TRUSTED® COMPUTING GROUP

SPECIFICATION

TCG Feature API (FAPI) Specification

Version 0.94
Revision 04
September 25, 2019

Contact: admin@trustedcomputinggroup.org

Work in Progress

This document is an intermediate draft for comment only and is subject to change without notice. Readers should not design products based on this document.

PUBLIC REVIEW

DISCLAIMERS, NOTICES, AND LICENSE TERMS

NORMATIVE-INFORMATIVE LANGUAGE

The key words “MUST,” “MUST NOT,” “REQUIRED,” “SHALL,” “SHALL NOT,” “SHOULD,” “SHOULD NOT,” “RECOMMENDED,” “MAY,” and “OPTIONAL” in this document’s normative statements are to be interpreted as described in RFC-2119, Key words for use in RFCs to Indicate Requirement Levels.

STATEMENT TYPE

Please note a very important distinction between different sections of text throughout this document. There are two distinctive kinds of text: informative comment and normative statements. Because most of the text in this specification will be of the kind normative statements, the authors have informally defined it as the default and, as such, have specifically called out text of the kind informative comment. They have done this by flagging the beginning and end of each informative comment and highlighting its text in gray. This means that unless text is specifically marked as of the kind informative comment, it can be considered a kind of normative statements.

EXAMPLE: Start of informative comment

This is the first paragraph of 1–n paragraphs containing text of the kind *informative comment* ...

This is the second paragraph of text of the kind *informative comment* ...

This is the nth paragraph of text of the kind *informative comment* ...

To understand the TCG specification the user must read the specification. (This use of MUST does not require any action).

End of informative comment

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	Security Innovation (OnBoard Security)
Dave Challener	Johns Hopkins University
Mike Cox	Security Innovation (OnBoard Security)
Paul England	Microsoft
Andreas Fuchs	Fraunhofer SIT
Ken Goldman	IBM
Pauline Homolle	Infineon Technologies AG
Peter Huewe	Infineon Technologies AG
James Nguyen	Wave
Jürgen Repp	Fraunhofer SIT
William Roberts	Intel
Philip Tricca	Intel
Lee Wilson	Security Innovation (OnBoard Security)

CHANGE HISTORY

REVISION	DATE	DESCRIPTION
0.92r01	June 28, 2019	• Initial Rewrite
0.92r04	August 09, 2019	• Many minor changes to parameter descriptions and Error codes • New functions CreateSeal, CreateKey, CreateNV, Get/SetAppData • Use of sized array for PCR lists in Quote
0.94r01	August 13, 2019	• Incorporated feedback from TSS-WG and Graeme
0.94r02	September 24, 2019	• Incorporated feedback from TC
0.94r03	September 24,2019	• Incorporated feedback form TSS-WG call
0.94r04	September 25, 2019	• Inserted table captions • Incorporated feedback from TC

CONTENTS

1 Information and Document Scope

1.1 Scope of this Specification

This specification defines an application programming interface (API) for interacting with a TPM 2.0 [11] on an abstract level.

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 Structure

At the time of writing, the documents that specify 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 (MU) 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 TSS Response Code (RC) API Specification

[9] TCG TSS 2.0 JSON Data Type and Policy Language Specification (Draft)

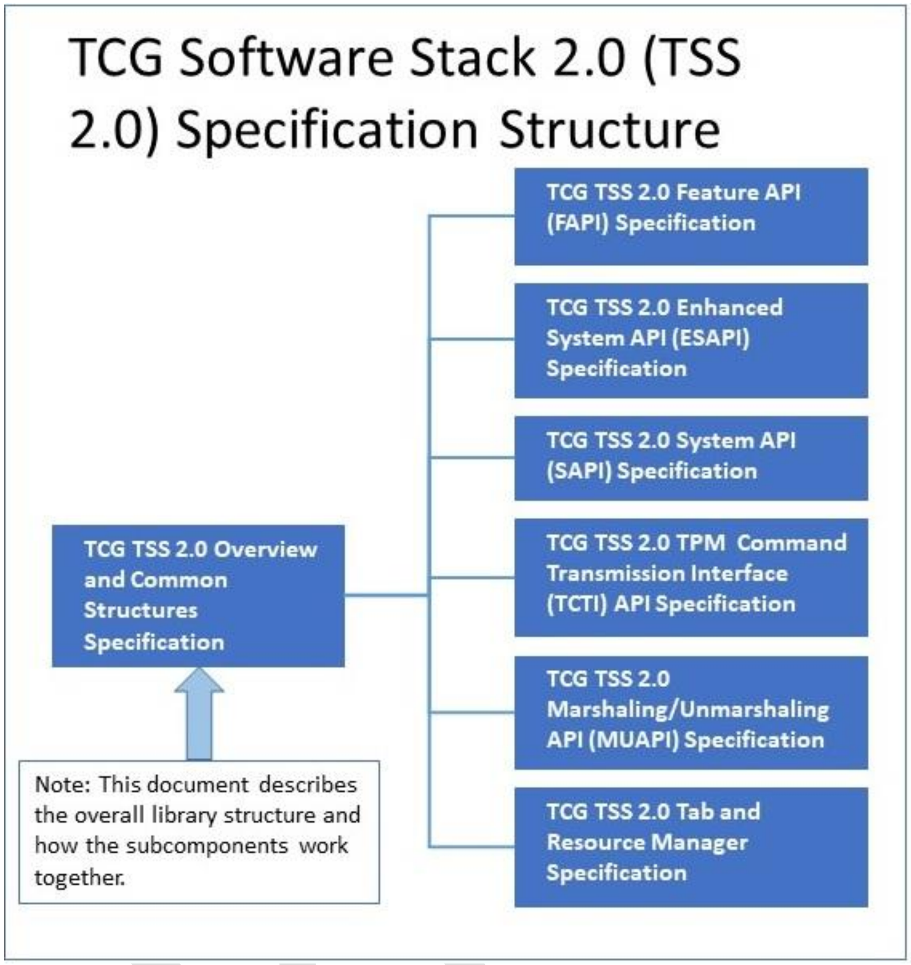
TCG Software Stack 2.0 (TSS 2.0) Specification Structure
TCG TSS 2.0 Feature API (FAPI) Specification
TCG TSS 2.0 Enhanced System API (ESAPI) Specification
TCG TSS 2.0 System API (SAPI) Specification
TCG TSS 2.0 TPM Command Transmission Interface (TCTI) API Specification
TCG TSS 2.0 Marshaling/Unmarshaling API (MUAPI) Specification
TCG TSS 2.0 Tab and Resource Manager Specification
TCG TSS 2.0 Overview and Common Structures Specification
Note: This document describes the overall library structure and how the subcomponents work together.

Figure 1: TSS 2.0 Specification Structure

1.4 References

Most references for the TSS 2.0 specifications are provided in the TCG TSS 2.0 Overview and Common Structures Specification [1].

The following additional references are used by this specification. The numbering continues from section 1.3:

[10] ISO C99 standard

[11] Trusted Platform Module Library Part 1: Architecture Family “2.0”

[12] Trusted Platform Module Library Part 2: Structures Family “2.0”

[13] TCG Registry of reserved TPM 2.0 handles and localities Version 1.0 Revision 1

2 Introduction

Start of informative comment

This TSS 2.0 Feature API is meant to be a very high-level API, aimed at providing 80% of programmers who write a program using the TPM with everything they require. The remaining 20% of programmers will have to supplement this set of APIs with the Extended System API (ESAPI) [5] or System API (SAPI) [4].

