:
    i. Some system process initializes the TSS stack, TAB instantiation(s), including resource manager(s), and TPM 2.0 driver(s). How and when this is done is implementation-specific with the only requirement being that this must be done before SAPI functions are called.
    ii. Initialization of the TCTI context structure(s) is implementation-specific with the only requirement being that this must be done before SAPI functions are called.
    iii. Application(s) start.
    iv. Applications call SAPI functions, initializing the cmdContext structures; part of this initialization includes setting a pointer to point to the TCTI structure which contains the correct send/receive function pointers and interface name.

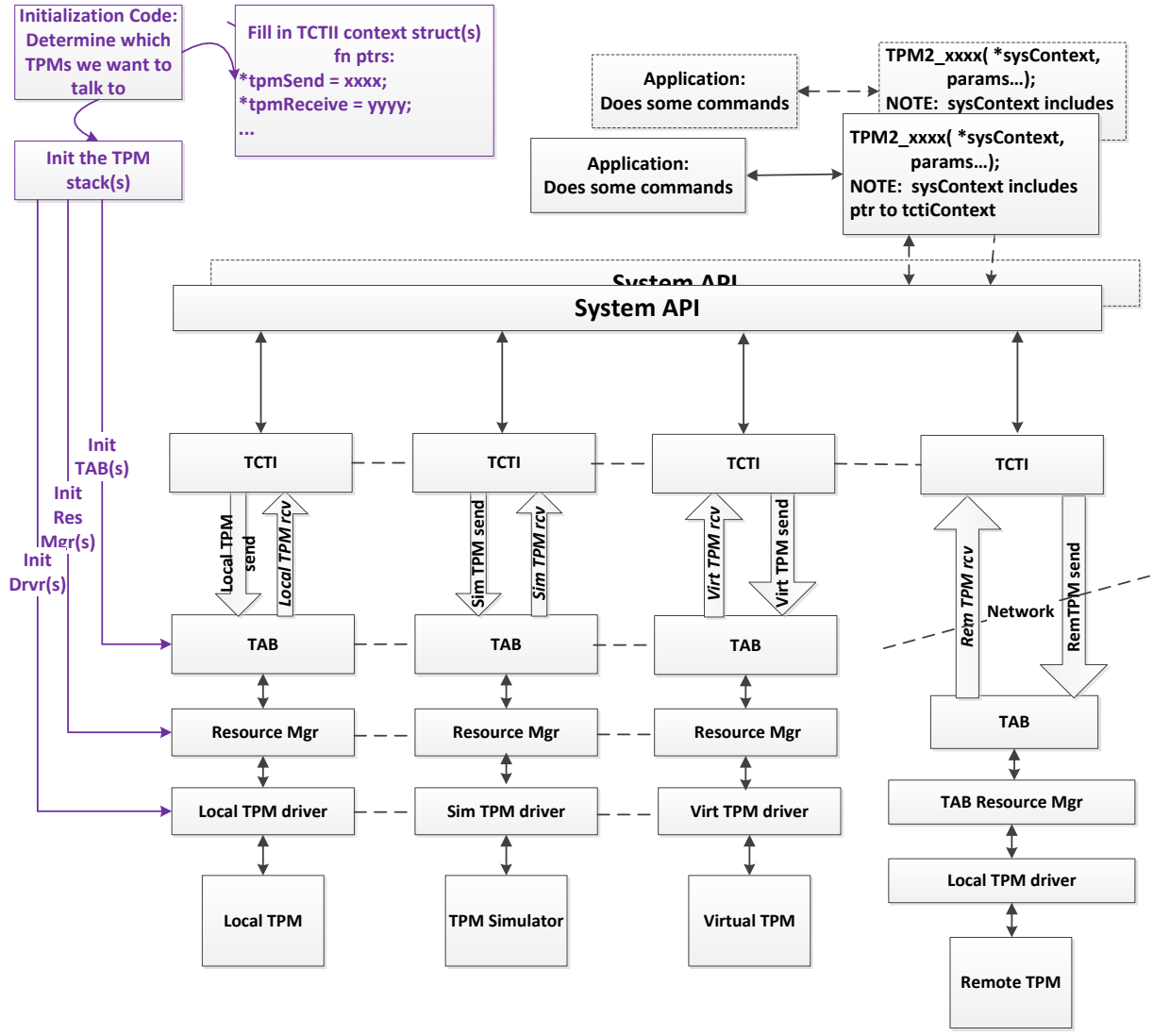

Figure 2 — Low level stack details

3.5 SAPI data structures

All SAPI data structures are defined in the tss2_sys.h header file. The contents of tss2_sys.h are specified in the “TSS 2.0 Header File Specification”.

3.6 Application Binary Interface (ABI) Negotiation

The wide applicability of this specification and the library-nature of the TPM specification make it very likely that multiple variations of the TSS APIs will exist. In order to be able to differentiate them at runtime, an ABI negotiation shall take place inside each “initialize” call. The TCTI layer provides ABI negotiation through a version field. The following ABI negotiation scheme is used by the SAPI and higher layers of the TSS.

The TSS2_ABI_VERSION structure is defined to hold the relevant information about the ABI of a specific TSS implementation. A `tssCreator` value of 0x1 identifies the structure sizes defined by the TSS WG

believed to be generally appropriate for “typical” systems. If the value for `tssCreator` is 1 then the data definitions of the TSS 2.0 header file specification must be used.

For vendor-specific implementations that use different structure sizes, the `tssCreator` field must be set to a value larger than 0x20000000 and hold a TPM capabilities vendor ID (CAP_VID) from the TCG Vendor ID Registry. The family, level and version field can be used by the vendor for their own purposes.

```
typedef struct {
        UINT32 tssCreator; /* If == 1, this equals TSSWG-Interop */
                    /* IF > 0x20000000 , this equals TCG TPM capabilities Vendor-ID */
        UINT32 tssFamily;  /* Free-to-use for creator > TCG_VENDOR_FIRST */
        UINT32 tssLevel;   /* Free-to-use for creator > TCG_VENDOR_FIRST */
        UINT32 tssVersion; /* Free-to-use for creator > TCG_VENDOR_FIRST */
} TSS2_ABI_VERSION;
```

For each layer that uses ABI negotiation, a function will validate that the ABI version requested by the layer is supported by the layer below it. For the SAPI, the function that does this is Tss2_Sys_Initialize. If the requested ABI-Version is not supported it will return TSS2_SYS_RC_ABI_MISMATCH and set the fields to the values that the implementation supports.

Note: A module may support multiple versions of ABI at the same time.

A typical set of TSS header files for a layer should include a definition of the ABI version used throughout the layer.

```
#define TSS2_ABI_CURRENT_VERSION {.tssCreator = 0xXX, .tssFamily = 0xXXXX, tssLevel = xx,
tssVersion = xx }
```

3.7 Command Parameters

The parameters for Part 3 command-specific functions, _Prepare, _Complete, and one-call, are the C structures as specified in the Part 2 Data Types section of this specification.

In order to guarantee binary compatibility between applications and SAPI implementations built with different compiler tool chains, wherever the TPM specification Part 2 specifies a bit field, the API expects a 32 bit unsigned integer where bit 0 is the least significant bit.

3.7.1 System API Parameter Rules

All parameters in authorization areas and Part 3 specific functions will be passed in as:

1. Simple parameters if they are command inputs or fields in the authorization regions AND they are one of the following base types:
    - Types that evaluate to UINT8, UINT16, UINT32 or UINT64 values and their signed equivalents.
    - Types that alias to base types e.g. TPM_HANDLE or TPMA_SESSION.
2. Pointers to the parameter types, if they are anything other than base types OR if they are response parameters.

3.8 SAPI Function APIs (by Category)

Within the SAPI functions, there are two types of functions:

- TPM 2.0 Part 3 command specific functions

- Generic functions which are not specific to particular Part 3 commands

The SAPI function APIs are split into the following groups, each of which may be Part 3 specific or not:

- Command Context Allocation Functions: all non-specific.
- Command Context Setup Functions: all non-specific.
- Command Preparation Functions: Tss2_Sys_<COMMAND_NAME>_Prepare calls are Part 3 specific, all others are not specific (see Section 3.8.2.2).
- Command Execution Functions: Tss2_Sys_<COMMAND_NAME><COMMAND_NAME> "one-call" functions are Part 3 specific, all others are not specific (see Section 3.8.3.4).
- Command Completion Functions: Tss2_Sys_<COMMAND_NAME>_Complete functions are Part 3 specific, all others are not specific (see Section 3.8.4.3).

The function prototypes for all of these functions are in the tss2_sys.h header file. The contents of tss2_sys.h are specified in the "TSS 2.0 Header File Specification".

3.8.1 Command Context Allocation Functions

3.8.1.1 Tss2_Sys_GetContextSize

```
size_t  Tss2_Sys_GetContextSize(
      size_t                        maxCommandResponseSize
);
```

This function returns the required size for the opaque SAPI command context. The caller must allocate a context of at least this size.

If *maxCommandResponseSize* is 0, Tss2_Sys_GetContextSize returns a size guaranteed to handle any TPM command and response supported by this specification. The caller may specify a non-zero *maxCommandResponseSize* corresponding to the maximum expected command and response size in order to save memory within the TSS2_SYS_CONTEXT SAPI context.

NOTE: The returned value will be larger than maxCommandResponseSize. How much larger depends on the System API implementation. For instance, if an implementation uses two independent buffers for transmit and receive, then the size returned could be twice the passed in size plus whatever extra memory is needed by the System API for context.

NOTE: If the caller constrains the size, a subsequent command may then return TSS2_SYS_RC_INSUFFICIENT_CONTEXT if the actual size of a command or response exceeds the size specified in this function. For an error while forming the command, the caller may be able to start over with a larger context. For a response, error handling is more complicated because the response may be in transit. It is therefore recommended to use maxCommandResponseSize set to 0 for context allocation.

3.8.1.2 Tss2_Sys_Initialize

```
TSS2_RC Tss2_Sys_Initialize(
      TSS2_SYS_CONTEXT        *sysContext,
      size_t                        contextSize,
      TSS2_TCTI_CONTEXT             *tctiContext,
```

```
      TSS2_ABI_VERSION        *abiVersion
);
```

This function must be called once in a newly allocated *sysContext*. It need not be called to reuse a sysContext. The SAPI may perform getCapability calls or anything else required for enabling workarounds of non-compliant, unspecified or erroneous TPM behaviors. These SHOULD be done during the Tss2_Sys_Initialize call and these SHOULD NOT be done at other times.

The *contextSize* parameter specifies the size of the memory area reserved for the *sysContext*. It is expected to be the same size returned from a corresponding tss2_sys_getContextSize call (see above).

The *tctiContext* parameter holds the pointer to an initialized TCTI Context that will be used by subsequent calls to communicate with the TPM. tctiContext can be retrieved later using tss2_sys_getTctiContext but cannot be altered during the lifetime of one systemAPI context. The Tss2_Sys_Initialize function SHOULD check that the essential TCTI function pointers (transmit and receive) are not NULL and return an error code otherwise.

The *abiVersion* parameter holds information about the version and revision of this specification as used by the application. The SAPI implementation will check whether it is compatible with this version. If not, SAPI will return an error and override abiVersion with the ABI version it supports. The corresponding variable that is passed into abiVersion can be initialized via the TSS2_ABI_VERSION_CURRENT definition (see the TCTI Context section).

Response Codes:

- TSS2_SYS_RC_ABI_MISMATCH: if input ABI doesn't match that of the SAPI implementation.
- TSS2_SYS_RC_INSUFFICIENT_CONTEXT: if the provided context structure is too small for the SAPI function.
- TSS2_SYS_RC_BAD_VALUE: Returned if a bad value for any parameter is detected
- TSS2_SYS_RC_BAD_REFERENCE: if any of *sysContext*, t*ctiContext*, or *abiVersion* are NULL pointers.
- TSS2_SYS_RC_BAD_TCTI_STRUCTURE: if the implementation checks the TCTI function pointers and any of the essential ones (transmit and receive) are set to NULL, or bad magic or other malformed part of the structure
- Any TPM or TCTI errors that could result from GetCapability calls that are made to get TPM version info.
- TSS2_SYS_RC_INCOMPATIBLE_TCTI: unknown or unusable TCTI version.
- TSS2_RCs produced by lower layers of the software stack SHALL be returned to the caller unaltered unless handled internally.

3.8.1.3 Tss2_Sys_Finalize

```
void Tss2_Sys_Finalize(
      TSS2_SYS_CONTEXT        *sysContext
);
```

This function should be called before freeing a sysContext.

3.8.2 Command Preparation Functions

3.8.2.1 Tss2_Sys_GetTctiContext

```
TSS2_RC Tss2_Sys_GetTctiContext(
      TSS2_SYS_CONTEXT        *sysContext,
      TSS2_TCTI_CONTEXT             **tctiContext
);
```

This function returns the pointer to the tctiContext that was passed in during the Tss2_Sys_Initialize call. This function can, for example, be used to retrieve the tctiContext associated with a given sysContext before freeing the sysContext in order to either reuse or free the associated tctiContext.

Response Codes:

TSS2_SYS_RC_BAD_REFERENCE: if *sysContext* or *tctiContext* are NULL pointers.

3.8.2.2 Tss2_Sys_<COMMAND_NAME>_Prepare

To construct a SAPI command for each of the TPM 2.0 Specification Part 3 commands, replace the <COMMAND_NAME> with the appropriate TPM 2.0 Specification Part 3 command with the TPM2_ prefix removed (e.g. Tss2_Sys_GenRandom_Prepare).

```
TSS2_RC Tss2_Sys_<COMMAND_NAME>_Prepare(
      TSS2_SYS_CONTEXT        *sysContext,
                              inHandles,
                              inParams
);
```

This is a template for a _Prepare function. There is one of these per Part 3 TPM command. The number and types of input parameters and handles are defined in Part 3 of the TPM specification.

All passed in information is copied and saved in the *sysContext* for use during subsequent commands.

Since the command doesn't yet know whether authorizations will be sent or not, this function sets the tag to TPM_ST_NO_SESSIONS. Later, if Tss2_Sys_SetCmdAuths is called, the tag will be set to TPM_ST_SESSIONS if sessions are added.

If any command parameter (after the handles) is a TPM2B:

- If the TPM2B parameter is NULL, the implementation marshals a TPM2B with a size of 0.

  NOTE: The TPM often uses this pattern of setting the TPM2B size field to 0 for optional parameters.