This specification is intended to make programming with the TPM as simple as possible – but no simpler. The cognitive load for a new programmer using this API is meant to be as low as possible. Because of this, a number of design considerations have been made, including:

- Cryptographic profiles determine the cryptographic algorithms and parameters for all keys and operations of a specific TPM interaction. One of these profiles is deemed the default profile of the platform and used if the application does not specify a profile for an operation.
- Key types with a reduced set of attributes; i.e. not all attributes that the TPM supports are exposed to the application.
- Objects associated with either an authentication value (password) or with a policy written in JSON. Upon access to an object these policies are executed automatically.
    - A tool for editing these JSON policies is not included
    - Callbacks to the application are performed whenever necessary during policy evaluation, e.g. when deciding which OR branch of a policy to follow.
- All communication with the TPM is performed in salted HMAC sessions and parameter encryption is enabled wherever applicable.
- A key and policy metadata store stores all necessary data on the platform’s storage medium.
- PCR logs are supported in all operations, including Attestation generation and verification.
- The host platform’s TPM is the default TPM for all FAPI interactions and the key and policy store are stored on the local filesystem.

End of informative comment

All prototype definitions, function parameters and return values in this document are REQUIRED.

2.1 Asynchronous invocation model

Start of informative comment

All functionality in the Feature API that requires I/O operations, e.g. TPM, disk, network, are provided via three functions. The synchronous version, without a suffix, will block until it is finished. Functions with the suffix _Async initiate asynchronous execution. Functions with the suffix _Finish test whether asynchronous execution is finished and return either the result or TSS2_FAPI_RC_TRY_AGAIN. A FAPI function will most likely return TSS2_FAPI_RC_TRY_AGAIN multiple times independent of the time passed, if multiple I/O steps need to be executed, since the FAPI implementation will transition from one I/O state to the next I/O state internally. After invoking an _Async function, a user is expected to call the corresponding _Finish function repeatedly as long as the _Finish function returns TSS2_FAPI_RC_TRY_AGAIN

End of informative comment

Functions with the suffixes _Async or _Finish SHALL return promptly without waiting for I/O operations to complete. Their symmetric counterpart (i.e. without any suffix) may wait for I/O operations to complete.

Some functions do not have an _Async and _Finish version but only a single function prototype. These functions SHALL return promptly without waiting for I/O operations to complete.

.

3 Structures and data types

Start of informative comment

This specification contains almost no data type definitions.

Object references are encoded as path-like strings (since they are persistent). Serialized objects are encoded as PEM or as JSON.

End of informative comment

3.1 FAPI_CONTEXT

All functions of FAPI include a FAPI_CONTEXT as the first parameter (except for FAPI_Free()). All internal state information of FAPI SHALL be stored inside this context object in order to allow several independent contexts to operate in parallel within the same process environment.

```
typedef struct _FAPI_CONTEXT FAPI_CONTEXT;
```

3.2 Entity paths

Keys, NV indices and policies are all referenced via a path. The separator between nodes in entity paths is "/". A leading "/" is optional, since no relative paths are supported. All paths SHALL follow the schemes defined in this section.

The elements inside a path are composed of the characters A-Z, a-z, 0-9, _ and -.

3.2.1 Key paths

A keyPath SHALL start with either:

- `<cryptoprofile>` followed by the next item in the list. The <cryptoprofile> always starts with "`P_`" followed by the name of the profile. Examples are P_ECCP256 or P_RSASHA1. If no <cryptoprofile> is provided then the "default profile" SHALL be used. The default profile is determined by an implementation specific configuration mechanism not defined in this specification. Examples may be Registry entries or configuration files.
- `<hierarchy>` followed by a primary object. The hierarchy SHALL be one of the following values: HE (for endorsement), HP (for platform), HS (for storage) or HN (for null hierarchy). The hierarchy MAY be omitted by the caller if one of the following primary objects is referenced: SRK (Storage Primary Key; implies HS), EK (Endorsement Key; implies HE).
- "/ext":A special hierarchy that contains public keys of remote TPMs that are used in Fapi_ExportKey(), for example.

Such a keypath start is followed by a sequence of key objects. The first key object is a primary key. If any key is a restricted storage/decrypt key then it may have child keys, otherwise it cannot have child keys.

Start of informative comment

By convention, most keys are anticipated to be children of the SRK.

By convention, a vendor or application is expected to use the form [<vendor>-]<software>-<keyname>

Examples: /SRK/tcg-fapi-attestationkey, /SRK/fapi-attestationkey

End of informative comment

3.2.2 NV paths

An nvPath is composed of three elements, separated by "/".

An nvPath SHALL start with "/nv".

The second path element SHALL identify the NV handle range for the nv object (consistent with the TCG Registry of reserved TPM 2.0 handles and localities [13]). At the time of writing this includes the following values: TPM, Platform, Owner, Endorsement_Certificate, Platform_Certificate, Component_OEM, TPM_OEM, Platform_OEM, PC-Client, Server, Virtualized_Platform, MPWG, Embedded.

The third path element SHALL identify the actual NV-Index using a meaningful name.

Example: /nv/Endorsement_Certificate/EK_Certificate

3.2.3 Policy Paths

A policyPath consists of two elements, separated by "/".

A policyPath SHALL start with "/policy".

The second path element SHALL identify the policy or policy template using a meaningful name.

3.3 Object types

An object type is used during entity creation and consists of a list of comma and/or space separated keywords. If a keyword is not present the inverse of the reference TPM attribute bits SHALL be set or cleared. These keywords are:

- sign: Sets the sign attribute of a key.
- decrypt: Sets the decrypt attribute of a key.
  If neither sign nor decrypt are provided, both attributes SHALL be set.
- restricted: Sets the restricted attribute of a key.
  If restricted is set, either sign or decrypt (but not both) SHALL be set.
- exportable: Clears the fixedTPM and fixedParent attributes of a key or sealed object.
- noda: Sets the noda attribute of a key or NV index.
- bitfield: Sets the NV type to bitfield.
- counter: Sets the NV type to counter.
- pcr: Sets the NV type to pcr-like behavior.
  If none of the previous three keywords is provided a regular NV index SHALL be created.
- system: Stores the data blobs and metadata for a created key or seal in the system-wide directory instead of user's personal directory.
- A hexadecimal number: Marks a key object to be made persistent and sets the persistent object handle to this value.

If a policy is provided during creation of a key, seal or NV index, then the userWithAuth flag SHALL be CLEAR. If no policy is provided during creation, then the userWithAuth flag SHALL be SET.

3.4 Cryptographic profiles

The cryptographic profiles are configured in an implementation specific way. The values affected by these profiles SHALL be:

- Name hash algorithm
- Asymmetric signature algorithm, scheme and parameters (such as curve, keysize, default padding, hash, etc)
- Asymmetric decryption algorithm, scheme and parameters (such as curve, keysize, etc)
- PCR bank selection (which PCR banks shall be extended, quoted and read)

3.5 Policies and policy templates encoding

Policies and policy templates throughout this specification SHALL be encoded in the JSON format defined in TCG TSS 2.0 JSON Policy Specification [9].

3.6 Exported key encoding

The exported key data is defined in Table 1.

Table 1 TPMS_EXPORTEDKEY

TYPE	FIELD	DESCRIPTION
TPM2B_PRIVATE	duplicate	The encrypted duplicate of the private portion of the key.
TPM2B_ENCRYPTED_SECRET	encrypted_seed	The encrypted seed required for importing.
TPM2B_PUBLIC	public	The public portion of the exported key.
TPM2B_PUBLIC	parent_public	The public area of the new parent object.
string	certificate	The PEM encoded certificate of the exported key.
TPMS_POLICY	policy	The JSON encoded policy of the exported key.

The data type in Table 1 SHALL be encoded according to the JSON format description defined in TCG TSS 2.0 JSON Policy Specification.

Start of informative comment

Basic JSON field:

```
{
    "duplicate": …,
    "encrypted_seed": …,
    "public": …,
    "public_parent": …,
    "certificate: …,
    "policy": …
}
```

End of informative comment

3.7 PCR event log encoding

PCR event logs are a list (arbitrary length JSON array) of log entries. These entries are defined in Table 2.

Table 2 TPMS_EVENTLOGENTRY

TYPE	FIELD	DESCRIPTION
UINT32	recnum	Unique record number
UINT32	pcr	PCR index
TPML_DIGEST_VALUES	digest	The digests
string	type	The type of event. Possible values: • "LINUX_IMA" (legacy IMA)
TPM2B_DIGEST	eventDigest	Digest of the event; i.e. the digest of the measured file

string	eventName	Name of the event; i.e. the name of the measured file.

The data type in Table 2 SHALL be encoded according to the JSON format description defined in TCG TSS 2.0 JSON Policy Specification.

Start of informative comment

Example:

```
{
    "recnum": …,
    "pcr": …,
    "digest": …,
    "type": …,
    "eventDigest": …,
    "eventName": …
}
```

End of informative comment

3.8 QuoteInfo encoding

A quote info in FAPI is defined in Table 3.

Table 3 TPMS_QUOTEINFORMATION

TYPE	FIELD	DESCRIPTION
TPMT_SIG_SCHEME	sig_scheme	The signature scheme used during the quote
TPMS_ATTEST	attest	The attestation information

The data type in Table 3 SHALL be encoded according to the JSON format description defined in TCG TSS 2.0 JSON Policy Specification.

Start of informative comment

Example:

```
{
    "sig_scheme": …,
    "attest": …
}
```

End of informative comment

3.9 Encrypted data encoding

Encrypted data in FAPI is defined in Table 4.

Table 4 TPMS_ENCRYPTEDDATA

TYPE	FIELD	DESCRIPTION
UINT32	type	encryption type
TPM2B_NAME	key_name	Name of the decryption key
UINT8[]	cipher	Ciphertext
TPM2B_PRIVATE	sym_private	Private portion of the symmetric key (TPM encrypted)
TPM2B_PUBLIC	sym_public	Public portion of the symmetric key
UINT32	sym_key_size	Size of the encrypted symmetric key

TPM2B_DIGEST	sym_iv	Initialization vector for the symmetric encryption
TPMS_POLICY	policy	JSON-encoded policy

The data type in Table 4 SHALL be encoded according to the JSON format description defined in TCG TSS 2.0 JSON Policy Specification.

Start of informative comment

Example:

```
{
    "type": …,
    "key_name": …,
    "cipher": …,
    "sym_private": …,
    "sym_public": …,
    "sym_key_size": …,
    "sym_iv": …,
    "policy": …,
}
```

End of informative comment

4 Context functions

All functions presented in this section work in contexts without a TPM. Some of them might however still communicate with the TPM.

4.1 Fapi_Initialize

Fapi_Initialize() allocates and initializes a FAPI context and establishes a connection to a TPM. Once created contexts can be released by calling Fapi_Finalize().

In the asynchronous version of this operation the initialization is divided between the Fapi_Initialize_Async and Fapi_Initialize_Finish functions. The Fapi_Initialize_Async function allocates the FAPI context, initiates whatever asynchronous operations it needs internally, and returns the partially-initialized context to the caller. The Fapi_Initialize_Finish function accepts the partially-initialized context as input, checks the status of the outstanding internal operations, and either returns success (initialization complete), TSS2_FAPI_RC_TRY_AGAIN (initialization still in progress), or some other error indicating initialization has failed.

If Fapi_Initialize() or Fapi_Initialize_Finish() return anything other than TSS2_RC_SUCCESS or TSS2_FAPI_RC_TRY_AGAIN the function MUST release all resources allocated during the initialization and set the context pointer to NULL.

The uri parameter is intended to allow the caller to specify how to connect to the TPM if non-default options are needed. A value of NULL indicates the TPM in the local machine should be accessed using whatever defaults are used by the FAPI implementation. For this version of this specification it is an error to supply a non-NULL uri. Future versions of this specification can make use of this parameter to determine which TPM to interact with.

After successful initialization the FAPI context can be released by calling Fapi_Finalize().

4.1.1 Prototype

```
TSS2_RC Fapi_Initialize(
    FAPI_CONTEXT **context,
    char    const *uri);
TSS2_RC Fapi_Initialize_Async(
    FAPI_CONTEXT **context,
    char    const *uri);
TSS2_RC Fapi_Initialize_Finish(
    FAPI_CONTEXT **context);
```

4.1.2 Parameters

- context is a pointer to the opaque context blob currently being operated on. context MUST NOT be NULL.
- The ‘uri’ parameter is unused in this version of the specification and MUST be NULL.

4.1.3 Return Values

- TSS2_RC_SUCCESS: if the function is successful.
- TSS2_FAPI_RC_BAD_REFERENCE: if the context pointer is null.
- TSS2_FAPI_RC_BAD_VALUE: if uri is not NULL.
- TSS2_FAPI_RC_MEMORY: if the FAPI cannot allocate enough memory to create the context.
- TSS2_FAPI_RC_BAD_SEQUENCE: (Finish only) if the operation is called out of sequence.
- TSS2_FAPI_RC_TRY_AGAIN: (Finish only) if the asynchronous operation is incomplete and the Finish should be re-executed later to check for the final result.

4.2 Fapi_Finalize

Fapi_Finalize() closes a context by freeing all resources associated with the context. Any transient objects held in the TPM by the context MUST be flushed by this function. The context pointer is set to NULL.

This function cannot be called while an asynchronous operation is outstanding. If this function is called while an asynchronous operation is outstanding the behavior is undefined.

4.2.1 Prototype

```
void Fapi_Finalize(
    FAPI_CONTEXT  **context);
```

4.2.2 Parameters

- context is a pointer to the opaque context blob currently being operated on. context MUST NOT be NULL.

4.2.3 Return Values

This function does not return any value.

4.3 Fapi_Free

Fapi_Free() frees memory that has been allocated by a FAPI function to hold output parameter values.

This function should not be used to free the context pointer itself. Applications SHALL use Fapi_Finalize() to free the context itself.

When passed NULL this function SHALL do nothing.

4.3.1 Prototype

```
void Fapi_Free(
    void *ptr);
```

4.3.2 Parameters

- ptr is the pointer to to be freed. ptr SHOULD NOT be NULL.

4.3.3 Return Values

This function has no return value.

4.4 Fapi_GetInfo

Fapi_GetInfo() returns a UTF-8 string identifying the version of the FAPI, the TPM, configurations and other relevant information in a human readable format. The concrete content of this string is implementation specific.

4.4.1 Prototype

```
TSS2_RC Fapi_GetInfo(
    FAPI_CONTEXT *context,
    char **info);
TSS2_RC Fapi_GetInfo_Async(
    FAPI_CONTEXT *context);
TSS2_RC Fapi_GetInfo_Finish(
    FAPI_CONTEXT *context,
    char **info);
```

4.4.2 Parameters

- context is a pointer to the opaque context blob currently being operated on. context MUST NOT be NULL.
- info returns the FAPI and TPM information. info MUST NOT be NULL.

4.4.3 Return Values

- TSS2_RC_SUCCESS: if the function call was a success.
- TSS2_FAPI_RC_BAD_REFERENCE: if the context or info pointer is NULL.
- TSS2_FAPI_RC_BAD_CONTEXT: if context corruption is detected.
- TSS2_FAPI_RC_MEMORY: if the FAPI cannot allocate enough memory for internal operations or return parameters.
- TSS2_FAPI_RC_BAD_SEQUENCE: if the synchronous or Async functions are called while the context has another asynchronous operation outstanding, or the Finish function is called while the context does not have an appropriate asynchronous operation outstanding.
- TSS2_FAPI_RC_TRY_AGAIN: (Finish only) if the asynchronous operation is incomplete and the Finish should be re-executed later to check for the final result.