- If the TPM2B parameter is not NULL:
  - If the TPM2B is a simple TPM2B (that is, a TPM2B which contains only a size and a byte array, such as TPM2B_DIGEST), the TPM2B size field indicates the used size of the UINT8 array. The implementation marshals the size and buffer into the byte stream. The implementation will not read beyond the used size.

NOTE: The size may be zero.

NOTE: For the first command parameter, the UINT8 array may be encrypted before or after _Prepare. To encrypt after _Prepare, see Tss2_Sys_GetDecryptParam.

If the TPM2B is a complex TPM2B (that is, a TPM2B that is not simple, such as a TPM2B_PUBLIC which contains a size and a TPMT_PUBLIC), the TPM2B size field will be ignored.

NOTES:
1. The implementation will calculate the size and marshal the TPM2B second parameter based on its data type.
2. Ignoring the size field permits the application to use a response parameter unmodified as an input to a subsequent command. Since the size is ignored, if the caller wants a complex TPM2B to be marshaled with a size of 0, it should be passed in as NULL.
3. For the first command parameter that is a TPM2B, the following also apply:
- If the first command parameter is non-NULL, the implementation will marshal it. After _Prepare, the parameter may be encrypted. See Tss2_Sys_GetDecryptParam.
- If the first command parameter is NULL, then a zero sized TPM2B is inserted for that command parameter. The caller may (and normally should) add the parameter after the _Prepare. See Tss2_Sys_SetDecryptParam.

The _Prepare command resets the *sysContext* and makes it ready for the next flow of command functions. After any _Prepare, previously set authorizations become unavailable and Tss2_Sys_SetCmdAuths must be called again.

NOTE: As an example of why it is required that _Prepare reset the sysContext, suppose the following sequence occurs: _Prepare, Tss2_Sys_SetCmdAuths, _Prepare . If the second _Prepare is done with different parameters than the first _Prepare, the authorizations previously set by Tss2_Sys_SetCmdAuths are no longer valid after the second _Prepare. To handle this as cleanly as possible and avoid burdening the implementation with the overhead of checking the parameters for changes, after any _Prepare (whether the parameters have changed or not), previously set authorizations are made unavailable and Tss2_Sys_SetCmdAuths must be called again.

Response Codes:

- TSS2_SYS_RC_INSUFFICIENT_CONTEXT: if the provided context structure is too small for the SAPI function.
- TSS2_SYS_RC_BAD_VALUE: Returned, if bad values for parameters are detected,
- TSS2_SYS_RC_BAD_REFERENCE: if sysContext or a parameter that is a pointer and is not a TPM2B is set to NULL.
- TSS2_SYS_RC_BAD_SEQUENCE: if called between Tss2_Sys_ExecuteAsync and Tss2_Sys_ExecuteFinish.
- TSS2_RCs produced by lower layers of the software stack SHALL be returned to the caller unaltered unless handled internally.

3.8.2.3 Tss2_Sys_GetDecryptParam

```
TSS2_RC Tss2_Sys_GetDecryptParam(
      TSS2_SYS_CONTEXT        *sysContext,
      size_t                        *decryptParamSize,
      const uint8_t                 **decryptParamBuffer
```

```
);
```

This function returns the size and pointer to a buffer corresponding to the first marshaled TPM2B command parameter as *decryptParamSize* and *decryptParamBuffer*. If the first command parameter passed to _Prepare was NULL, *decryptParamSize* is 0 and *decryptParamBuffer* is unspecified. For consistency with the TPM terminology a TPM input parameter that is encrypted by the application and decrypted by the TPM is referred to as a *decrypt* parameter and the session controlling the encryption parameters is referred to as the decrypt session and must have its TPMA_SESSION_DECRYPT bit set.

This function must be called after _Prepare and before the Tss2_Sys_Execute or Tss2_Sys_ExecuteAsync.

The application must not write to the returned *decryptParamBuffer* and this buffer may only be considered readable until the next invocation of any function that uses the same sysContext. If any other SAPI call is made to the same *sysContext* a previously retrieved *decryptParamBuffer* contains undefined data and Tss2_Sys_GetDecryptParam must be called again.

If this function is called after a Tss2_Sys_SetDecryptParam, the newly set parameter is returned.

The intent of this call is to provide the size and location of the parameter to be encrypted by the caller in a decrypt session after the parameter has been marshaled by _Prepare. After calling this, the encrypted result can be set by calling Tss2_Sys_SetDecryptParam.

NOTE: If encryption is performed, it must be performed after _Prepare and before Tss2_Sys_Execute or Tss2_Sys _ExecuteAync. It should typically be called before _GetCpBuffer is called for the cpHash calculation, since the cpHash must be calculated using encrypted version of this parameter.

Response Codes:

TSS2_SYS_RC_BAD_SEQUENCE: if not called after _Prepare and before Tss2_Sys_Execute/Tss2_Sys_ExecuteAsync.

TSS2_SYS_RC_NO_DECRYPT_PARAM: if called when *sysContext* is set for a command that doesn't have a decrypt parameter.

TSS2_SYS_RC_BAD_REFERENCE: if any of the inputs is a NULL pointer.

3.8.2.4 Tss2_Sys_SetDecryptParam

```
TSS2_RC Tss2_Sys_SetDecryptParam(
      TSS2_SYS_CONTEXT        *sysContext,
      size_t                        decryptParamSize,
      const uint8_t                 *decryptParamBuffer
);
```

If the first command parameter is a TPM2B type, this function sets the size and buffer of the TPM2B. If the first parameter is not a TPM2B, this function returns an error.

If _Prepare received a non-NULL first command parameter TPM2B, this function replaces the TPM2B buffer and the TPM2B size field must match *decryptParamSize*. If the _Prepare received a NULL first command parameter, this function updates the size and inserts the TPM2B buffer.

This function must be called after _Prepare and before Tss2_Sys_Execute or Tss2_Sys_ExecuteAsync.

NOTE: Tss2_Sys_SetDecryptParam should typically be called before Tss2_Sys_GetCpBuffer is called in preparation for cpHash calculation, since the cpHash calculation has to be done using the encrypted version of this parameter.

NOTE: The intent of this call is to set the first command TPM2B parameter when the caller has marshaled and encrypted the parameter.

Response Codes:

- TSS2_SYS_RC_BAD_SEQUENCE: if not called after _Prepare and before Tss2_Sys_Execute/Tss2_Sys_ExecuteAsync.
- TSS2_SYS_RC_INSUFFICIENT_CONTEXT: if the provided context structure is too small for the SAPI function.
- TSS2_SYS_RC_BAD_REFERENCE: if *sysContext* or *decryptParamBuffer* is null.
- TSS2_SYS_RC_BAD_SIZE: if the first TPM2B parameter was not NULL at _Prepare and *decryptParamSize* does not equal the marshaled size.
- TSS2_SYS_RC_NO_DECRYPT_PARAM: if called when *sysContext* is set for a command that doesn't have a decrypt parameter.
- TSS2_RCs produced by lower layers of the software stack SHALL be returned to the caller unaltered unless handled internally.

3.8.2.5 Tss2_Sys_GetCpBuffer

```
TSS2_RC Tss2_Sys_GetCpBuffer(
      TSS2_SYS_CONTEXT        *sysContext,
      size_t                        *cpBufferUsedSize,
      const uint8_t                 **cpBuffer
);
```

This function returns the *cpBuffer*, a pointer to the marshaled command parameters, and the number of used bytes in the cpBuffer.

This function can only be called after a _Prepare and before a Tss2_Sys_Execute, Tss2_Sys_ExecuteAsync, or one call function.

The application must not write to the returned *cpBuffer* and this buffer may only be considered readable until the next invocation of any function that uses the same *sysContext*. If any other SAPI call is made to the same *sysContext* a previously retrieved cpBuffer will contain undefined data and Tss2_Sys_GetCpBuffer must be called again.

NOTE: Tss2_Sys_GetCpBuffer is typically used for calculating the cpHash value for command authorization. It is typically called after an optional Tss2_Sys_SetDecryptParam call (in conjunction with encryption) and before a Tss2_Sys_SetCmdAuthorization call.

Response Codes:

- TSS2_SYS_RC_BAD_SEQUENCE: if not called after _Prepare and before Tss2_Sys_Execute/Tss2_Sys_ExecuteAsync.
- TSS2_SYS_RC_BAD_REFERENCE: if any NULL pointer is passed in.

3.8.2.6 Tss2_Sys_SetCmdAuths

```
TSS2_RC Tss2_Sys_SetCmdAuths(
      TSS2_SYS_CONTEXT        *sysContext,
      const TSS2L_SYS_AUTH_COMMAND *cmdAuthsArray
);
```

This function copies and saves the authorization data to the sysContext.

cmdAuthsArray->count indicates the number of authorizations to add. The count must be between 1 and TSS2_SYS_MAX_SESSIONS, inclusive.

If this command causes authorizations to be added, the command tag will be set to TPM_ST_SESSIONS.

NOTE: This command is optional for Part 3 commands that don't require any authorization sessions. If it is not called, the command tag defaults to TPM_ST_NO_SESSIONS.

Response Codes:

- TSS2_SYS_RC_BAD_SEQUENCE: if not called after Tss2_Sys_<COMMAND_NAME>_Prepare and before Tss2_Sys_Execute/Tss2_Sys_ExecuteAsync.
- TSS2_SYS_RC_INSUFFICIENT_CONTEXT: if the provided context structure is too small for the SAPI function.
- TSS2_SYS_RC_BAD_REFERENCE: if sysContext or cmdAuthsArray are NULL
- TSS2_SYS_RC_BAD_SIZE: if cmdAuthsArray->count is 0 or larger than TSS2_SYS_MAX_SESSIONS..
- TSS2_SYS_RC_INVALID_SESSIONS: Returned if the command requires a different number of authorizations.
- TSS2_RCs produced by lower layers of the software stack SHALL be returned to the caller unaltered unless handled internally.

3.8.3 Command Execution Functions

3.8.3.1 Tss2_Sys_ExecuteAsync

```
TSS2_RC Tss2_Sys_ExecuteAsync(
      TSS2_SYS_CONTEXT        *sysContext
);
```

This function calls the TCTI transmit callback to send the TPM command stream. It does not call the receive function. This function is called when all the necessary command data has been set via _Prepare and optionally Tss2_Sys_SetDecryptParam and Tss2_Sys_SetCmdAuths.

This function is blocking but it is expected to return quickly.

This function can only be called once after a _Prepare if the TPM command succeeds. If the TPM command does not succeed and Tss2_Sys_ExecuteFinish returns a TPM error this function can be called again to resend the same command buffer. After this call, only the Tss2_Sys_ExecuteFinish and Tss2_Sys_GetTctiContext functions can be called on the same *sysContext*.

Response Codes:

- TSS2_SYS_RC_INVALID_SESSIONS: Returned if the command requires a different number of authorizations.
- TSS2_SYS_RC_INSUFFICIENT_CONTEXT: if the provided context structure is too small for the SAPI function.
- TSS2_SYS_RC_BAD_SEQUENCE: if called before _Prepare or if, after the most recent _Prepare, one of the following functions has been called: executeAsync, Execute, one call, or Tss2_Sys_ExecuteFinish

- TSS2_SYS_RC_BAD_REFERENCE: if *sysContext* is a NULL pointer.
- TSS2_RCs produced by lower layers of the software stack SHALL be returned to the caller unaltered unless handled internally.

3.8.3.2 Tss2_Sys_ExecuteFinish

```
Tss2_Sys_ExecuteFinish(
      TSS2_SYS_CONTEXT        *sysContext,
int32_t                 timeout
);
```

This function calls the receive callback to receive the response stream.

This function can only be called after a Tss2_Sys_ExecuteAsync. Additionally, it can be called repeatedly as long as SAPI responds to it with a return code of TSS2_TCTI_RC_TRY_AGAIN. When Tss2_Sys_ExecuteFinish returns anything other than TSS2_TCTI_RC_TRY_AGAIN, subsequent invocations of the function using the same context MUST return TSS2_SYS_RC_BAD_SEQUENCE until the context is used to successfully issue another TPM command (via Tss2_Sys_ExecuteAsync).

If the timeout (in milliseconds) is:

- positive: return after the timeout, indicating whether the response was received
- 0: return immediately, indicating whether the response was received
- -1: return after the response is received

If this command returns success the response buffer can be manipulated using the Tss2_Sys_Get/SetEncryptParam functions, and the contents retrieved using Tss2_Sys_GetRspAuths and the appropriate _Finish function. If this command returns a TPM error the Tss2_Sys_ExecuteAsync or Tss2_Sys_Execute functions can be used to reissue the command. If this command returns TSS2_TCTI_RC_TRY_AGAIN a timeout occurred and the caller should call this function again later to get the TPM result. On any other error the command is finished and the next call should be to a _Prepare function.

Response Codes:

- TSS2_SYS_RC_INSUFFICIENT_CONTEXT: if the provided context structure is too small for the SAPI function.
- TSS2_TCTI_RC_TRY_AGAIN: if timeout occurs.
- TSS2_SYS_RC_INSUFFICIENT_RESPONSE: if the response does not contain at least a tag, response size, and response code.
- TSS2_SYS_RC_MALFORMED_RESPONSE: if any kind of malformed response is detected.
- TSS2_SYS_RC_BAD_REFERENCE: if *sysContext* is a NULL pointer.
- TSS2_SYS_RC_BAD_SEQUENCE: if not called immediately after Tss2_Sys_ExecuteAsync. Exception: can be called again if TSS2_TCTI_RC_TRY_AGAIN was received
- TSS2_RCs produced by lower layers of the software stack SHALL be returned to the caller unaltered unless handled internally.

3.8.3.3 Tss2_Sys_Execute

```
TSS2_RC Tss2_Sys_Execute(
      TSS2_SYS_CONTEXT        *sysContext
);
```

This function is equivalent to Tss2_Sys_ExecuteAsync followed by Tss2_Sys_ExecuteFinish with a timeout of -1.

Response Codes:

- TSS2_SYS_RC_BAD_SEQUENCE: Return this anytime there is not a command prepared to be sent and Tss2_Sys_Execute is called. Additionally, if a _Prepare command has been done and a _Async command is called prior to calling Tss2_Sys_Execute, this response code should be returned.
- All error codes that can be returned by Tss2_Sys_ExecuteAsync and Tss2_Sys_ExecuteFinish excluding TSS2_TCTI_RC_TRY_AGAIN and all cases of TSS2_SYS_RC_BAD_SEQUENCE in the Tss2_Sys_ExecuteAsync and Tss2_Sys_ExecuteFinish functions.