5 General functions

5.1 Fapi_Provision

Fapi_Provision() provisions a FAPI instance and its associated TPM. The steps taken SHALL be:

- Retrieve the EK template, nonce and certificate, verify that they match the TPM's EK and store them in the key store.
- Set the authValues and policies for the Owner (Storage Hierarchy), the Privacy Administrator (Endorsement Hierarchy) and the lockout authority.
- Scan the TPM's nv indices and create entries in the metadata store. This operation MAY use a heuristic to guess the originating programs for nv indices found and name the entries accordingly.
- Create the SRK (storage primary key) inside the TPM and make it persistent if required by the FAPI configuration and stored its metadata in the system-wide metadata store. The SRK will not have an authorization value associated.
  Note: If an authorization value is associated with the storage hierarchy, it is highly RECOMMENDED that the SRK is created without an authorization value and is made persistent in the TPM, such that it is easily accessible by users and applications.

5.1.1 Prototype

```
TSS2_RC Fapi_Provision(
    FAPI_CONTEXT *context,
    char   const *policyPathEh,
    char   const *authValueEh,
    char   const *policyPathSh,
    char   const *authValueSh,
    char   const *authValueLockout);
TSS2_RC Fapi_Provision_Async(
    FAPI_CONTEXT *context,
    char   const *policyPathEh,
    char   const *authValueEh,
    char   const *policyPathSh,
    char   const *authValueSh,
    char   const *authValueLockout);
TSS2_RC Fapi_Provision_Finish(
    FAPI_CONTEXT *context);
```

5.1.2 Parameters

- context is a pointer to the opaque context blob currently being operated on. context MUST NOT be NULL.
- policyPathEh is a policy to be set for the Privacy Administrator, i.e. the endorsement hierarchy. policyPathEh MAY be NULL.
- authValueEh is the authorization value for the Privacy Administrator, i.e. the endorsement hierarchy. authValueEh MAY be NULL.
- policyPathSh is a policy to be set for the owner, i.e. the storage hierarchy. policyPathSh SHOULD be NULL.
- authValueSh is the authorization value for the owner, i.e. the storage hierarchy. authValueSh SHOULD be NULL.
- authValueLockout is the authorization value for the lockout authorization. authValueLockout SHOULD NOT be NULL.

5.1.3 Return Values

- TSS2_RC_SUCCESS: if the function call was a success.

- TSS2_FAPI_RC_BAD_REFERENCE: if the context is NULL.
- TSS2_FAPI_RC_BAD_CONTEXT: if context corruption is detected.
- TSS2_FAPI_RC_BAD_PATH: if policyPathEh or policyPathSh do not map to a FAPI policy.
- TSS2_FAPI_RC_MEMORY: if the FAPI cannot allocate enough memory for internal operations or return parameters.
- TSS2_FAPI_RC_BAD_SEQUENCE: if the synchronous or Async functions are called while the context has another asynchronous operation outstanding, or the Finish function is called while the context does not have an appropriate asynchronous operation outstanding.
- TSS2_FAPI_RC_TRY_AGAIN: (Finish only) if the asynchronous operation is incomplete and the Finish should be re-executed later to check for the final result.

5.2 Fapi_GetPlatformCertificates

Fapi_GetPlatformCertificates() returns the set of Platform certificates concatenated in a continuous buffer if the platform provides platform certificates. Platform certificates for TPM 2.0 can consist not only of a single certificate but also a series of so-called delta certificates.

If no platform certificates are available, the certificatesSize SHALL be set to 0.

5.2.1 Prototype

```
TSS2_RC Fapi_GetPlatformCertificates(
    FAPI_CONTEXT *context,
    uint8_t     **certificates,
    size_t       *certificatesSize);
TSS2_RC Fapi_GetPlatformCertificates_Async(
    FAPI_CONTEXT *context);
TSS2_RC Fapi_GetPlatformCertificates_Finish(
    FAPI_CONTEXT *context,
    uint8_t     **certificates,
    size_t       *certificatesSize);
```

5.2.2 Parameters

- context is a pointer to the opaque context blob currently being operated on. context MUST NOT be NULL.
- certificates returns a continuous buffer containing the concatenated platform certificates. certificates MUST NOT be NULL.
- certificatesSize returns the size of the buffer returned by certificates. certificatesSize MAY be NULL.

5.2.3 Return Values

- TSS2_RC_SUCCESS: if the function call was a success.
- TSS2_FAPI_RC_BAD_REFERENCE: if the context or certificates pointer are NULL.
- TSS2_FAPI_RC_BAD_CONTEXT: if context corruption is detected.
- TSS2_FAPI_RC_MEMORY: if the FAPI cannot allocate enough memory for internal operations or return parameters.
- TSS2_FAPI_RC_BAD_SEQUENCE: if the synchronous or Async functions are called while the context has another asynchronous operation outstanding, or the Finish function is called while the context does not have an appropriate asynchronous operation outstanding.
- TSS2_FAPI_RC_TRY_AGAIN: (Finish only) if the asynchronous operation is incomplete and the Finish should be re-executed later to check for the final result.

5.3 Fapi_GetRandom

Fapi_GetRandom() uses the TPM to create an array of random bytes. This function may perform multiple calls to the TPM if the number of bytes requested by the caller is larger than the maximum number of bytes that the TPM will return per call.

5.3.1 Prototype

```
TSS2_RC Fapi_GetRandom(
    FAPI_CONTEXT *context,
    size_t         numBytes,
    uint8_t      **data);
TSS2_RC Fapi_GetRandom_Async(
    FAPI_CONTEXT *context,
    size_t         numBytes);
TSS2_RC Fapi_GetRandom_Finish(
    FAPI_CONTEXT *context,
    uint8_t      **data);
```

5.3.2 Parameters

- context is a pointer to the opaque context blob currently being operated on. context MUST NOT be NULL.
- numBytes is the number of bytes requested by the caller
- data is the array of random bytes. data MUST NOT be NULL.

5.3.3 Return Codes

- TSS2_RC_SUCCESS: if the function call was a success.
- TSS2_FAPI_RC_BAD_REFERENCE: if context or data is NULL.
- TSS2_FAPI_RC_BAD_CONTEXT: if context corruption is detected.
- TSS2_FAPI_RC_BAD_VALUE: if numBytes is 0.
- TSS2_FAPI_RC_MEMORY: if the FAPI cannot allocate enough memory for internal operations or return parameters.
- TSS2_FAPI_RC_BAD_SEQUENCE: if the synchronous or Async functions are called while the context has another asynchronous operation outstanding, or the Finish function is called while the context does not have an appropriate asynchronous operation outstanding.
- TSS2_FAPI_RC_TRY_AGAIN: (Finish only) if the asynchronous operation is incomplete and the Finish should be re-executed later to check for the final result.

5.4 Fapi_Import

Fapi_Import() imports a JSON encoded policy or policy template encoded according to TCG TSS 2.0 JSON Policy Language Specification and stores it under the provided path or imports a JSON encoded key under the provided path.

5.4.1 Prototype

```
TSS2_RC Fapi_Import(
    FAPI_CONTEXT *context,
    char    const *path,
    char    const *importData);
TSS2_RC Fapi_Import_Async(
    FAPI_CONTEXT *context,
    char    const *path,
    char    const *importData);
```

```
TSS2_RC Fapi_Import_Finish(
    FAPI_CONTEXT *context);
```

5.4.2 Parameters

- context is a pointer to the opaque context blob currently being operated on. context MUST NOT be NULL.
- path is the path of the new object. path MUST NOT be NULL.
- importData is the data to be imported. importData MUST NOT be NULL.

5.4.3 Return Values

- TSS2_RC_SUCCESS: if the function call was a success.
- TSS2_FAPI_RC_BAD_REFERENCE: if context or path is NULL.
- TSS2_FAPI_RC_BAD_CONTEXT: if context corruption is detected.
- TSS2_FAPI_RC_BAD_PATH: if path does not map to a FAPI policy or key.
- TSS2_FAPI_RC_PATH_ALREADY_EXISTS: if a policy at path already exists.
- TSS2_FAPI_RC_BAD_VALUE: if importData contains invalid data.
- TSS2_FAPI_RC_STORAGE_ERROR: if the FAPI storage cannot be updated.
- TSS2_FAPI_RC_MEMORY: if the FAPI cannot allocate enough memory for internal operations or return parameters.
- TSS2_FAPI_RC_BAD_SEQUENCE: if the synchronous or Async functions are called while the context has another asynchronous operation outstanding, or the Finish function is called while the context does not have an appropriate asynchronous operation outstanding.
- TSS2_FAPI_RC_TRY_AGAIN: (Finish only) if the asynchronous operation is incomplete and the Finish should be re-executed later to check for the final result.

5.5 Fapi_List

Fapi_List() enumerates all objects in the metadata store in a given path. The returned list SHALL consist of complete paths from the root (not relative paths from the search path), such that they can be directly used in another query. The values in this list SHALL be colon-separated.

5.5.1 Prototype

```
TSS2_RC Fapi_List(
    FAPI_CONTEXT *context,
    char   const *searchPath,
    char         **pathList);
TSS2_RC Fapi_List_Async(
    FAPI_CONTEXT *context,
    char    char *searchPath);
TSS2_RC Fapi_List_Finish(
    FAPI_CONTEXT *context,
    char         **pathList);
```

5.5.2 Parameters

- context is a pointer to the opaque context blob currently being operated on. context MUST NOT be NULL.
- searchPath is the path identifying the root of the search. searchPath MUST NOT be NULL.
- pathList returns colon-separated list of paths. pathList MUST NOT be NULL.

5.5.3 Return Values

- TSS2_RC_SUCCESS: if the function call was a success.
- TSS2_FAPI_RC_BAD_REFERENCE: if context, searchPath or pathList is NULL.

- TSS2_FAPI_RC_BAD_CONTEXT: if context corruption is detected.
- TSS2_FAPI_RC_BAD_PATH: if searchPath does not map to a FAPI entity.
- TSS2_FAPI_RC_MEMORY: if the FAPI cannot allocate enough memory for internal operations or return parameters.
- TSS2_FAPI_RC_BAD_SEQUENCE: if the synchronous or Async functions are called while the context has another asynchronous operation outstanding, or the Finish function is called while the context does not have an appropriate asynchronous operation outstanding.
- TSS2_FAPI_RC_TRY_AGAIN: (Finish only) if the asynchronous operation is incomplete and the Finish should be re-executed later to check for the final result.

5.6 Fapi_Delete

Fapi_Delete deletes the given key, policy or NV from the system. Depending on the entity type, one of the following actions SHALL be taken:

- Non-persistent key: Flush from TPM (if loaded) and delete public and private blobs from keystore.
- Persistent keys: Evict from TPM and delete public and private blobs from keystore
- Primary keys: Flush from TPM and delete public blobs from keystore
- NV index: Undefine NV index from TPM and delete public blob from metadata store
- Policies: Delete entry from policy store
- Hierarchy, PCR: Return TSS2_FAPI_RC_NOT_DELETABLE
- Special keys EK, SRK: Return TSS2_FAPI_RC_NOT_DELETABLE

5.6.1 Prototype

```
TSS2_RC Fapi_Delete(
    FAPI_CONTEXT *context,
    char   const *path);
TSS2_RC Fapi_Delete_Async(
    FAPI_CONTEXT *context,
    char   const *path);
TSS2_RC Fapi_Delete_Finish(
    FAPI_CONTEXT *context);
```

5.6.2 Parameters

- context is a pointer to the opaque context blob currently being operated on. context MUST NOT be NULL.
- path is the path to the entity to delete. path MUST NOT be NULL.

5.6.3 Return Values

- TSS2_RC_SUCCESS: if the function call was a success.
- TSS2_FAPI_RC_BAD_REFERENCE: if context or path is NULL.
- TSS2_FAPI_RC_BAD_CONTEXT: if context corruption is detected.
- TSS2_FAPI_RC_BAD_PATH: if path does not map to a FAPI entity.
- TSS2_FAPI_RC_NOT_DELETABLE: if the entity is not deletable or the path is read-only.
- TSS2_FAPI_RC_STORAGE_ERROR: if the FAPI storage cannot be updated for any other reason.
- TSS2_FAPI_RC_MEMORY: if the FAPI cannot allocate enough memory for internal operations or return parameters.
- TSS2_FAPI_RC_BAD_SEQUENCE: if the synchronous or Async functions are called while the context has another asynchronous operation outstanding, or the Finish function is called while the context does not have an appropriate asynchronous operation outstanding.
- TSS2_FAPI_RC_TRY_AGAIN: (Finish only) if the asynchronous operation is incomplete and the Finish should be re-executed later to check for the final result.

5.7 Fapi_ChangeAuth

Fapi_ChangeAuth changes the authorization data of an entity referenced by the path.

The authValue is a UTF-8 password. If the length of the password is larger than the digest size of the entity's nameAlg (which is stored internally as part of its meta data) then the FAPI should hash the password, in accordance with the TPM specification, part 1 rev 138, section 19.6.4.3 "Authorization Size Convention."

5.7.1 Prototype

```
TSS2_RC Fapi_ChangeAuth(
    FAPI_CONTEXT *context,
    char   const *entityPath,
    char   const *authValue);
TSS2_RC Fapi_ChangeAuth_Async(
    FAPI_CONTEXT *context,
    char   const *entityPath,
    char   const *authValue);
TSS2_RC Fapi_ChangeAuth_Finish(
    FAPI_CONTEXT *context);
```

5.7.2 Parameters

- context is a pointer to the opaque context blob currently being operated on. context MUST NOT be NULL.
- entityPath is the path identifying the entity to modify. entityPath MUST NOT be NULL.
- authValue is the new 0-terminated password. authValue MAY be NULL. If authValue is NULL then the password is set to the empty string.

5.7.3 Return Values

- TSS2_RC_SUCCESS: if the function call was a success.
- TSS2_FAPI_RC_BAD_REFERENCE: if context, entityPath, or authValue is NULL.
- TSS2_FAPI_RC_BAD_CONTEXT: if context corruption is detected.
- TSS2_FAPI_RC_BAD_PATH: if entityPath does not map to a FAPI entity.
- TSS2_FAPI_RC_STORAGE_ERROR: if the updated data cannot be saved.
- TSS2_FAPI_RC_MEMORY: if the FAPI cannot allocate enough memory for internal operations or return parameters.
- TSS2_FAPI_RC_BAD_SEQUENCE: if the synchronous or Async functions are called while the context has another asynchronous operation outstanding, or the Finish function is called while the context does not have an appropriate asynchronous operation outstanding.
- TSS2_FAPI_RC_TRY_AGAIN: (Finish only) if the asynchronous operation is incomplete and the Finish should be re-executed later to check for the final result.

5.8 Fapi_SetDescription

Fapi_SetDescription() allows an application to assign a human readable description to an object in the metadata store. Previously stored descriptions SHALL be overwritten by this function. If NULL is passed in, any stored description SHALL be deleted.