3.8.3.4 Tss2_Sys_<COMMAND_NAME>

To construct a SAPI command for each of the TPM 2.0 Specification Part 3 commands, replace the <COMMAND_NAME> with the appropriate TPM 2.0 Specification Part 3 command with the TPM2_ prefix removed (e.g. Tss2_Sys_GenRandom)

```
TSS2_RC Tss2_Sys_<COMMAND_NAME>(
      TSS2_SYS_CONTEXT        *sysContext,
                              inHandles,
      const TSS2L_SYS_AUTH_COMMAND  *cmdAuthsArray,
                              inParams,
                              *outHandles,
                              *outParams,
      TSS2L_SYS_AUTH_RESPONSE *rspAuthsArray
);
```

This function is a template for a “one call” function. The number and types of the input handles and parameters, the number and types of the pointers to output handles and parameters, and the presence or absence of the cmdAuthsArray and rspAuthsArray parameters is defined by the commands as described in Part 3 of the TPM specification.

There is one of these one call (or one shot) functions per TPM command. These one call (or one shot) functions can be used irrespective of authorization for:

1. Sending a command that never takes authorizations.
2. Sending optional audit sessions with a command.
3. Sending a command with simple password authorizations.
4. Sending a command with HMAC authorization.

Note use of sessions with the encrypt and/or decrypt flags is not allowed through the one call functions. Parameter encryption is only supported through the longer sequence of commands starting with a Tss2_Sys_<COMMAND_NAME>_Prepare function and ending with a Tss2_Sys_<COMMAND_NAME>_Complete function.

If any command parameter (after the handles) is a TPM2B:

- If the TPM2B parameter is NULL, the implementation marshals a TPM2B with a size of 0.

  NOTE: The TPM often uses this pattern of setting the TPM2B size field to 0 for optional parameters.

- If the TPM2B parameter is not NULL:
  - If the TPB2B is a simple TPM2B, the TPM2B size field indicates the size of the UINT8 array. The implementation marshals the size and buffer into the byte stream. The implementation will not read beyond the used size.

    NOTE: The TPM2B size may be zero.

  - If the TPM2B is a complex TPM2B, the TPM2B size field will be ignored.

    NOTE: The implementation will calculate the TPM2B size and marshal the TPM2B second parameter based on its data type.

If any response parameter is a TPM2B:

- If the response parameter is a simple TPM2B:
  - On the call to the one call function, its size parameter must be either the caller allocated size of the array or 0 to denote the default allocation size for this type.
  - Before returning from the one call function, the implementation will write the used size of the array.

  The used size is unmarshalled from the response stream. If the used size is greater than the caller allocated size, this function returns TSS2_SYS_RC_INSUFFICIENT_BUFFER.

  NOTE: If the caller reuses the TPM2B, the size must be set back to the caller allocated before the next call to _Complete or the one call function.

- If any parameter is a complex TPM2B:
  - In the call to the one call function, its size parameter MUST be zero.
  - Before returning from the one call function, its size parameter will be the unmarshalled version of the size of the TPM2B's UINT8 array as returned from the TPM.

Response Codes:

- TSS2_SYS_RC_INVALID_SESSIONS: if command cannot take or return the number of authorizations specified by cmdAuthsArray->count.
- TSS2_SYS_RC_INSUFFICIENT_BUFFER: if any of the simple TPM2B output parameters do not provide enough buffer space.
- TSS2_SYS_RC_INSUFFICIENT_CONTEXT: if the provided context structure is too small for the SAPI function.
- TSS2_SYS_RC_INSUFFICIENT_RESPONSE: if the response does not contain at least a tag, response size, and response code.
- TSS2_SYS_RC_MALFORMED_RESPONSE: if any kind of malformed response is detected
- TSS2_SYS_RC_BAD_SEQUENCE: if called between Tss2_Sys_ExecuteAsync and Finish.

- TSS2_SYS_RC_BAD_REFERENCE: if sysContext is NULL or one of input parameters that is not a TPM2B is a NULL pointer.
- TSS2_SYS_RC_BAD_VALUE: SHALL be returned if bad values for parameters are detected.
- TSS2_SYS_RC_NO_DECRYPT_PARAM: if any session has its TPMA_OBJECT_DECRYPT bit set.
- TSS2_SYS_RC_NO_ENCRYPT_PARAM: if any session has its TPMA_OBJECT_ENCRYPT bit set.
- TSS2_RCs produced by lower layers (e.g. the TPM, TCTI etc) of the software stack SHALL be returned to the caller unaltered unless handled internally.

3.8.4 Command Completion

3.8.4.1 Tss2_Sys_GetCommandCode

```
TSS2_RC Tss2_Sys_GetCommandCode(
      TSS2_SYS_CONTEXT        *sysContext,
      UINT8                   *commandCode
);
```

This function gets the command code for the command. The command code is returned as an array of bytes in big endian order. This array can be used for calculating the cpHash for a command or rpHash for a response.

The command code returned is valid from one _Prepare or one one-call function call to the next _Prepare or one-call function call.

Response Codes:

- TSS2_SYS_RC_BAD_SEQUENCE: if called before first _Prepare or first one-call function is called on a given sysContext.
- TSS2_SYS_RC_BAD_REFERENCE: If *sysContext* or *commandCode* are NULL.

3.8.4.2 Tss2_Sys_GetRspAuths

```
TSS2_RC Tss2_Sys_GetRspAuths(
      TSS2_SYS_CONTEXT        *sysContext,
      TSS2L_SYS_AUTH_RESPONSE       *rspAuthsArray
);
```

This function gets the response authorization data from the sysContext.

This function can only be called after Tss2_Sys_Execute/Tss2_Sys_ExecuteFinish and before the next _Prepare.

Response Codes:

- TSS2_SYS_RC_INVALID_SESSIONS: This RC MUST be returned if the TPM response does not return the same number of authorizations as were sent in the command.

- TSS2_SYS_RC_INSUFFICIENT_CONTEXT: if the provided context structure is too small for the SAPI function.
- TSS2_SYS_RC_BAD_SEQUENCE: if one of the following is true:
  - Tss2_Sys_Execute or Tss2_Sys_ExecuteFinish returned anything other than TPM_RC_SUCCESS
  - If not called after Tss2_Sys_Execute/Tss2_Sys_ExecuteFinish and before the next _Prepare
  - If called for a command that can never take authorizations
- TSS2_SYS_RC_MALFORMED_RESPONSE: if any kind of malformed response is detected.
- TSS2_SYS_RC_BAD_REFERENCE: if *sysContext or rspAuthsArray* are NULL
- TSS2_RCs produced by lower layers (e.g. the TPM, TCTI etc) of the software stack SHALL be returned to the caller unaltered unless handled internally.

3.8.4.3 Tss2_Sys_<COMMAND_NAME>_Complete

To construct a SAPI command for each of the TPM 2.0 Specification Part 3 commands, replace the <COMMAND_NAME> with the appropriate TPM 2.0 Specification Part 3 command with the TPM2_ prefix removed.

```
TSS2_RC Tss2_Sys_<COMMAND_NAME>_Complete(
      TSS2_SYS_CONTEXT        *sysContext,
                              *outHandles,
                              *outParams
);
```

This is a template for a complete function. There is one _Complete function per Part 3 TPM command. The number and types of pointers to output parameters and of the handles are defined in Part 3 of the TPM specification.

This function unmarshals the response parameters and response handles from a previously executed TPM command.

If the caller does not require a certain parameter or handle to be returned, it may pass in NULL for any of the response handles or parameters and these values will not be returned.

This function must only be called after Tss2_Sys_Execute or Tss2_Sys_ExecuteAsync and before the next _Prepare call.

If any parameter is a simple TPM2B:

- On the call to _Complete, its size parameter must be either the caller allocated size of the array or 0 to denote the default allocation size for this type.
- On the return from _Complete, the implementation will write the used size of the array.

The used size is unmarshaled from the response stream. If the used size is greater than the caller allocated size, returns TSS2_SYS_RC_INSUFFICIENT_BUFFER.

NOTE: If the caller reuses the TPM2B, the size must be set back to the caller allocated size before the next call to _Complete or one call function.

If any parameter is a complex TPM2B:

- In the call to _Complete, its size parameter MUST be zero.
- On the return from _Complete, its size parameter will be the unmarshalled version of the size of the TPM2B's UINT8 array as returned from the TPM.

If the first response output parameter is a complex TPM2B, and the second parameter of the TPM2B is encrypted, then the caller has two options:

- The caller can decrypt the parameter before _Complete is called with the first parameter non-NULL. See Tss2_Sys_GetEncryptParam and Tss2_Sys_SetEncryptParam.
- The caller can call Tss2_Sys_Complete with the first parameter NULL. The implementation will not unmarshal it. The caller must decrypt and unmarshal the parameter. See Tss2_Sys_GetEncryptParam.

Response Codes:

- TSS2_SYS_RC_INSUFFICIENT_CONTEXT: if the provided context structure is too small for the SAPI function.
- TSS2_SYS_RC_INSUFFICIENT_BUFFER: if any of the simple TPM2B output parameters do not provide enough buffer space.
- TSS2_SYS_RC_BAD_SEQUENCE: if one of the following is true:
    - Tss2_Sys_Execute or Tss2_Sys_ExecuteFinish returned anything other than TSS2_RC_SUCCESS
    - If not called after Tss2_Sys_Execute or Tss2_Sys_ExecuteFinish and before subsequent Tss2_Sys_<COMMAND_NAME>_Prepare.
- TSS2_SYS_RC_MALFORMED_RESPONSE:
    - MUST be returned if response is larger than the maximum sized response the TSS can receive, meaning the recommended size from Tss2_Sys_GetContextSize function which is returned when 0 is passed in as the requested size.
    - Returned if any kind of malformed response is detected
- TSS2_SYS_RC_BAD_REFERENCE: if sysContext is NULL
- TSS2_RCs produced by lower layers (e.g. the TPM, TCTI etc) of the software stack SHALL be returned to the caller unaltered unless handled internally.

3.8.4.4 Tss2_Sys_GetEncryptParam

```
TSS2_RC Tss2_Sys_GetEncryptParam(
      TSS2_SYS_CONTEXT        *sysContext,
      size_t                        *encryptParamSize,
      const uint8_t                 **encryptParamBuffer
);
```

If the first response parameter is a TPM2B type, this function returns the size of the buffer and a pointer to the buffer of the marshaled TPM2B. If the first parameter is not a TPM2B, this function returns TSS2_SYS_RC_NO_ENCRYPT_PARAM. For consistency with TPM terminology a TPM output parameter that is encrypted by the TPM and decrypted by the application is referred to as an *encrypt* parameter and the session controlling the encryption parameters is referred to as the encrypt session and must have its TPMA_SESSION_ENCRYPT attribute bit set.

This function is only called after Tss2_Sys_Execute or Tss2_Sys_ExecuteFinish function and before the next Tss2_Sys_XXX_Prepare or one-call Tss2_Sys_XXX. Typically, this function will be called between Tss2_Sys_GetRpBuffer and _Complete, since the rpHash must contain the encrypted value before _Complete is called. If _Complete is called without having decrypted the parameter, _Complete may fail with an unmarshalling error.

The application must not write to the returned *encryptParamBuffer* and this buffer may only be read until the next invocation of any function that uses the same sysContext. If any other SAPI call is made to the same *sysContext* a previously retrieved en*cryptParamBuffer* contains undefined data and Tss2_Sys_GetEncryptParam must be called again.

The intent of Tss2_Sys_GetEncryptParam is to provide the size and readable buffer of the parameter to be decrypted by the caller in an encrypted session.

After calling Tss2_Sys_GetEncryptParam, the decrypted result can be set by calling Tss2_Sys_SetEncryptParam; this allows the _Complete call to properly unmarshal the result.

Response Codes:

- TSS2_SYS_RC_BAD_SEQUENCE: if not called in the correct order, or if Tss2_Sys_Execute or Tss2_Sys_ExecuteFinish returned anything other than TPM_RC_SUCCESS
- TSS2_SYS_RC_NO_ENCRYPT_PARAM: if called when sysContext is set for a command that doesn't have an encrypt response parameter.
- TSS2_SYS_RC_MALFORMED_RESPONSE: Returned if any kind of malformed response is detected.
- TSS2_SYS_RC_BAD_REFERENCES: if any of the inputs are NULL.

3.8.4.5 Tss2_Sys_SetEncryptParam

```
TSS2_RC Tss2_Sys_SetEncryptParam(
      TSS2_SYS_CONTEXT        *sysContext,
      size_t                        encryptParamSize,
      const uint8_t                 *encryptParamBuffer
);
```

If the first response parameter is a TPM2B type, this function sets the size and buffer of the TPM2B. The TPM2B size field must match *encryptParamSize*.

This function must only be called after Tss2_Sys_Execute / Tss2_Sys_ExecuteFinish and before the next _Prepare or one call.

The intent of this function is to set the first response TPM2B parameter's UINT8 array to the decrypted value after the caller has decrypted the parameter and before the response parameter is unmarshaled using _Complete.

It is typically called after the rpHash calculation, since the rpHash calculation uses the encrypted version of this parameter.

Response Codes:

- TSS2_SYS_RC_BAD_SEQUENCE: if not called after Tss2_Sys_Execute/Tss2_Sys_ExecuteAsync and before _Complete.
- TSS2_SYS_RC_INSUFFICIENT_CONTEXT: if the provided context structure is too small for the SAPI function.

- TSS2_SYS_RC_BAD_REFERENCE: if *sysContext* or *encryptParambuffer* is null.
- TSS2_SYS_RC_BAD_SIZE: if the first TPM2B parameter size field does not equal *encryptParamSize*.
- TSS2_SYS_RC_NO_ENCRYPT_PARAM: if called when *sysContext* is set for a response that doesn't have an encrypt response parameter.
- TSS2_RCs produced by lower layers (e.g. the TPM, TCTI etc) of the software stack SHALL be returned to the caller unaltered unless handled internally.

3.8.4.6 Tss2_Sys_GetRpBuffer

```
TSS2_RC Tss2_Sys_GetRpBuffer(
      TSS2_SYS_CONTEXT        *sysContext,
      size_t                        *rpBufferUsedSize,
      const uint8_t                 **rpBuffer
);
```

This function returns a pointer to the rpBuffer, a pointer to the marshaled response parameters, and the used rpBuffer bytes after command execution.

This function is only called after Tss2_Sys_Execute, or Tss2_Sys_ExecuteFinish(), or one call function and before the next _Prepare.

The caller must not write to the returned rpBuffer and this buffer may only be read until the next invocation of any function that uses the same sysContext. If any other SAPI call is being made to the same sysContext a previously retrieved rpBuffer contains undefined data and Tss2_Sys_GetRpBuffer must be called again.

The rpBuffer is used in the calculation of the rpHash value for response authorization. Tss2_Sys_GetRpBuffer is typically called before an optional Tss2_Sys_GetEncryptParam /Tss2_Sys_SetEncryptParam and before a Tss2_Sys_XXX_Complete.

If Tss2_Sys_GetRpBuffer is called for a command that doesn't return any parameters, rpBufferUsedSize should be set to 0 and TSS2_RC_SUCCESS returned.

Response Codes**:**

- TSS2_SYS_RC_BAD_SEQUENCE:
    - If Tss2_Sys_Execute, Tss2_Sys_ExecuteFinish, or one-call function returned anything other than TPM_RC_SUCCESS.
    - If not called after Tss2_Sys_Execute, Tss2_Sys_ExecuteFinish, or one-call function and before subsequent Tss2_Sys_<COMMAND_NAME>_Prepare.
- TSS2_SYS_RC_MALFORMED_RESPONSE: Returned if any kind of malformed response is detected.
- TSS2_SYS_RC_BAD_REFERENCE: if any of the inputs are NULL.

4 SYS Header File

tss2_sys.h

4.1 tss2_sys.h Prelude

```
#ifndef TSS2_SYS_H
#define TSS2_SYS_H

#include <stdlib.h>
#include "tss2_common.h"
#include "tss2_tpm2_types.h"
#include "tss2_tcti.h"

#ifndef TSS2_API_VERSION_1_2_1_108
#error Version mismatch among TSS2 header files.
#endif

#ifdef __cplusplus
extern "C" {
#endif
```

4.2 tss2_sys.h sysContext Structure

```
/*
 * System API Structures
 */

/* Opaque context structure */
typedef struct TSS2_SYS_OPAQUE_CONTEXT_BLOB TSS2_SYS_CONTEXT;
```

4.3 tss2_sys.h Command and Response Session Structures

This structure is used to set the session data that is passed to and returned from the SAPI Part 3 functions. Input structure for command authorization area(s).

```
/* Maximum number of sessions supported in a command */
#define TSS2_SYS_MAX_SESSIONS 3
```



```
/* Structures to hold authorization data to and from the TPM */
typedef struct {
    uint16_t          count;
    TPMS_AUTH_COMMAND auths[TSS2_SYS_MAX_SESSIONS];
} TSS2L_SYS_AUTH_COMMAND;

typedef struct {
    uint16_t           count;
    TPMS_AUTH_RESPONSE auths[TSS2_SYS_MAX_SESSIONS];
} TSS2L_SYS_AUTH_RESPONSE;
```

4.4 tss2_sys.h Command Context Management Functions

```
/*
 * System API Context Management Functions
 */
TSS2_DLL_EXPORT size_t Tss2_Sys_GetContextSize(
    size_t maxCommandReponseSize);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_Initialize(
    TSS2_SYS_CONTEXT  *sysContext,
    size_t             contextSize,
    TSS2_TCTI_CONTEXT *tctiContext,
    TSS2_ABI_VERSION  *abiVersion);

TSS2_DLL_EXPORT void Tss2_Sys_Finalize(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_GetTctiContext(
    TSS2_SYS_CONTEXT   *sysContext,
    TSS2_TCTI_CONTEXT **tctiContext);
```

4.5 tss2_sys.h Command Preparation Functions

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_GetDecryptParam(
```

```
    TSS2_SYS_CONTEXT *sysContext,
    size_t           *decryptParamSize,
    uint8_t   const **decryptParamBuffer);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_SetDecryptParam(
    TSS2_SYS_CONTEXT *sysContext,
    size_t            decryptParamSize,
    uint8_t    const *decryptParamBuffer);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_GetCpBuffer(
    TSS2_SYS_CONTEXT *sysContext,
    size_t           *cpBufferUsedSize,
    uint8_t   const **cpBuffer);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_SetCmdAuths(
    TSS2_SYS_CONTEXT             *sysContext,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths);
```

4.6 tss2_sys.h Command Execution Functions

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_ExecuteAsync(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_ExecuteFinish(
    TSS2_SYS_CONTEXT *sysContext,
    int32_t           timeout);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_Execute(
    TSS2_SYS_CONTEXT *sysContext);
```

4.7 tss2_sys.h Command Completion Functions

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_GetCommandCode(
    TSS2_SYS_CONTEXT *sysContext,
    UINT8            *commandCode);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_GetRspAuths(
```

```
    TSS2_SYS_CONTEXT        *sysContext,
    TSS2L_SYS_AUTH_RESPONSE *rspAuths);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_GetEncryptParam(
    TSS2_SYS_CONTEXT *sysContext,
    size_t           *encryptParamSize,
    uint8_t   const **encryptParamBuffer);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_SetEncryptParam(
    TSS2_SYS_CONTEXT *sysContext,
    size_t            encryptParamSize,
    uint8_t    const *encryptParamBuffer);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_GetRpBuffer(
    TSS2_SYS_CONTEXT *sysContext,
    size_t           *rpBufferUsedSize,
    uint8_t   const **rpBuffer);
```

4.8 tss2_sys.h Functions for Invoking TPM Commands

```
/*
 * The following functions are the Prepare, Complete, and One-Shot
 * functions corresponding to each command in part 3 of the TPM
 * specification.
 */
```

4.8.1 TPM Tss2_Sys_Startup Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_Startup_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
    TPM2_SU           startupType);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_Startup_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_Startup(
    TSS2_SYS_CONTEXT *sysContext,
```

```
    TPM2_SU           startupType);
```

4.8.2 TPM Tss2_Sys_Shutdown Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_Shutdown_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
    TPM2_SU           shutdownType);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_Shutdown_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_Shutdown(
    TSS2_SYS_CONTEXT             *sysContext,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2_SU                       shutdownType,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.3 TPM Tss2_Sys_SelfTest Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_SelfTest_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
    TPMI_YES_NO       fullTest);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_SelfTest_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_SelfTest(
    TSS2_SYS_CONTEXT             *sysContext,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPMI_YES_NO                   fullTest,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.4 TPM Tss2_Sys_IncrementalSelfTest Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_IncrementalSelfTest_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
    TPML_ALG   const *toTest);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_IncrementalSelfTest_Complete(
    TSS2_SYS_CONTEXT *sysContext,
    TPML_ALG         *toDoList);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_IncrementalSelfTest(
    TSS2_SYS_CONTEXT             *sysContext,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPML_ALG               const *toTest,
    TPML_ALG                     *toDoList,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.5 TPM Tss2_Sys_GetTestResult Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_GetTestResult_Prepare(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_GetTestResult_Complete(
    TSS2_SYS_CONTEXT *sysContext,
    TPM2B_MAX_BUFFER *outData,
    TPM2_RC          *testResult);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_GetTestResult(
    TSS2_SYS_CONTEXT             *sysContext,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_MAX_BUFFER             *outData,
    TPM2_RC                      *testResult,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.6 TPM Tss2_Sys_StartAuthSession Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_StartAuthSession_Prepare(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_DH_OBJECT                tpmKey,
    TPMI_DH_ENTITY                bind,
    TPM2B_NONCE            const *nonceCaller,
    TPM2B_ENCRYPTED_SECRET const *encryptedSalt,
    TPM2_SE                       sessionType,
    TPMT_SYM_DEF           const *symmetric,
    TPMI_ALG_HASH                 authHash);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_StartAuthSession_Complete(
    TSS2_SYS_CONTEXT     *sysContext,
    TPMI_SH_AUTH_SESSION *sessionHandle,
    TPM2B_NONCE          *nonceTPM);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_StartAuthSession(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_DH_OBJECT                tpmKey,
    TPMI_DH_ENTITY                bind,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_NONCE            const *nonceCaller,
    TPM2B_ENCRYPTED_SECRET const *encryptedSalt,
    TPM2_SE                       sessionType,
    TPMT_SYM_DEF           const *symmetric,
    TPMI_ALG_HASH                 authHash,
    TPMI_SH_AUTH_SESSION         *sessionHandle,
    TPM2B_NONCE                  *nonceTPM,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.7 TPM Tss2_Sys_PolicyRestart Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyRestart_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
    TPMI_SH_POLICY    sessionHandle);
```

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyRestart_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyRestart(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_SH_POLICY                sessionHandle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.8 TPM Tss2_Sys_Create Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_Create_Prepare(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_DH_OBJECT                parentHandle,
    TPM2B_SENSITIVE_CREATE const *inSensitive,
    TPM2B_PUBLIC           const *inPublic,
    TPM2B_DATA             const *outsideInfo,
    TPML_PCR_SELECTION     const *creationPCR);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_Create_Complete(
    TSS2_SYS_CONTEXT    *sysContext,
    TPM2B_PRIVATE       *outPrivate,
    TPM2B_PUBLIC        *outPublic,
    TPM2B_CREATION_DATA *creationData,
    TPM2B_DIGEST        *creationHash,
    TPMT_TK_CREATION    *creationTicket);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_Create(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_DH_OBJECT                parentHandle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_SENSITIVE_CREATE const *inSensitive,
    TPM2B_PUBLIC           const *inPublic,
    TPM2B_DATA             const *outsideInfo,
    TPML_PCR_SELECTION     const *creationPCR,
```

```
    TPM2B_PRIVATE               *outPrivate,
    TPM2B_PUBLIC                *outPublic,
    TPM2B_CREATION_DATA         *creationData,
    TPM2B_DIGEST                *creationHash,
    TPMT_TK_CREATION            *creationTicket,
    TSS2L_SYS_AUTH_RESPONSE     *rspAuths);
```

4.8.9 TPM Tss2_Sys_Load Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_Load_Prepare(
    TSS2_SYS_CONTEXT    *sysContext,
    TPMI_DH_OBJECT       parentHandle,
    TPM2B_PRIVATE const *inPrivate,
    TPM2B_PUBLIC  const *inPublic);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_Load_Complete(
    TSS2_SYS_CONTEXT *sysContext,
    TPM2_HANDLE      *objectHandle,
    TPM2B_NAME       *name);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_Load(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_DH_OBJECT                parentHandle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_PRIVATE          const *inPrivate,
    TPM2B_PUBLIC           const *inPublic,
    TPM2_HANDLE                  *objectHandle,
    TPM2B_NAME                   *name,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.10 TPM Tss2_Sys_LoadExternal Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_LoadExternal_Prepare(
    TSS2_SYS_CONTEXT      *sysContext,
    TPM2B_SENSITIVE const *inPrivate,
```

```
    TPM2B_PUBLIC    const *inPublic,
    TPMI_RH_HIERARCHY      hierarchy);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_LoadExternal_Complete(
    TSS2_SYS_CONTEXT *sysContext,
    TPM2_HANDLE      *objectHandle,
    TPM2B_NAME       *name);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_LoadExternal(
    TSS2_SYS_CONTEXT             *sysContext,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_SENSITIVE        const *inPrivate,
    TPM2B_PUBLIC           const *inPublic,
    TPMI_RH_HIERARCHY             hierarchy,
    TPM2_HANDLE                  *objectHandle,
    TPM2B_NAME                   *name,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.11 TPM Tss2_Sys_ReadPublic Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_ReadPublic_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
    TPMI_DH_OBJECT    objectHandle);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_ReadPublic_Complete(
    TSS2_SYS_CONTEXT *sysContext,
    TPM2B_PUBLIC     *outPublic,
    TPM2B_NAME       *name,
    TPM2B_NAME       *qualifiedName);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_ReadPublic(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_DH_OBJECT                objectHandle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_PUBLIC                 *outPublic,
    TPM2B_NAME                   *name,
```

```
    TPM2B_NAME                  *qualifiedName,
    TSS2L_SYS_AUTH_RESPONSE     *rspAuths);
```

4.8.12 TPM Tss2_Sys_ActivateCredential Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_ActivateCredential_Prepare(
    TSS2_SYS_CONTEXT            *sysContext,
    TPMI_DH_OBJECT               activateHandle,
    TPMI_DH_OBJECT               keyHandle,
    TPM2B_ID_OBJECT       const *credentialBlob,
    TPM2B_ENCRYPTED_SECRET const *secret);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_ActivateCredential_Complete(
    TSS2_SYS_CONTEXT *sysContext,
    TPM2B_DIGEST     *certInfo);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_ActivateCredential(
    TSS2_SYS_CONTEXT            *sysContext,
    TPMI_DH_OBJECT               activateHandle,
    TPMI_DH_OBJECT               keyHandle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_ID_OBJECT       const *credentialBlob,
    TPM2B_ENCRYPTED_SECRET const *secret,
    TPM2B_DIGEST                 *certInfo,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.13 TPM Tss2_Sys_MakeCredential Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_MakeCredential_Prepare(
    TSS2_SYS_CONTEXT   *sysContext,
    TPMI_DH_OBJECT      handle,
    TPM2B_DIGEST const *credential,
    TPM2B_NAME   const *objectName);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_MakeCredential_Complete(
```

```
    TSS2_SYS_CONTEXT       *sysContext,
    TPM2B_ID_OBJECT        *credentialBlob,
    TPM2B_ENCRYPTED_SECRET *secret);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_MakeCredential(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_DH_OBJECT                handle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_DIGEST           const *credential,
    TPM2B_NAME             const *objectName,
    TPM2B_ID_OBJECT              *credentialBlob,
    TPM2B_ENCRYPTED_SECRET       *secret,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.14 TPM Tss2_Sys_Unseal Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_Unseal_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
    TPMI_DH_OBJECT    itemHandle);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_Unseal_Complete(
    TSS2_SYS_CONTEXT     *sysContext,
    TPM2B_SENSITIVE_DATA *outData);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_Unseal(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_DH_OBJECT                itemHandle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_SENSITIVE_DATA         *outData,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.15 TPM Tss2_Sys_ObjectChangeAuth Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_ObjectChangeAuth_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
```

```
    TPMI_DH_OBJECT    objectHandle,
    TPMI_DH_OBJECT    parentHandle,
    TPM2B_AUTH const *newAuth);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_ObjectChangeAuth_Complete(
    TSS2_SYS_CONTEXT *sysContext,
    TPM2B_PRIVATE    *outPrivate);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_ObjectChangeAuth(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_DH_OBJECT                objectHandle,
    TPMI_DH_OBJECT                parentHandle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_AUTH             const *newAuth,
    TPM2B_PRIVATE                 *outPrivate,
    TSS2L_SYS_AUTH_RESPONSE       *rspAuths);
```