5.8.1 Prototype

```
TSS2_RC Fapi_SetDescription(
    FAPI_CONTEXT   *context,
    const char     *path,
    char     const *description);
```

```
TSS2_RC Fapi_SetDescription_Async(
    FAPI_CONTEXT   *context,
    const char     *path,
    char      const *description);
TSS2_RC Fapi_SetDescription_Finish(
    FAPI_CONTEXT   *context);
```

5.8.2 Parameters

- context is a pointer to the opaque context blob currently being operated on. context MUST NOT be NULL.
- path is the path of the object for which the appData will be stored. path MUST NOT be NULL.
- description is the date to be stored. description MAY be NULL.

5.8.3 Return Values

- TSS2_RC_SUCCESS: if the function call was a success.
- TSS2_FAPI_RC_BAD_REFERENCE: if context or path is NULL or if appData is NULL whilst appDataSize is not zero.
- TSS2_FAPI_RC_BAD_CONTEXT: if context corruption is detected.
- TSS2_FAPI_RC_BAD_PATH: if entityPath does not map to a FAPI entity.
- TSS2_FAPI_RC_STORAGE_ERROR: if the updated data cannot be saved.
- TSS2_FAPI_RC_MEMORY: if the FAPI cannot allocate enough memory for internal operations or return parameters.
- TSS2_FAPI_RC_BAD_SEQUENCE: if the synchronous or Async functions are called while the context has another asynchronous operation outstanding, or the Finish function is called while the context does not have an appropriate asynchronous operation outstanding.
- TSS2_FAPI_RC_TRY_AGAIN: (Finish only) if the asynchronous operation is incomplete and the Finish should be re-executed later to check for the final result.

5.9 Fapi_GetDescription

Fapi_GetDescription() returns the previously stored application data for an object. If no description is present, description SHALL be set to an empty string.

5.9.1 Prototype

```
TSS2_RC Fapi_GetDescription(
    FAPI_CONTEXT   *context,
    const char     *path,
    char          **description);
TSS2_RC Fapi_GetDescription_Async(
    FAPI_CONTEXT   *context,
    const char     *path);
TSS2_RC Fapi_GetDescription_Finish(
    FAPI_CONTEXT   *context,
    char          **description);
```

5.9.2 Parameters

- context is a pointer to the opaque context blob currently being operated on. context MUST NOT be NULL.
- path is the path of the object for which the appData will be loaded. path MUST NOT be NULL.
- description returns the stored description. description MUST NOT be NULL.

5.9.3 Return Values

- TSS2_RC_SUCCESS: if the function call was a success.
- TSS2_FAPI_RC_BAD_REFERENCE: if context or path is NULL.

- TSS2_FAPI_RC_BAD_CONTEXT: if context corruption is detected.
- TSS2_FAPI_RC_BAD_PATH: if entityPath does not map to a FAPI entity.
- TSS2_FAPI_RC_STORAGE_ERROR: if the updated data cannot be loaded.
- TSS2_FAPI_RC_MEMORY: if the FAPI cannot allocate enough memory for internal operations or return parameters.
- TSS2_FAPI_RC_BAD_SEQUENCE: if the synchronous or Async functions are called while the context has another asynchronous operation outstanding, or the Finish function is called while the context does not have an appropriate asynchronous operation outstanding.
- TSS2_FAPI_RC_TRY_AGAIN: (Finish only) if the asynchronous operation is incomplete and the Finish should be re-executed later to check for the final result.

5.10 Fapi_SetAppData

Fapi_SetAppData() allows an application to associate an arbitrary data blob with a given object. The data SHALL be stored and the same data SHALL be returned upon Fapi_GetAppData. Previously stored data SHALL be overwritten by this function. If NULL is passed in, stored data SHALL be deleted.

5.10.1 Prototype

```
TSS2_RC Fapi_SetAppData(
    FAPI_CONTEXT   *context,
    const char     *path,
    uint8_t  const *appData,
    size_t          appDataSize);
TSS2_RC Fapi_SetAppData_Async(
    FAPI_CONTEXT   *context,
    const char     *path,
    uint8_t  const *appData,
    size_t          appDataSize);
TSS2_RC Fapi_SetAppData_Finish(
    FAPI_CONTEXT   *context);
```

5.10.2 Parameters

- context is a pointer to the opaque context blob currently being operated on. context MUST NOT be NULL.
- path is the path of the object for which the appData will be stored. path MUST NOT be NULL.
- appData is the data to be stored. appData MAY be NULL.
- appDataSize is the size of appData.

5.10.3 Return Values

- TSS2_RC_SUCCESS: if the function call was a success.
- TSS2_FAPI_RC_BAD_REFERENCE: if context or path is NULL or if appData is NULL whilst appDataSize is not zero.
- TSS2_FAPI_RC_BAD_CONTEXT: if context corruption is detected.
- TSS2_FAPI_RC_BAD_PATH: if entityPath does not map to a FAPI entity.
- TSS2_FAPI_RC_STORAGE_ERROR: if the updated data cannot be saved.
- TSS2_FAPI_RC_MEMORY: if the FAPI cannot allocate enough memory for internal operations or return parameters.
- TSS2_FAPI_RC_BAD_SEQUENCE: if the synchronous or Async functions are called while the context has another asynchronous operation outstanding, or the Finish function is called while the context does not have an appropriate asynchronous operation outstanding.
- TSS2_FAPI_RC_TRY_AGAIN: (Finish only) if the asynchronous operation is incomplete and the Finish should be re-executed later to check for the final result.

5.11 Fapi_GetAppData

Fapi_GetAppData() returns the previously stored application data for an object. If no application data is present, then appDataSize SHALL be 0.

5.11.1 Prototype

```
TSS2_RC Fapi_GetAppData(
    FAPI_CONTEXT   *context,
    const char     *path,
    uint8_t       **appData,
    size_t         *appDataSize);
TSS2_RC Fapi_GetAppData_Async(
    FAPI_CONTEXT   *context,
    const char     *path);
TSS2_RC Fapi_GetAppData_Finish(
    FAPI_CONTEXT   *context,
    uint8_t       **appData,
    size_t         *appDataSize);
```

5.11.2 Parameters

- context is a pointer to the opaque context blob currently being operated on. context MUST NOT be NULL.
- path is the path of the object for which the appData will be loaded. path MUST NOT be NULL.
- appData returns a copy of the stored data. appData MAY be NULL.
- appDataSize returns the size of appData. appDataSize MAY be NULL.

5.11.3 Return Values

- TSS2_RC_SUCCESS: if the function call was a success.
- TSS2_FAPI_RC_BAD_REFERENCE: if context or path is NULL.
- TSS2_FAPI_RC_BAD_CONTEXT: if context corruption is detected.
- TSS2_FAPI_RC_BAD_PATH: if entityPath does not map to a FAPI entity.
- TSS2_FAPI_RC_STORAGE_ERROR: if the updated data cannot be loaded.
- TSS2_FAPI_RC_MEMORY: if the FAPI cannot allocate enough memory for internal operations or return parameters.
- TSS2_FAPI_RC_BAD_SEQUENCE: if the synchronous or Async functions are called while the context has another asynchronous operation outstanding, or the Finish function is called while the context does not have an appropriate asynchronous operation outstanding.
- TSS2_FAPI_RC_TRY_AGAIN: (Finish only) if the asynchronous operation is incomplete and the Finish should be re-executed later to check for the final result.

5.12 Fapi_GetTpmBlobs

Fapi_GetTpmBlobs() returns the public and private blobs of an object, such that they could be loaded by a low-level API (e.g. ESAPI). It also returns the policy associated with these blobs in JSON format.