4.8.16 TPM Tss2_Sys_CreateLoaded Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_CreateLoaded_Prepare(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_DH_PARENT                parentHandle,
    TPM2B_SENSITIVE_CREATE const *inSensitive,
    TPM2B_TEMPLATE         const *inPublic);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_CreateLoaded_Complete(
    TSS2_SYS_CONTEXT *sysContext,
    TPM2_HANDLE      *objectHandle,
    TPM2B_PRIVATE    *outPrivate,
    TPM2B_PUBLIC     *outPublic,
    TPM2B_NAME       *name);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_CreateLoaded(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_DH_PARENT                parentHandle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
```

```
    TPM2B_SENSITIVE_CREATE const *inSensitive,
    TPM2B_TEMPLATE         const *inPublic,
    TPM2_HANDLE                  *objectHandle,
    TPM2B_PRIVATE                *outPrivate,
    TPM2B_PUBLIC                 *outPublic,
    TPM2B_NAME                   *name,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.17 TPM Tss2_Sys_Duplicate Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_Duplicate_Prepare(
    TSS2_SYS_CONTEXT          *sysContext,
    TPMI_DH_OBJECT             objectHandle,
    TPMI_DH_OBJECT             newParentHandle,
    TPM2B_DATA          const *encryptionKeyIn,
    TPMT_SYM_DEF_OBJECT const *symmetricAlg);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_Duplicate_Complete(
    TSS2_SYS_CONTEXT       *sysContext,
    TPM2B_DATA             *encryptionKeyOut,
    TPM2B_PRIVATE          *duplicate,
    TPM2B_ENCRYPTED_SECRET *outSymSeed);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_Duplicate(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_DH_OBJECT                objectHandle,
    TPMI_DH_OBJECT                newParentHandle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_DATA             const *encryptionKeyIn,
    TPMT_SYM_DEF_OBJECT    const *symmetricAlg,
    TPM2B_DATA                   *encryptionKeyOut,
    TPM2B_PRIVATE                *duplicate,
    TPM2B_ENCRYPTED_SECRET       *outSymSeed,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.18 TPM Tss2_Sys_Rewrap Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_Rewrap_Prepare(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_DH_OBJECT                oldParent,
    TPMI_DH_OBJECT                newParent,
    TPM2B_PRIVATE          const *inDuplicate,
    TPM2B_NAME             const *name,
    TPM2B_ENCRYPTED_SECRET const *inSymSeed);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_Rewrap_Complete(
    TSS2_SYS_CONTEXT       *sysContext,
    TPM2B_PRIVATE          *outDuplicate,
    TPM2B_ENCRYPTED_SECRET *outSymSeed);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_Rewrap(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_DH_OBJECT                oldParent,
    TPMI_DH_OBJECT                newParent,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_PRIVATE          const *inDuplicate,
    TPM2B_NAME             const *name,
    TPM2B_ENCRYPTED_SECRET const *inSymSeed,
    TPM2B_PRIVATE                *outDuplicate,
    TPM2B_ENCRYPTED_SECRET       *outSymSeed,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.19 TPM Tss2_Sys_Import Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_Import_Prepare(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_DH_OBJECT                parentHandle,
    TPM2B_DATA             const *encryptionKey,
    TPM2B_PUBLIC           const *objectPublic,
    TPM2B_PRIVATE          const *duplicate,
    TPM2B_ENCRYPTED_SECRET const *inSymSeed,
    TPMT_SYM_DEF_OBJECT    const *symmetricAlg);
```

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_Import_Complete(
    TSS2_SYS_CONTEXT *sysContext,
    TPM2B_PRIVATE    *outPrivate);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_Import(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_DH_OBJECT                parentHandle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_DATA             const *encryptionKey,
    TPM2B_PUBLIC           const *objectPublic,
    TPM2B_PRIVATE          const *duplicate,
    TPM2B_ENCRYPTED_SECRET const *inSymSeed,
    TPMT_SYM_DEF_OBJECT    const *symmetricAlg,
    TPM2B_PRIVATE                *outPrivate,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.20 TPM Tss2_Sys_RSA_Encrypt Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_RSA_Encrypt_Prepare(
    TSS2_SYS_CONTEXT           *sysContext,
    TPMI_DH_OBJECT              keyHandle,
    TPM2B_PUBLIC_KEY_RSA const *message,
    TPMT_RSA_DECRYPT     const *inScheme,
    TPM2B_DATA           const *label);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_RSA_Encrypt_Complete(
    TSS2_SYS_CONTEXT     *sysContext,
    TPM2B_PUBLIC_KEY_RSA *outData);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_RSA_Encrypt(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_DH_OBJECT                keyHandle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_PUBLIC_KEY_RSA   const *message,
    TPMT_RSA_DECRYPT       const *inScheme,
```

```
    TPM2B_DATA             const *label,
    TPM2B_PUBLIC_KEY_RSA         *outData,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.21 TPM Tss2_Sys_RSA_Decrypt Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_RSA_Decrypt_Prepare(
    TSS2_SYS_CONTEXT           *sysContext,
    TPMI_DH_OBJECT              keyHandle,
    TPM2B_PUBLIC_KEY_RSA const *cipherText,
    TPMT_RSA_DECRYPT     const *inScheme,
    TPM2B_DATA           const *label);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_RSA_Decrypt_Complete(
    TSS2_SYS_CONTEXT     *sysContext,
    TPM2B_PUBLIC_KEY_RSA *message);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_RSA_Decrypt(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_DH_OBJECT                keyHandle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_PUBLIC_KEY_RSA   const *cipherText,
    TPMT_RSA_DECRYPT       const *inScheme,
    TPM2B_DATA             const *label,
    TPM2B_PUBLIC_KEY_RSA         *message,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.22 TPM Tss2_Sys_ECDH_Keygen Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_ECDH_KeyGen_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
    TPMI_DH_OBJECT    keyHandle);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_ECDH_KeyGen_Complete(
    TSS2_SYS_CONTEXT *sysContext,
```

```
    TPM2B_ECC_POINT  *zPoint,
    TPM2B_ECC_POINT  *pubPoint);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_ECDH_KeyGen(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_DH_OBJECT                keyHandle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_ECC_POINT              *zPoint,
    TPM2B_ECC_POINT              *pubPoint,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.23 TPM Tss2_Sys_ECDH_ZGen Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_ECDH_ZGen_Prepare(
    TSS2_SYS_CONTEXT      *sysContext,
    TPMI_DH_OBJECT         keyHandle,
    TPM2B_ECC_POINT const *inPoint);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_ECDH_ZGen_Complete(
    TSS2_SYS_CONTEXT *sysContext,
    TPM2B_ECC_POINT  *outPoint);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_ECDH_ZGen(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_DH_OBJECT                keyHandle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_ECC_POINT        const *inPoint,
    TPM2B_ECC_POINT              *outPoint,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.24 TPM Tss2_Sys_ECC_Parameters Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_ECC_Parameters_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
    TPMI_ECC_CURVE    curveID);
```

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_ECC_Parameters_Complete(
    TSS2_SYS_CONTEXT          *sysContext,
    TPMS_ALGORITHM_DETAIL_ECC *parameters);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_ECC_Parameters(
    TSS2_SYS_CONTEXT             *sysContext,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPMI_ECC_CURVE                curveID,
    TPMS_ALGORITHM_DETAIL_ECC    *parameters,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.25 TPM Tss2_Sys_ZGen_2Phase Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_ZGen_2Phase_Prepare(
    TSS2_SYS_CONTEXT      *sysContext,
    TPMI_DH_OBJECT         keyA,
    TPM2B_ECC_POINT const *inQsB,
    TPM2B_ECC_POINT const *inQeB,
    TPMI_ECC_KEY_EXCHANGE  inScheme,
    UINT16                 counter);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_ZGen_2Phase_Complete(
    TSS2_SYS_CONTEXT *sysContext,
    TPM2B_ECC_POINT  *outZ1,
    TPM2B_ECC_POINT  *outZ2);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_ZGen_2Phase(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_DH_OBJECT                keyA,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_ECC_POINT        const *inQsB,
    TPM2B_ECC_POINT        const *inQeB,
    TPMI_ECC_KEY_EXCHANGE         inScheme,
    UINT16                        counter,
    TPM2B_ECC_POINT              *outZ1,
```

```
    TPM2B_ECC_POINT              *outZ2,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.26 TPM Tss2_Sys_EncryptDecrypt Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_EncryptDecrypt_Prepare(
    TSS2_SYS_CONTEXT       *sysContext,
    TPMI_DH_OBJECT          keyHandle,
    TPMI_YES_NO             decrypt,
    TPMI_ALG_SYM_MODE       mode,
    TPM2B_IV         const *ivIn,
    TPM2B_MAX_BUFFER const *inData);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_EncryptDecrypt_Complete(
    TSS2_SYS_CONTEXT *sysContext,
    TPM2B_MAX_BUFFER *outData,
    TPM2B_IV         *ivOut);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_EncryptDecrypt(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_DH_OBJECT                keyHandle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPMI_YES_NO                   decrypt,
    TPMI_ALG_SYM_MODE             mode,
    TPM2B_IV               const *ivIn,
    TPM2B_MAX_BUFFER       const *inData,
    TPM2B_MAX_BUFFER             *outData,
    TPM2B_IV                     *ivOut,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.27 TPM Tss2_Sys_EncryptDecrypt2 Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_EncryptDecrypt2_Prepare(
    TSS2_SYS_CONTEXT       *sysContext,
    TPMI_DH_OBJECT          keyHandle,
```

```
    TPM2B_MAX_BUFFER const *inData,
    TPMI_YES_NO             decrypt,
    TPMI_ALG_SYM_MODE       mode,
    TPM2B_IV         const *ivIn);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_EncryptDecrypt2_Complete(
    TSS2_SYS_CONTEXT *sysContext,
    TPM2B_MAX_BUFFER *outData,
    TPM2B_IV         *ivOut);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_EncryptDecrypt2(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_DH_OBJECT                keyHandle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_MAX_BUFFER       const *inData,
    TPMI_YES_NO                   decrypt,
    TPMI_ALG_SYM_MODE             mode,
    TPM2B_IV               const *ivIn,
    TPM2B_MAX_BUFFER              *outData,
    TPM2B_IV                      *ivOut,
    TSS2L_SYS_AUTH_RESPONSE       *rspAuths);
```

4.8.28 TPM Tss2_Sys_Hash Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_Hash_Prepare(
    TSS2_SYS_CONTEXT       *sysContext,
    TPM2B_MAX_BUFFER const *data,
    TPMI_ALG_HASH           hashAlg,
    TPMI_RH_HIERARCHY       hierarchy);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_Hash_Complete(
    TSS2_SYS_CONTEXT  *sysContext,
    TPM2B_DIGEST      *outHash,
    TPMT_TK_HASHCHECK *validation);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_Hash(
```

```
    TSS2_SYS_CONTEXT             *sysContext,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_MAX_BUFFER       const *data,
    TPMI_ALG_HASH                 hashAlg,
    TPMI_RH_HIERARCHY             hierarchy,
    TPM2B_DIGEST                 *outHash,
    TPMT_TK_HASHCHECK            *validation,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.29 TPM Tss2_Sys_HMAC Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_HMAC_Prepare(
    TSS2_SYS_CONTEXT       *sysContext,
    TPMI_DH_OBJECT          handle,
    TPM2B_MAX_BUFFER const *buffer,
    TPMI_ALG_HASH           hashAlg);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_HMAC_Complete(
    TSS2_SYS_CONTEXT *sysContext,
    TPM2B_DIGEST     *outHMAC);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_HMAC(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_DH_OBJECT                handle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_MAX_BUFFER       const *buffer,
    TPMI_ALG_HASH                 hashAlg,
    TPM2B_DIGEST                 *outHMAC,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.30 TPM Tss2_Sys_GetRandom Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_GetRandom_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
    UINT16            bytesRequested);
```

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_GetRandom_Complete(
    TSS2_SYS_CONTEXT *sysContext,
    TPM2B_DIGEST     *randomBytes);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_GetRandom(
    TSS2_SYS_CONTEXT             *sysContext,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    UINT16                        bytesRequested,
    TPM2B_DIGEST                 *randomBytes,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.31 TPM Tss2_Sys_StirRandom Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_StirRandom_Prepare(
    TSS2_SYS_CONTEXT           *sysContext,
    TPM2B_SENSITIVE_DATA const *inData);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_StirRandom_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_StirRandom(
    TSS2_SYS_CONTEXT             *sysContext,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_SENSITIVE_DATA   const *inData,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.32 TPM Tss2_Sys_HMAC_Start Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_HMAC_Start_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
    TPMI_DH_OBJECT    handle,
    TPM2B_AUTH const *auth,
    TPMI_ALG_HASH     hashAlg);
```

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_HMAC_Start_Complete(
    TSS2_SYS_CONTEXT *sysContext,
    TPMI_DH_OBJECT   *sequenceHandle);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_HMAC_Start(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_DH_OBJECT                handle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_AUTH             const *auth,
    TPMI_ALG_HASH                 hashAlg,
    TPMI_DH_OBJECT               *sequenceHandle,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.33 TPM Tss2_Sys_HashSequenceStart Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_HashSequenceStart_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
    TPM2B_AUTH const *auth,
    TPMI_ALG_HASH     hashAlg);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_HashSequenceStart_Complete(
    TSS2_SYS_CONTEXT *sysContext,
    TPMI_DH_OBJECT   *sequenceHandle);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_HashSequenceStart(
    TSS2_SYS_CONTEXT             *sysContext,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_AUTH             const *auth,
    TPMI_ALG_HASH                 hashAlg,
    TPMI_DH_OBJECT               *sequenceHandle,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.34 TPM Tss2_Sys_SequenceUpdate Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_SequenceUpdate_Prepare(
```

```
    TSS2_SYS_CONTEXT       *sysContext,
    TPMI_DH_OBJECT          sequenceHandle,
    TPM2B_MAX_BUFFER const *buffer);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_SequenceUpdate_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_SequenceUpdate(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_DH_OBJECT                sequenceHandle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_MAX_BUFFER       const *buffer,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.35 TPM Tss2_Sys_SequenceComplete Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_SequenceComplete_Prepare(
    TSS2_SYS_CONTEXT       *sysContext,
    TPMI_DH_OBJECT          sequenceHandle,
    TPM2B_MAX_BUFFER const *buffer,
    TPMI_RH_HIERARCHY       hierarchy);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_SequenceComplete_Complete(
    TSS2_SYS_CONTEXT  *sysContext,
    TPM2B_DIGEST      *result,
    TPMT_TK_HASHCHECK *validation);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_SequenceComplete(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_DH_OBJECT                sequenceHandle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_MAX_BUFFER       const *buffer,
    TPMI_RH_HIERARCHY             hierarchy,
    TPM2B_DIGEST                 *result,
    TPMT_TK_HASHCHECK            *validation,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.36 TPM Tss2_Sys_EventSequenceComplete Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_EventSequenceComplete_Prepare(
    TSS2_SYS_CONTEXT       *sysContext,
    TPMI_DH_PCR             pcrHandle,
    TPMI_DH_OBJECT          sequenceHandle,
    TPM2B_MAX_BUFFER const *buffer);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_EventSequenceComplete_Complete(
    TSS2_SYS_CONTEXT   *sysContext,
    TPML_DIGEST_VALUES *results);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_EventSequenceComplete(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_DH_PCR                   pcrHandle,
    TPMI_DH_OBJECT                sequenceHandle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_MAX_BUFFER       const *buffer,
    TPML_DIGEST_VALUES           *results,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.37 TPM Tss2_Sys_Certify Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_Certify_Prepare(
    TSS2_SYS_CONTEXT      *sysContext,
    TPMI_DH_OBJECT         objectHandle,
    TPMI_DH_OBJECT         signHandle,
    TPM2B_DATA      const *qualifyingData,
    TPMT_SIG_SCHEME const *inScheme);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_Certify_Complete(
    TSS2_SYS_CONTEXT *sysContext,
    TPM2B_ATTEST     *certifyInfo,
    TPMT_SIGNATURE   *signature);
```

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_Certify(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_DH_OBJECT                objectHandle,
    TPMI_DH_OBJECT                signHandle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_DATA             const *qualifyingData,
    TPMT_SIG_SCHEME        const *inScheme,
    TPM2B_ATTEST                 *certifyInfo,
    TPMT_SIGNATURE               *signature,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.38 TPM Tss2_Sys_CertifyCreation Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_CertifyCreation_Prepare(
    TSS2_SYS_CONTEXT       *sysContext,
    TPMI_DH_OBJECT          signHandle,
    TPMI_DH_OBJECT          objectHandle,
    TPM2B_DATA       const *qualifyingData,
    TPM2B_DIGEST     const *creationHash,
    TPMT_SIG_SCHEME  const *inScheme,
    TPMT_TK_CREATION const *creationTicket);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_CertifyCreation_Complete(
    TSS2_SYS_CONTEXT *sysContext,
    TPM2B_ATTEST     *certifyInfo,
    TPMT_SIGNATURE   *signature);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_CertifyCreation(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_DH_OBJECT                signHandle,
    TPMI_DH_OBJECT                objectHandle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_DATA             const *qualifyingData,
    TPM2B_DIGEST           const *creationHash,
    TPMT_SIG_SCHEME        const *inScheme,
```

```
    TPMT_TK_CREATION       const *creationTicket,
    TPM2B_ATTEST                 *certifyInfo,
    TPMT_SIGNATURE               *signature,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.39 TPM Tss2_Sys_Quote Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_Quote_Prepare(
    TSS2_SYS_CONTEXT         *sysContext,
    TPMI_DH_OBJECT            signHandle,
    TPM2B_DATA         const *qualifyingData,
    TPMT_SIG_SCHEME    const *inScheme,
    TPML_PCR_SELECTION const *PCRselect);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_Quote_Complete(
    TSS2_SYS_CONTEXT *sysContext,
    TPM2B_ATTEST     *quoted,
    TPMT_SIGNATURE   *signature);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_Quote(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_DH_OBJECT                signHandle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_DATA             const *qualifyingData,
    TPMT_SIG_SCHEME        const *inScheme,
    TPML_PCR_SELECTION     const *PCRselect,
    TPM2B_ATTEST                 *quoted,
    TPMT_SIGNATURE               *signature,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.40 TPM Tss2_Sys_GetSessionAuditDigest Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_GetSessionAuditDigest_Prepare(
    TSS2_SYS_CONTEXT      *sysContext,
    TPMI_RH_ENDORSEMENT    privacyAdminHandle,
```

```
    TPMI_DH_OBJECT          signHandle,
    TPMI_SH_HMAC            sessionHandle,
    TPM2B_DATA      const *qualifyingData,
    TPMT_SIG_SCHEME const *inScheme);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_GetSessionAuditDigest_Complete(
    TSS2_SYS_CONTEXT *sysContext,
    TPM2B_ATTEST     *auditInfo,
    TPMT_SIGNATURE   *signature);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_GetSessionAuditDigest(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_RH_ENDORSEMENT           privacyAdminHandle,
    TPMI_DH_OBJECT                signHandle,
    TPMI_SH_HMAC                  sessionHandle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_DATA             const *qualifyingData,
    TPMT_SIG_SCHEME        const *inScheme,
    TPM2B_ATTEST                 *auditInfo,
    TPMT_SIGNATURE               *signature,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.41 TPM Tss2_Sys_GetCommandAuditDigest Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_GetCommandAuditDigest_Prepare(
    TSS2_SYS_CONTEXT      *sysContext,
    TPMI_RH_ENDORSEMENT    privacyHandle,
    TPMI_DH_OBJECT         signHandle,
    TPM2B_DATA      const *qualifyingData,
    TPMT_SIG_SCHEME const *inScheme);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_GetCommandAuditDigest_Complete(
    TSS2_SYS_CONTEXT *sysContext,
    TPM2B_ATTEST     *auditInfo,
    TPMT_SIGNATURE   *signature);
```

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_GetCommandAuditDigest(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_RH_ENDORSEMENT           privacyHandle,
    TPMI_DH_OBJECT                signHandle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_DATA             const *qualifyingData,
    TPMT_SIG_SCHEME        const *inScheme,
    TPM2B_ATTEST                 *auditInfo,
    TPMT_SIGNATURE               *signature,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.42 TPM Tss2_Sys_GetTime Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_GetTime_Prepare(
    TSS2_SYS_CONTEXT      *sysContext,
    TPMI_RH_ENDORSEMENT    privacyAdminHandle,
    TPMI_DH_OBJECT         signHandle,
    TPM2B_DATA      const *qualifyingData,
    TPMT_SIG_SCHEME const *inScheme);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_GetTime_Complete(
    TSS2_SYS_CONTEXT *sysContext,
    TPM2B_ATTEST     *timeInfo,
    TPMT_SIGNATURE   *signature);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_GetTime(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_RH_ENDORSEMENT           privacyAdminHandle,
    TPMI_DH_OBJECT                signHandle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_DATA             const *qualifyingData,
    TPMT_SIG_SCHEME        const *inScheme,
    TPM2B_ATTEST                 *timeInfo,
    TPMT_SIGNATURE               *signature,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.43 TPM Tss2_Sys_Commit Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_Commit_Prepare(
    TSS2_SYS_CONTEXT           *sysContext,
    TPMI_DH_OBJECT              signHandle,
    TPM2B_ECC_POINT      const *P1,
    TPM2B_SENSITIVE_DATA const *s2,
    TPM2B_ECC_PARAMETER  const *y2);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_Commit_Complete(
    TSS2_SYS_CONTEXT *sysContext,
    TPM2B_ECC_POINT  *K,
    TPM2B_ECC_POINT  *L,
    TPM2B_ECC_POINT  *E,
    UINT16           *counter);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_Commit(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_DH_OBJECT                signHandle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_ECC_POINT        const *P1,
    TPM2B_SENSITIVE_DATA   const *s2,
    TPM2B_ECC_PARAMETER    const *y2,
    TPM2B_ECC_POINT              *K,
    TPM2B_ECC_POINT              *L,
    TPM2B_ECC_POINT              *E,
    UINT16                       *counter,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.44 TPM Tss2_Sys_EC_Ephemeral Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_EC_Ephemeral_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
    TPMI_ECC_CURVE    curveID);
```

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_EC_Ephemeral_Complete(
    TSS2_SYS_CONTEXT *sysContext,
    TPM2B_ECC_POINT  *Q,
    UINT16           *counter);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_EC_Ephemeral(
    TSS2_SYS_CONTEXT             *sysContext,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPMI_ECC_CURVE                curveID,
    TPM2B_ECC_POINT              *Q,
    UINT16                       *counter,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.45 TPM Tss2_Sys_VerifySignature Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_VerifySignature_Prepare(
    TSS2_SYS_CONTEXT     *sysContext,
    TPMI_DH_OBJECT        keyHandle,
    TPM2B_DIGEST   const *digest,
    TPMT_SIGNATURE const *signature);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_VerifySignature_Complete(
    TSS2_SYS_CONTEXT *sysContext,
    TPMT_TK_VERIFIED *validation);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_VerifySignature(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_DH_OBJECT                keyHandle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_DIGEST           const *digest,
    TPMT_SIGNATURE         const *signature,
    TPMT_TK_VERIFIED             *validation,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.46 TPM Tss2_Sys_Sign Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_Sign_Prepare(
    TSS2_SYS_CONTEXT        *sysContext,
    TPMI_DH_OBJECT           keyHandle,
    TPM2B_DIGEST      const *digest,
    TPMT_SIG_SCHEME   const *inScheme,
    TPMT_TK_HASHCHECK const *validation);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_Sign_Complete(
    TSS2_SYS_CONTEXT *sysContext,
    TPMT_SIGNATURE   *signature);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_Sign(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_DH_OBJECT                keyHandle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_DIGEST           const *digest,
    TPMT_SIG_SCHEME        const *inScheme,
    TPMT_TK_HASHCHECK      const *validation,
    TPMT_SIGNATURE               *signature,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.47 TPM Tss2_Sys_SetCommandCodeAuditStatus Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_SetCommandCodeAuditStatus_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
    TPMI_RH_PROVISION auth,
    TPMI_ALG_HASH     auditAlg,
    TPML_CC    const *setList,
    TPML_CC    const *clearList);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_SetCommandCodeAuditStatus_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_SetCommandCodeAuditStatus(
    TSS2_SYS_CONTEXT             *sysContext,
```

```
    TPMI_RH_PROVISION             auth,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPMI_ALG_HASH                 auditAlg,
    TPML_CC               const *setList,
    TPML_CC               const *clearList,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.48 TPM Tss2_Sys_PCR_Extend Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PCR_Extend_Prepare(
    TSS2_SYS_CONTEXT         *sysContext,
    TPMI_DH_PCR               pcrHandle,
    TPML_DIGEST_VALUES const *digests);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PCR_Extend_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PCR_Extend(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_DH_PCR                   pcrHandle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPML_DIGEST_VALUES     const *digests,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.49 TPM Tss2_Sys_PCR_Event Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PCR_Event_Prepare(
    TSS2_SYS_CONTEXT  *sysContext,
    TPMI_DH_PCR        pcrHandle,
    TPM2B_EVENT const *eventData);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PCR_Event_Complete(
    TSS2_SYS_CONTEXT   *sysContext,
    TPML_DIGEST_VALUES *digests);
```

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PCR_Event(
    TSS2_SYS_CONTEXT              *sysContext,
    TPMI_DH_PCR                    pcrHandle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_EVENT             const *eventData,
    TPML_DIGEST_VALUES            *digests,
    TSS2L_SYS_AUTH_RESPONSE       *rspAuths);
```

4.8.50 TPM Tss2_Sys_PCR_Read Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PCR_Read_Prepare(
    TSS2_SYS_CONTEXT         *sysContext,
    TPML_PCR_SELECTION const *pcrSelectionIn);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PCR_Read_Complete(
    TSS2_SYS_CONTEXT   *sysContext,
    UINT32             *pcrUpdateCounter,
    TPML_PCR_SELECTION *pcrSelectionOut,
    TPML_DIGEST        *pcrValues);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PCR_Read(
    TSS2_SYS_CONTEXT              *sysContext,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPML_PCR_SELECTION     const *pcrSelectionIn,
    UINT32                        *pcrUpdateCounter,
    TPML_PCR_SELECTION            *pcrSelectionOut,
    TPML_DIGEST                   *pcrValues,
    TSS2L_SYS_AUTH_RESPONSE       *rspAuths);
```

4.8.51 TPM Tss2_Sys_Allocate Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PCR_Allocate_Prepare(
    TSS2_SYS_CONTEXT         *sysContext,
    TPMI_RH_PLATFORM          authHandle,
    TPML_PCR_SELECTION const *pcrAllocation);
```

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PCR_Allocate_Complete(
    TSS2_SYS_CONTEXT *sysContext,
    TPMI_YES_NO      *allocationSuccess,
    UINT32           *maxPCR,
    UINT32           *sizeNeeded,
    UINT32           *sizeAvailable);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PCR_Allocate(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_RH_PLATFORM              authHandle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPML_PCR_SELECTION     const *pcrAllocation,
    TPMI_YES_NO                  *allocationSuccess,
    UINT32                       *maxPCR,
    UINT32                       *sizeNeeded,
    UINT32                       *sizeAvailable,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.52 TPM Tss2_Sys_SetAuthPolicy Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PCR_SetAuthPolicy_Prepare(
    TSS2_SYS_CONTEXT   *sysContext,
    TPMI_RH_PLATFORM    authHandle,
    TPM2B_DIGEST const *authPolicy,
    TPMI_ALG_HASH       hashAlg,
    TPMI_DH_PCR         pcrNum);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PCR_SetAuthPolicy_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PCR_SetAuthPolicy(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_RH_PLATFORM              authHandle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_DIGEST           const *authPolicy,
```