5.12.1 Prototype

```
TSS2_RC Fapi_GetTpmBlobs(
    FAPI_CONTEXT *context,
    const char   *path,
    uint8_t     **tpm2bPublic,
    size_t       *tpm2bPublicSize,
    uint8_t     **tpm2bPrivate,
    size_t       *tpm2bPrivateSize,
    char        **policy);
TSS2_RC Fapi_GetTpmBlobs_Async(
    FAPI_CONTEXT *context,
    const char   *path);
TSS2_RC Fapi_GetTpmBlobs_Finish(
    FAPI_CONTEXT *context,
    uint8_t     **tpm2bPublic,
    size_t       *tpm2bPublicSize,
    uint8_t     **tpm2bPrivate,
    size_t       *tpm2bPrivateSize,
    char        **policy);
```

5.12.2 Parameters

- context is a pointer to the opaque context blob currently being operated on. context MUST NOT be NULL.
- path is the path of the object for which the blobs will be returned. path MUST NOT be NULL.
- tpm2bPublic is the returned public area of the object as a marshalled TPM2B_PUBLIC. tpm2bPublic MAY be NULL.
- tpm2bPublicSize is the size of tpm2bPublic. tpm2bPublicSize MAY be NULL.
- tpm2bPrivate is the returned private area of the object as a marshalled TPM2B_PRIVATE. tpm2bPrivate MAY be NULL.
- tpm2bPrivateSize is the size of tpm2bPrivate. tpm2bPrivateSize MAY be NULL.
- policy is the returned policy associated with the object, encoded in JSON. policy MAY be NULL.

5.12.3 Return Values

- TSS2_RC_SUCCESS: if the function call was a success.
- TSS2_FAPI_RC_BAD_REFERENCE: if context or path is NULL.
- TSS2_FAPI_RC_BAD_CONTEXT: if context corruption is detected.
- TSS2_FAPI_RC_BAD_PATH: if path does not map to a FAPI entity.
- TSS2_FAPI_RC_STORAGE_ERROR: if the updated data cannot be saved.
- TSS2_FAPI_RC_MEMORY: if the FAPI cannot allocate enough memory for internal operations or return parameters.
- TSS2_FAPI_RC_BAD_SEQUENCE: if the synchronous or Async functions are called while the context has another asynchronous operation outstanding, or the Finish function is called while the context does not have an appropriate asynchronous operation outstanding.
- TSS2_FAPI_RC_TRY_AGAIN: (Finish only) if the asynchronous operation is incomplete and the Finish should be re-executed later to check for the final result.

6 Key functions

6.1 Fapi_CreateKey

Fapi_CreateKey() creates a key inside the TPM and stores it in the FAPI metadata store and if requested persistently inside the TPM.

6.1.1 Prototype

```
TSS2_RC Fapi_Create(
    FAPI_CONTEXT *context,
    char   const *path,
    char   const *type,
    char   const *policyPath,
    char   const *authValue);
TSS2_RC Fapi_Create_Async(
    FAPI_CONTEXT *context,
    char   const *path,
    char   const *type,
    char   const *policyPath,
    char   const *authValue);
TSS2_RC Fapi_Create_Finish(
    FAPI_CONTEXT *context);
```

6.1.2 Parameters

- context is a pointer to the opaque context blob currently being operated on. context MUST NOT be NULL.
- path is the path to the new key. path MUST NOT be NULL.
- type identifies the intended usage, see section 3.3. type MAY be NULL.
- policyPath identifies the policy to be associated with the new key. policyPath MAY be NULL. If policyPath is NULL then no policy will be associated with the key.
- authValue is the new authorization value for the key. authValue MAY be NULL. If authValue is NULL then the authorization value will be the empty string.

6.1.3 Return Values

- TSS2_RC_SUCCESS: if the function call was a success.
- TSS2_FAPI_RC_BAD_REFERENCE: if context or path is NULL.
- TSS2_FAPI_RC_BAD_CONTEXT: if context corruption is detected.
- TSS2_FAPI_RC_KEY_NOT_FOUND: if the parent specified in path does not map to a FAPI key.
- TSS2_FAPI_RC_BAD_PATH: if policyPath is non-NULL and does not map to a FAPI policy.
- TSS2_FAPI_RC_PATH_ALREADY_EXISTS: if path already exists.
- TSS2_FAPI_RC_BAD_VALUE: if type is non-NULL and invalid.
- TSS2_FAPI_RC_STORAGE_ERROR: if the FAPI storage cannot be updated.
- TSS2_FAPI_RC_IO_ERROR: if the data cannot be saved.
- TSS2_FAPI_RC_MEMORY: if the FAPI cannot allocate enough memory for internal operations or return parameters.
- TSS2_FAPI_RC_BAD_SEQUENCE: if the synchronous or Async functions are called while the context has another asynchronous operation outstanding, or the Finish function is called while the context does not have an appropriate asynchronous operation outstanding.
- TSS2_FAPI_RC_TRY_AGAIN: (Finish only) if the asynchronous operation is incomplete and the Finish should be re-executed later to check for the final result.

6.2 Fapi_Sign

Fapi_Sign() uses a key inside the TPM to sign a digest value.

6.2.1 Prototype

```
TSS2_RC Fapi_Sign(
    FAPI_CONTEXT  *context,
    char    const *keyPath,
    char    const *padding,
    uint8_t const *digest,
    size_t         digestSize,
    uint8_t      **signature,
    size_t        *signatureSize,
    uint8_t      **publicKey,
    uint8_t      **certificate);
TSS2_RC Fapi_Sign_Async(
    FAPI_CONTEXT  *context,
    char    const *keyPath,
    uint8_t const *digest,
    size_t         digestSize);
TSS2_RC Fapi_Sign_Finish(
    FAPI_CONTEXT *context,
    uint8_t     **signature,
    size_t       *signatureSize,
    uint8_t     **publicKey,
    uint8_t     **certificate);
```

6.2.2 Parameters

- context is a pointer to the opaque context blob currently being operated on. context MUST NOT be NULL.
- keyPath is the path to the signing key. keyPath MUST NOT be NULL.
- padding is the padding algorithm used. Possible values are “RSA_SSA”, “RSA_PPS” (case insensitive). padding MAY be NULL.
- digest is the data to be signed, already hashed. digest MUST NOT be NULL.
- digestSize is the number of bytes in digest.
- signature returns the signature in binary form. signature MUST NOT be NULL.
- signatureSize is the number of bytes in signature. signatureSize MAY be NULL
- publicKey is the public key associated with keyPath in PEM format. publicKey MAY be NULL.
- certificate is the certificate associated with keyPath in PEM format. certificate MAY be NULL.

6.2.3 Return Values

- TSS2_RC_SUCCESS: if the function call was a success.
- TSS2_FAPI_RC_BAD_REFERENCE: if context, keyPath, digest, signature or signatureSize is NULL.
- TSS2_FAPI_RC_BAD_CONTEXT: if context corruption is detected.
- TSS2_FAPI_RC_KEY_NOT_FOUND: if keyPath does not map to a FAPI object.
- TSS2_FAPI_RC_BAD_KEY: if the object at keyPath is not a key, or is a key that is unsuitable for the requested operation.
- TSS2_FAPI_RC_BAD_VALUE: if digestSize is 0.
- TSS2_FAPI_RC_MEMORY: if the FAPI cannot allocate enough memory for internal operations or return parameters.

- TSS2_FAPI_RC_BAD_SEQUENCE: if the synchronous or Async functions are called while the context has another asynchronous operation outstanding, or the Finish function is called while the context does not have an appropriate asynchronous operation outstanding.
- TSS2_FAPI_RC_TRY_AGAIN: (Finish only) if the asynchronous operation is incomplete and the Finish should be re-executed later to check for the final result.

6.3 Fapi_VerifySignature

Fapi_VerifySignature() verifies a signature using a public key found in a keyPath. This function MAY or MAY NOT use the TPM for this operation.