```
    TPMI_ALG_HASH                hashAlg,
    TPMI_DH_PCR                  pcrNum,
    TSS2L_SYS_AUTH_RESPONSE     *rspAuths);
```

4.8.53 TPM Tss2_Sys_SetAuthValue Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PCR_SetAuthValue_Prepare(
    TSS2_SYS_CONTEXT   *sysContext,
    TPMI_DH_PCR         pcrHandle,
    TPM2B_DIGEST const *auth);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PCR_SetAuthValue_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PCR_SetAuthValue(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_DH_PCR                   pcrHandle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_DIGEST           const *auth,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.54 TPM Tss2_PCR_Reset Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PCR_Reset_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
    TPMI_DH_PCR       pcrHandle);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PCR_Reset_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PCR_Reset(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_DH_PCR                   pcrHandle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```


4.8.55 TPM Tss2_Sys_PolicySigned Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicySigned_Prepare(
    TSS2_SYS_CONTEXT     *sysContext,
    TPMI_DH_OBJECT        authObject,
    TPMI_SH_POLICY        policySession,
    TPM2B_NONCE    const *nonceTPM,
    TPM2B_DIGEST   const *cpHashA,
    TPM2B_NONCE    const *policyRef,
    INT32                 expiration,
    TPMT_SIGNATURE const *auth);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicySigned_Complete(
    TSS2_SYS_CONTEXT *sysContext,
    TPM2B_TIMEOUT    *timeout,
    TPMT_TK_AUTH     *policyTicket);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicySigned(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_DH_OBJECT                authObject,
    TPMI_SH_POLICY                policySession,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_NONCE            const *nonceTPM,
    TPM2B_DIGEST           const *cpHashA,
    TPM2B_NONCE            const *policyRef,
    INT32                         expiration,
    TPMT_SIGNATURE         const *auth,
    TPM2B_TIMEOUT                *timeout,
    TPMT_TK_AUTH                 *policyTicket,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.56 TPM Tss2_Sys_PolicySecret Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicySecret_Prepare(
```

```
    TSS2_SYS_CONTEXT   *sysContext,
    TPMI_DH_ENTITY      authHandle,
    TPMI_SH_POLICY      policySession,
    TPM2B_NONCE  const *nonceTPM,
    TPM2B_DIGEST const *cpHashA,
    TPM2B_NONCE  const *policyRef,
    INT32               expiration);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicySecret_Complete(
    TSS2_SYS_CONTEXT *sysContext,
    TPM2B_TIMEOUT    *timeout,
    TPMT_TK_AUTH     *policyTicket);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicySecret(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_DH_ENTITY                authHandle,
    TPMI_SH_POLICY                policySession,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_NONCE           const *nonceTPM,
    TPM2B_DIGEST          const *cpHashA,
    TPM2B_NONCE           const *policyRef,
    INT32                        expiration,
    TPM2B_TIMEOUT               *timeout,
    TPMT_TK_AUTH                *policyTicket,
    TSS2L_SYS_AUTH_RESPONSE     *rspAuths);
```

4.8.57 TPM Tss2_Sys_PolicyTicket Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyTicket_Prepare(
    TSS2_SYS_CONTEXT    *sysContext,
    TPMI_SH_POLICY       policySession,
    TPM2B_TIMEOUT const *timeout,
    TPM2B_DIGEST  const *cpHashA,
    TPM2B_NONCE   const *policyRef,
    TPM2B_NAME    const *authName,
    TPMT_TK_AUTH  const *ticket);
```

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyTicket_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyTicket(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_SH_POLICY                policySession,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_TIMEOUT          const *timeout,
    TPM2B_DIGEST           const *cpHashA,
    TPM2B_NONCE            const *policyRef,
    TPM2B_NAME             const *authName,
    TPMT_TK_AUTH           const *ticket,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.58 TPM Tss2_Sys_PolicyOR Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyOR_Prepare(
    TSS2_SYS_CONTEXT  *sysContext,
    TPMI_SH_POLICY     policySession,
    TPML_DIGEST const *pHashList);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyOR_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyOR(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_SH_POLICY                policySession,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPML_DIGEST            const *pHashList,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.59 TPM Tss2_Sys_PolicyPCR Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyPCR_Prepare(
```

```
    TSS2_SYS_CONTEXT         *sysContext,
    TPMI_SH_POLICY            policySession,
    TPM2B_DIGEST       const *pcrDigest,
    TPML_PCR_SELECTION const *pcrs);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyPCR_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyPCR(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_SH_POLICY                policySession,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_DIGEST           const *pcrDigest,
    TPML_PCR_SELECTION     const *pcrs,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.60 TPM Tss2_Sys_PolicyLocality Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyLocality_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
    TPMI_SH_POLICY    policySession,
    TPMA_LOCALITY     locality);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyLocality_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyLocality(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_SH_POLICY                policySession,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPMA_LOCALITY                 locality,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.61 TPM Tss2_Sys_PolicyNV Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyNV_Prepare(
    TSS2_SYS_CONTEXT    *sysContext,
    TPMI_RH_NV_AUTH      authHandle,
    TPMI_RH_NV_INDEX     nvIndex,
    TPMI_SH_POLICY       policySession,
    TPM2B_OPERAND const *operandB,
    UINT16               offset,
    TPM2_EO              operation);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyNV_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyNV(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_RH_NV_AUTH               authHandle,
    TPMI_RH_NV_INDEX              nvIndex,
    TPMI_SH_POLICY                policySession,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_OPERAND          const *operandB,
    UINT16                        offset,
    TPM2_EO                       operation,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.62 TPM Tss2_Sys_PolicyCounterTimer Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyCounterTimer_Prepare(
    TSS2_SYS_CONTEXT    *sysContext,
    TPMI_SH_POLICY       policySession,
    TPM2B_OPERAND const *operandB,
    UINT16               offset,
    TPM2_EO              operation);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyCounterTimer_Complete(
    TSS2_SYS_CONTEXT *sysContext);
```

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyCounterTimer(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_SH_POLICY                policySession,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_OPERAND          const *operandB,
    UINT16                        offset,
    TPM2_EO                       operation,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.63 TPM Tss2_Sys_PolicyCommandCode Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyCommandCode_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
    TPMI_SH_POLICY    policySession,
    TPM2_CC           code);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyCommandCode_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyCommandCode(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_SH_POLICY                policySession,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2_CC                       code,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.64 TPM Tss2_Sys_PolicyPhysicalPresence Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyPhysicalPresence_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
    TPMI_SH_POLICY    policySession);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyPhysicalPresence_Complete(
    TSS2_SYS_CONTEXT *sysContext);
```

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyPhysicalPresence(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_SH_POLICY                policySession,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.65 TPM Tss2_Sys_PolicyCpHash Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyCpHash_Prepare(
    TSS2_SYS_CONTEXT   *sysContext,
    TPMI_SH_POLICY      policySession,
    TPM2B_DIGEST const *cpHashA);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyCpHash_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyCpHash(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_SH_POLICY                policySession,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_DIGEST           const *cpHashA,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.66 TPM Tss2_Sys_PolicyNameHash Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyNameHash_Prepare(
    TSS2_SYS_CONTEXT   *sysContext,
    TPMI_SH_POLICY      policySession,
    TPM2B_DIGEST const *nameHash);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyNameHash_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyNameHash(
    TSS2_SYS_CONTEXT             *sysContext,
```

```
    TPMI_SH_POLICY                policySession,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_DIGEST           const *nameHash,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.67 TPM Tss2_Sys_PolicyDuplicationSelect Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyDuplicationSelect_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
    TPMI_SH_POLICY    policySession,
    TPM2B_NAME const *objectName,
    TPM2B_NAME const *newParentName,
    TPMI_YES_NO       includeObject);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyDuplicationSelect_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyDuplicationSelect(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_SH_POLICY                policySession,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_NAME             const *objectName,
    TPM2B_NAME             const *newParentName,
    TPMI_YES_NO                   includeObject,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.68 TPM Tss2_Sys_PolicyAuthorize Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyAuthorize_Prepare(
    TSS2_SYS_CONTEXT       *sysContext,
    TPMI_SH_POLICY          policySession,
    TPM2B_DIGEST     const *approvedPolicy,
    TPM2B_NONCE      const *policyRef,
    TPM2B_NAME       const *keySign,
    TPMT_TK_VERIFIED const *checkTicket);
```

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyAuthorize_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyAuthorize(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_SH_POLICY                policySession,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_DIGEST           const *approvedPolicy,
    TPM2B_NONCE            const *policyRef,
    TPM2B_NAME             const *keySign,
    TPMT_TK_VERIFIED       const *checkTicket,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.69 TPM Tss2_Sys_PolicyAuthValue Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyAuthValue_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
    TPMI_SH_POLICY    policySession);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyAuthValue_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyAuthValue(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_SH_POLICY                policySession,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.70 TPM Tss2_Sys_PolicyPassword Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyPassword_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
    TPMI_SH_POLICY    policySession);
```

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyPassword_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyPassword(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_SH_POLICY                policySession,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.71 TPM Tss2_Sys_PolicyGetDigest Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyGetDigest_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
    TPMI_SH_POLICY    policySession);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyGetDigest_Complete(
    TSS2_SYS_CONTEXT *sysContext,
    TPM2B_DIGEST     *policyDigest);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyGetDigest(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_SH_POLICY                policySession,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_DIGEST                 *policyDigest,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.72 TPM Tss2_Sys_PolicyNvWritten Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyNvWritten_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
    TPMI_SH_POLICY    policySession,
    TPMI_YES_NO       writtenSet);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyNvWritten_Complete(
    TSS2_SYS_CONTEXT *sysContext);
```

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyNvWritten(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_SH_POLICY                policySession,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPMI_YES_NO                   writtenSet,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.73 TPM Tss2_Sys_PolicyTemplate Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyTemplate_Prepare(
    TSS2_SYS_CONTEXT   *sysContext,
    TPMI_SH_POLICY      policySession,
    TPM2B_DIGEST const *templateHash);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyTemplate_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyTemplate(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_SH_POLICY                policySession,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_DIGEST           const *templateHash,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.74 TPM Tss2_Sys_PolicyAuthorizeNV Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyAuthorizeNV_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
    TPMI_RH_NV_AUTH   authHandle,
    TPMI_RH_NV_INDEX  nvIndex,
    TPMI_SH_POLICY    policySession);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyAuthorizeNV_Complete(
    TSS2_SYS_CONTEXT *sysContext);
```

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PolicyAuthorizeNV(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_RH_NV_AUTH               authHandle,
    TPMI_RH_NV_INDEX              nvIndex,
    TPMI_SH_POLICY                policySession,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.75 TPM Tss2_Sys_CreatePrimary Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_CreatePrimary_Prepare(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_RH_HIERARCHY             primaryHandle,
    TPM2B_SENSITIVE_CREATE const *inSensitive,
    TPM2B_PUBLIC           const *inPublic,
    TPM2B_DATA             const *outsideInfo,
    TPML_PCR_SELECTION     const *creationPCR);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_CreatePrimary_Complete(
    TSS2_SYS_CONTEXT    *sysContext,
    TPM2_HANDLE         *objectHandle,
    TPM2B_PUBLIC        *outPublic,
    TPM2B_CREATION_DATA *creationData,
    TPM2B_DIGEST        *creationHash,
    TPMT_TK_CREATION    *creationTicket,
    TPM2B_NAME          *name);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_CreatePrimary(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_RH_HIERARCHY             primaryHandle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_SENSITIVE_CREATE const *inSensitive,
    TPM2B_PUBLIC           const *inPublic,
    TPM2B_DATA             const *outsideInfo,
    TPML_PCR_SELECTION     const *creationPCR,
```

```
    TPM2_HANDLE                 *objectHandle,
    TPM2B_PUBLIC                *outPublic,
    TPM2B_CREATION_DATA         *creationData,
    TPM2B_DIGEST                *creationHash,
    TPMT_TK_CREATION            *creationTicket,
    TPM2B_NAME                  *name,
    TSS2L_SYS_AUTH_RESPONSE     *rspAuths);
```

4.8.76 TPM Tss2_Sys_HierarchyControl Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_HierarchyControl_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
    TPMI_RH_HIERARCHY authHandle,
    TPMI_RH_ENABLES   enable,
    TPMI_YES_NO       state);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_HierarchyControl_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_HierarchyControl(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_RH_HIERARCHY             authHandle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPMI_RH_ENABLES               enable,
    TPMI_YES_NO                   state,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.77 TPM Tss2_Sys_SetPrimaryPolicy Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_SetPrimaryPolicy_Prepare(
    TSS2_SYS_CONTEXT      *sysContext,
    TPMI_RH_HIERARCHY_AUTH authHandle,
    TPM2B_DIGEST    const *authPolicy,
    TPMI_ALG_HASH          hashAlg);
```

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_SetPrimaryPolicy_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_SetPrimaryPolicy(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_RH_HIERARCHY_AUTH        authHandle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_DIGEST           const *authPolicy,
    TPMI_ALG_HASH                 hashAlg,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.78 TPM Tss2_Sys_ChangePPS Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_ChangePPS_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
    TPMI_RH_PLATFORM  authHandle);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_ChangePPS_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_ChangePPS(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_RH_PLATFORM              authHandle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.79 TPM Tss2_Sys_ChangeEPS Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_ChangeEPS_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
    TPMI_RH_PLATFORM  authHandle);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_ChangeEPS_Complete(
    TSS2_SYS_CONTEXT *sysContext);
```



```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_ChangeEPS(
    TSS2_SYS_CONTEXT              *sysContext,
    TPMI_RH_PLATFORM               authHandle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.80 TPM Tss2_Sys_Clear Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_Clear_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
    TPMI_RH_CLEAR     authHandle);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_Clear_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_Clear(
    TSS2_SYS_CONTEXT              *sysContext,
    TPMI_RH_CLEAR                  authHandle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.81 TPM Tss2_Sys_ClearControl Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_ClearControl_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
    TPMI_RH_CLEAR     auth,
    TPMI_YES_NO       disable);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_ClearControl_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_ClearControl(
    TSS2_SYS_CONTEXT              *sysContext,
    TPMI_RH_CLEAR                  auth,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
```

```
    TPMI_YES_NO                  disable,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.82 TPM Tss2_Sys_HierarchyChangeAuth Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_HierarchyChangeAuth_Prepare(
    TSS2_SYS_CONTEXT      *sysContext,
    TPMI_RH_HIERARCHY_AUTH authHandle,
    TPM2B_AUTH      const *newAuth);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_HierarchyChangeAuth_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_HierarchyChangeAuth(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_RH_HIERARCHY_AUTH        authHandle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_AUTH             const *newAuth,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.83 TPM Tss2_Sys_DictionaryAttackLockReset Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_DictionaryAttackLockReset_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
    TPMI_RH_LOCKOUT   lockHandle);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_DictionaryAttackLockReset_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_DictionaryAttackLockReset(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_RH_LOCKOUT               lockHandle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.84 TPM Tss2_Sys_DictionaryAttackParameters Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_DictionaryAttackParameters_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
    TPMI_RH_LOCKOUT   lockHandle,
    UINT32            newMaxTries,
    UINT32            newRecoveryTime,
    UINT32            lockoutRecovery);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_DictionaryAttackParameters_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_DictionaryAttackParameters(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_RH_LOCKOUT               lockHandle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    UINT32                        newMaxTries,
    UINT32                        newRecoveryTime,
    UINT32                        lockoutRecovery,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.85 TPM Tss2_Sys_PP_Commands Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PP_Commands_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
    TPMI_RH_PLATFORM  auth,
    TPML_CC    const *setList,
    TPML_CC    const *clearList);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PP_Commands_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_PP_Commands(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_RH_PLATFORM              auth,
```

```
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPML_CC                const *setList,
    TPML_CC                const *clearList,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.86 TPM Tss2_Sys_SetAlgorithmSet Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_SetAlgorithmSet_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
    TPMI_RH_PLATFORM  authHandle,
    UINT32            algorithmSet);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_SetAlgorithmSet_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_SetAlgorithmSet(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_RH_PLATFORM              authHandle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    UINT32                        algorithmSet,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.87 TPM Tss2_Sys_FieldUpgradeStart Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_FieldUpgradeStart_Prepare(
    TSS2_SYS_CONTEXT     *sysContext,
    TPMI_RH_PLATFORM      authorization,
    TPMI_DH_OBJECT        keyHandle,
    TPM2B_DIGEST   const *fuDigest,
    TPMT_SIGNATURE const *manifestSignature);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_FieldUpgradeStart_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_FieldUpgradeStart(
```

```
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_RH_PLATFORM              authorization,
    TPMI_DH_OBJECT                keyHandle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_DIGEST           const *fuDigest,
    TPMT_SIGNATURE         const *manifestSignature,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.88 TPM Tss2_Sys_FieldUpgradeData Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_FieldUpgradeData_Prepare(
    TSS2_SYS_CONTEXT       *sysContext,
    TPM2B_MAX_BUFFER const *fuData);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_FieldUpgradeData_Complete(
    TSS2_SYS_CONTEXT *sysContext,
    TPMT_HA          *nextDigest,
    TPMT_HA          *firstDigest);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_FieldUpgradeData(
    TSS2_SYS_CONTEXT             *sysContext,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_MAX_BUFFER       const *fuData,
    TPMT_HA                      *nextDigest,
    TPMT_HA                      *firstDigest,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.89 TPM Tss2_Sys_FirmwareRead Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_FirmwareRead_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
    UINT32            sequenceNumber);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_FirmwareRead_Complete(
    TSS2_SYS_CONTEXT *sysContext,
```

```
    TPM2B_MAX_BUFFER *fuData);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_FirmwareRead(
    TSS2_SYS_CONTEXT             *sysContext,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    UINT32                        sequenceNumber,
    TPM2B_MAX_BUFFER             *fuData,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.90 TPM Tss2_Sys_ContextSave Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_ContextSave_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
    TPMI_DH_CONTEXT   saveHandle);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_ContextSave_Complete(
    TSS2_SYS_CONTEXT *sysContext,
    TPMS_CONTEXT     *context);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_ContextSave(
    TSS2_SYS_CONTEXT *sysContext,
    TPMI_DH_CONTEXT   saveHandle,
    TPMS_CONTEXT     *context);
```

4.8.91 TPM Tss2_Sys_ContextLoad Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_ContextLoad_Prepare(
    TSS2_SYS_CONTEXT   *sysContext,
    TPMS_CONTEXT const *context);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_ContextLoad_Complete(
    TSS2_SYS_CONTEXT *sysContext,
    TPMI_DH_CONTEXT  *loadedHandle);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_ContextLoad(
```

```
    TSS2_SYS_CONTEXT   *sysContext,
    TPMS_CONTEXT const *context,
    TPMI_DH_CONTEXT    *loadedHandle);
```

4.8.92 TPM Tss2_Sys_FlushContext Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_FlushContext_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
    TPMI_DH_CONTEXT   flushHandle);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_FlushContext_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_FlushContext(
    TSS2_SYS_CONTEXT *sysContext,
    TPMI_DH_CONTEXT   flushHandle);
```

4.8.93 TPM Tss2_Sys_EvictControl Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_EvictControl_Prepare(
    TSS2_SYS_CONTEXT  *sysContext,
    TPMI_RH_PROVISION  auth,
    TPMI_DH_OBJECT     objectHandle,
    TPMI_DH_PERSISTENT persistentHandle);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_EvictControl_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_EvictControl(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_RH_PROVISION             auth,
    TPMI_DH_OBJECT                objectHandle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPMI_DH_PERSISTENT            persistentHandle,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.94 TPM Tss2_Sys_ReadClock Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_ReadClock_Prepare(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_ReadClock_Complete(
    TSS2_SYS_CONTEXT *sysContext,
    TPMS_TIME_INFO   *currentTime);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_ReadClock(
    TSS2_SYS_CONTEXT             *sysContext,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPMS_TIME_INFO               *currentTime,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.95 TPM Tss2_Sys_ClockSet Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_ClockSet_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
    TPMI_RH_PROVISION auth,
    UINT64            newTime);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_ClockSet_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_ClockSet(
    TSS2_SYS_CONTEXT             *sysContext,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPMI_RH_PROVISION             auth,
    UINT64                        newTime,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.96 TPM Tss2_Sys_ClockRateAdjust Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_ClockRateAdjust_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
    TPMI_RH_PROVISION auth,
    TPM2_CLOCK_ADJUST rateAdjust);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_ClockRateAdjust_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_ClockRateAdjust(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_RH_PROVISION             auth,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2_CLOCK_ADJUST             rateAdjust,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.97 TPM Tss2_Sys_GetCapability Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_GetCapability_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
    TPM2_CAP          capability,
    UINT32            property,
    UINT32            propertyCount);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_GetCapability_Complete(
    TSS2_SYS_CONTEXT     *sysContext,
    TPMI_YES_NO          *moreData,
    TPMS_CAPABILITY_DATA *capabilityData);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_GetCapability(
    TSS2_SYS_CONTEXT             *sysContext,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2_CAP                      capability,
    UINT32                        property,
    UINT32                        propertyCount,
    TPMI_YES_NO                  *moreData,
```

```
    TPMS_CAPABILITY_DATA         *capabilityData,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.98 TPM Tss2_Sys_TestParms Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_TestParms_Prepare(
    TSS2_SYS_CONTEXT        *sysContext,
    TPMT_PUBLIC_PARMS const *parameters);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_TestParms_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_TestParms(
    TSS2_SYS_CONTEXT             *sysContext,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPMT_PUBLIC_PARMS      const *parameters,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.99 TPM Tss2_Sys_NV_DefineSpace Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_NV_DefineSpace_Prepare(
    TSS2_SYS_CONTEXT      *sysContext,
    TPMI_RH_PROVISION      authHandle,
    TPM2B_AUTH      const *auth,
    TPM2B_NV_PUBLIC const *publicInfo);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_NV_DefineSpace_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_NV_DefineSpace(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_RH_PROVISION             authHandle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_AUTH             const *auth,
    TPM2B_NV_PUBLIC        const *publicInfo,
```

```
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.100 TPM Tss2_Sys_UndefineSpace Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_NV_UndefineSpace_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
    TPMI_RH_PROVISION authHandle,
    TPMI_RH_NV_INDEX  nvIndex);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_NV_UndefineSpace_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_NV_UndefineSpace(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_RH_PROVISION             authHandle,
    TPMI_RH_NV_INDEX              nvIndex,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.101 TPM Tss2_Sys_UndefineSpaceSpecial Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_NV_UndefineSpaceSpecial_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
    TPMI_RH_NV_INDEX  nvIndex,
    TPMI_RH_PLATFORM  platform);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_NV_UndefineSpaceSpecial_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_NV_UndefineSpaceSpecial(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_RH_NV_INDEX              nvIndex,
    TPMI_RH_PLATFORM              platform,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.102 TPM Tss2_Sys_ReadPublic Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_NV_ReadPublic_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
    TPMI_RH_NV_INDEX  nvIndex);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_NV_ReadPublic_Complete(
    TSS2_SYS_CONTEXT *sysContext,
    TPM2B_NV_PUBLIC  *nvPublic,
    TPM2B_NAME       *nvName);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_NV_ReadPublic(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_RH_NV_INDEX              nvIndex,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_NV_PUBLIC              *nvPublic,
    TPM2B_NAME                   *nvName,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.103 TPM Tss2_Sys_NV_Write Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_NV_Write_Prepare(
    TSS2_SYS_CONTEXT          *sysContext,
    TPMI_RH_NV_AUTH            authHandle,
    TPMI_RH_NV_INDEX           nvIndex,
    TPM2B_MAX_NV_BUFFER const *data,
    UINT16                     offset);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_NV_Write_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_NV_Write(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_RH_NV_AUTH               authHandle,
```

```
    TPMI_RH_NV_INDEX              nvIndex,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_MAX_NV_BUFFER    const *data,
    UINT16                        offset,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.104 TPM Tss2_Sys_NV_Increment Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_NV_Increment_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
    TPMI_RH_NV_AUTH   authHandle,
    TPMI_RH_NV_INDEX  nvIndex);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_NV_Increment_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_NV_Increment(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_RH_NV_AUTH               authHandle,
    TPMI_RH_NV_INDEX              nvIndex,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.105 TPM Tss2_Sys_NV_Extend Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_NV_Extend_Prepare(
    TSS2_SYS_CONTEXT          *sysContext,
    TPMI_RH_NV_AUTH            authHandle,
    TPMI_RH_NV_INDEX           nvIndex,
    TPM2B_MAX_NV_BUFFER const *data);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_NV_Extend_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_NV_Extend(
```

```
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_RH_NV_AUTH               authHandle,
    TPMI_RH_NV_INDEX              nvIndex,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_MAX_NV_BUFFER    const *data,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.106 TPM Tss2_Sys_NV_SetBits Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_NV_SetBits_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
    TPMI_RH_NV_AUTH   authHandle,
    TPMI_RH_NV_INDEX  nvIndex,
    UINT64            bits);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_NV_SetBits_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_NV_SetBits(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_RH_NV_AUTH               authHandle,
    TPMI_RH_NV_INDEX              nvIndex,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    UINT64                        bits,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.107 TPM Tss2_Sys_WriteLock Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_NV_WriteLock_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
    TPMI_RH_NV_AUTH   authHandle,
    TPMI_RH_NV_INDEX  nvIndex);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_NV_WriteLock_Complete(
    TSS2_SYS_CONTEXT *sysContext);
```

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_NV_WriteLock(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_RH_NV_AUTH               authHandle,
    TPMI_RH_NV_INDEX              nvIndex,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.108 TPM Tss2_Sys_NV_GlobalWriteLock Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_NV_GlobalWriteLock_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
    TPMI_RH_PROVISION authHandle);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_NV_GlobalWriteLock_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_NV_GlobalWriteLock(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_RH_PROVISION             authHandle,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.109 TPM Tss2_Sys_NV_Read Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_NV_Read_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
    TPMI_RH_NV_AUTH   authHandle,
    TPMI_RH_NV_INDEX  nvIndex,
    UINT16            size,
    UINT16            offset);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_NV_Read_Complete(
    TSS2_SYS_CONTEXT    *sysContext,
    TPM2B_MAX_NV_BUFFER *data);
```

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_NV_Read(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_RH_NV_AUTH               authHandle,
    TPMI_RH_NV_INDEX              nvIndex,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    UINT16                        size,
    UINT16                        offset,
    TPM2B_MAX_NV_BUFFER          *data,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.110 TPM Tss2_Sys_NV_ReadLock Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_NV_ReadLock_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
    TPMI_RH_NV_AUTH   authHandle,
    TPMI_RH_NV_INDEX  nvIndex);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_NV_ReadLock_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_NV_ReadLock(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_RH_NV_AUTH               authHandle,
    TPMI_RH_NV_INDEX              nvIndex,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.111 TPM Tss2_Sys_NV_ChangeAuth Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_NV_ChangeAuth_Prepare(
    TSS2_SYS_CONTEXT *sysContext,
    TPMI_RH_NV_INDEX  nvIndex,
    TPM2B_AUTH const *newAuth);
```

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_NV_ChangeAuth_Complete(
    TSS2_SYS_CONTEXT *sysContext);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_NV_ChangeAuth(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_RH_NV_INDEX              nvIndex,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_AUTH             const *newAuth,
    TSS2L_SYS_AUTH_RESPONSE      *rspAuths);
```

4.8.112 TPM Tss2_Sys_NV_Certify Commands

```
TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_NV_Certify_Prepare(
    TSS2_SYS_CONTEXT      *sysContext,
    TPMI_DH_OBJECT         signHandle,
    TPMI_RH_NV_AUTH        authHandle,
    TPMI_RH_NV_INDEX       nvIndex,
    TPM2B_DATA      const *qualifyingData,
    TPMT_SIG_SCHEME const *inScheme,
    UINT16                 size,
    UINT16                 offset);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_NV_Certify_Complete(
    TSS2_SYS_CONTEXT *sysContext,
    TPM2B_ATTEST     *certifyInfo,
    TPMT_SIGNATURE   *signature);

TSS2_DLL_EXPORT TSS2_RC Tss2_Sys_NV_Certify(
    TSS2_SYS_CONTEXT             *sysContext,
    TPMI_DH_OBJECT                signHandle,
    TPMI_RH_NV_AUTH               authHandle,
    TPMI_RH_NV_INDEX              nvIndex,
    TSS2L_SYS_AUTH_COMMAND const *cmdAuths,
    TPM2B_DATA             const *qualifyingData,
    TPMT_SIG_SCHEME        const *inScheme,
    UINT16                        size,
```

```
    UINT16                       offset,
    TPM2B_ATTEST                *certifyInfo,
    TPMT_SIGNATURE              *signature,
    TSS2L_SYS_AUTH_RESPONSE     *rspAuths);
```

4.9 tss2_sys.h Postlude

```
#ifdef __cplusplus
} /* end extern "C" */
#endif

#endif /* TSS2_SYS_H */
```

5 Appendices

The following are all informative.

5.1 SAPI ABI Negotiation Pseudo Code

```
// A pseudo-code example for how to use this */
applicationExample()
{
       /* first initialize the TCTi-Layer (simplified) */
       Tss2_Tcti_DefaultInitialize(tctContext, &tct_size);

       sysContext = alloc(sys_size = tss2_sys_getContextSize(-1));
       // then initialize the SystemAPI; get the
       // values for abi-family and abi-level from
       // the tss2_sysapi.h file
       TSS2_ABI_VERSION currentAbi = TSS2_ABI_CURRENT_VERSION;
       /* Alternatively but not recommended
       TSS2_ABI_VERSION currentAbi =
       {
              .family = TSS2_ABI_FAMILY,
              .level = TSS2_ABI_LEVEL,
              ...
       };
       */
       ret = Tss2_Sys_Initialize(sysContext, sys_size, tctContext,
                     &currentAbi);

       if (ret == TSS2_ERROR_SYS_ABI_MISMATCH) {
              fprintf(stderr, "ERROR: Mismatch between application's \
ABI-Version and systemAPI's ABI-Version.\n");
              fprintf(stderr, "SystemAPI supports %d.%d.%d.%d\n",
                            currentAbi.creator, currentAbi.family,
                            currentAbi.level, currentAbi.version);
              exit (1);
       } else if (ret != TSS2_SUCCESS) {
              //Handle other errors
       }

       //Use the sysContext} etc etc etc...
```