6.3.1 Prototype

```
TSS2_RC Fapi_VerifySignature(
    FAPI_CONTEXT  *context,
    char    const *keyPath,
    uint8_t const *digest,
    size_t         digestSize,
    uint8_t const *signature,
    size_t         signatureSize);
TSS2_RC Fapi_VerifySignature_Async(
    FAPI_CONTEXT  *context,
    char    const *keyPath,
    uint8_t const *digest,
    size_t         digestSize,
    uint8_t const *signature,
    size_t         signatureSize);
TSS2_RC Fapi_VerifySignature_Finish(
    FAPI_CONTEXT  *context);
```

6.3.2 Parameters

- context is a pointer to the opaque context blob currently being operated on. context MUST NOT be NULL.
- keyPath is the path to the verification public key. keyPath MUST NOT be NULL.
- digest is the data that was signed, already hashed. digest MUST NOT be NULL.
- digestSize is the number of bytes in digest.
- signature is the signature to be verified. signature MUST NOT be NULL.
- signatureSize is the number of bytes in signature.

6.3.3 Return Values

- TSS2_RC_SUCCESS: if the function call was a success.
- TSS2_FAPI_RC_BAD_REFERENCE: if context, keyPath signature or digest is NULL.
- TSS2_FAPI_RC_BAD_CONTEXT: if context corruption is detected.
- TSS2_FAPI_RC_KEY_NOT_FOUND: if publicKeyPath does not map to a FAPI object.
- TSS2_FAPI_RC_BAD_KEY: if the object at publicKeyPath is not a key, or is a key that is unsuitable for the requested operation.
- TSS2_FAPI_RC_BAD_VALUE: if digestSize is 0.
- TSS2_FAPI_RC_SIGNATURE_VERIFICATION_FAILED: if the signature verification fails.
- TSS2_FAPI_RC_IO_ERROR: if the data cannot be saved.
- TSS2_FAPI_RC_MEMORY: if the FAPI cannot allocate enough memory for internal operations or return parameters.

- TSS2_FAPI_RC_BAD_SEQUENCE: if the synchronous or Async functions are called while the context has another asynchronous operation outstanding, or the Finish function is called while the context does not have an appropriate asynchronous operation outstanding.
- TSS2_FAPI_RC_TRY_AGAIN: (Finish only) if the asynchronous operation is incomplete and the Finish should be re-executed later to check for the final result.

6.4 Fapi_Encrypt

Fapi_Encrypt() encrypts the provided data for a target key.

If keypath is an asymmetric key and a plaintext with size >= TPM2_MAX_SYM_SIZE is provided, Fapi_Encrypt() will bulk-encrypt the plaintext with an intermediate symmetric key and then “seal” this intermediate symmetric key with keyPath as a KEYEDHASH TPM object. This keyPath may refer to the local TPM or to a public key of a remote TPM where the KEYEDHASH can be imported. A following decrypt operation would perform a TPM2_Unseal. ciphertext output contains a reference to the decryption key, the sealed symmetric key (if any), the policy instance, and the encrypted plaintext.

If plaintext has a size <= TPM2_MAX_SYM_SIZE the plaintext is sealed directly to keyPath.

If encrypting for the local TPM (if keyPath is not from the external hierarchy), a storage key (symmetric or asymmetric) is required as keyPath (aka parent key) and the data intermediate symmetric key is created using TPM2_Create() as a KEYEDHASH object.

If encrypting for a remote TPM, an asymmetric storage key is required as keyPath (aka parent key), and the data/intermediate symmetric key is encrypted such that it can be used in a TPM2_Import operation.

The format of the cipherText is described in Section 3.9.

6.4.1 Prototype

```
TSS2_RC Fapi_Encrypt(
    FAPI_CONTEXT  *context,
    char    const *keyPath,
    char    const *policyPath,
    uint8_t const *plainText,
    size_t         plainTextSize,
    char         **cipherText);
TSS2_RC Fapi_Encrypt_Async(
    FAPI_CONTEXT  *context,
    char    const *keyPath,
    char    const *policyPath,
    uint8_t const *plainText,
    size_t         plaintextSize);
TSS2_RC Fapi_Encrypt_Finish(
    FAPI_CONTEXT  *context,
    uint8_t      **cipherText);
```

6.4.2 Parameters

- context is a pointer to the opaque context blob currently being operated on. context MUST NOT be NULL.
- keyPath identifies the encryption key. keyPath MUST NOT be NULL.
- policyPath identifies the policy to be associated with the sealed data. policyPath MAY be NULL. If policyPath is NULL then the sealed data will have no policy.
- plainText is the data to be encrypted. plaintext MUST NOT be NULL.
- plainTextSize is the number of bytes in plaintext.

- cipherText returns the JSON-encoded ciphertext. cipherText MUST NOT be NULL.

6.4.3 Return Values

- TSS2_RC_SUCCESS: if the function call was a success.
- TSS2_FAPI_RC_BAD_REFERENCE: if context, keyPath, plainText, or cipherText is NULL.
- TSS2_FAPI_RC_BAD_CONTEXT: if context corruption is detected.
- TSS2_FAPI_RC_KEY_NOT_FOUND: if keyPath does not map to a FAPI entity.
- TSS2_FAPI_RC_BAD_KEY: if the entity at keyPath is not a key, or is a key that is unsuitable for the requested operation.
- TSS2_FAPI_RC_BAD_VALUE: if plainTextSize is 0.
- TSS2_FAPI_RC_MEMORY: if the FAPI cannot allocate enough memory for internal operations or return parameters.
- TSS2_FAPI_RC_BAD_SEQUENCE: if the synchronous or Async functions are called while the context has another asynchronous operation outstanding, or the Finish function is called while the context does not have an appropriate asynchronous operation outstanding.
- TSS2_FAPI_RC_TRY_AGAIN: (Finish only) if the asynchronous operation is incomplete and the Finish should be re-executed later to check for the final result.

6.5 Fapi_Decrypt

Fapi_Decrypt() decrypts data that was encrypted using Fapi_Encrypt(). The method is described in section 6.4.

6.5.1 Prototype

```
TSS2_RC Fapi_Decrypt(
    FAPI_CONTEXT *context,
    char   const *cipherText,
    uint8_t     **plainText,
    size_t       *plainTextSize);
TSS2_RC Fapi_Decrypt_Async(
    FAPI_CONTEXT *context,
    char   const *cipherText);
TSS2_RC Fapi_Decrypt_Finish(
    FAPI_CONTEXT *context,
    uint8_t     **plainText,
    size_t       *plainTextSize);
```

6.5.2 Parameters

- context is a pointer to the opaque context blob currently being operated on. context MUST NOT be NULL.
- cipherText is the JSON-encoded cipherText. cipherText MUST NOT be NULL.
- plainText returns the decrypted data. plainText MAY be NULL.
- plainTextSize returns the number of bytes in plainText. plainTextSize MAY be NULL.

6.5.3 Return Values

- TSS2_RC_SUCCESS: if the function call was a success.
- TSS2_FAPI_RC_BAD_REFERENCE: if context, newPathName, or importedData is NULL.
- TSS2_FAPI_RC_BAD_CONTEXT: if context corruption is detected.
- TSS2_FAPI_RC_KEY_NOT_FOUND: if the decryption key cannot be found.
- TSS2_FAPI_RC_BAD_KEY: if the decryption key is unsuitable for the requested operation.
- TSS2_FAPI_RC_BAD_VALUE: if the decryption fails.
- TSS2_FAPI_RC_MEMORY: if the FAPI cannot allocate enough memory for internal operations or return parameters.

- TSS2_FAPI_RC_BAD_SEQUENCE: if the synchronous or Async functions are called while the context has another asynchronous operation outstanding, or the Finish function is called while the context does not have an appropriate asynchronous operation outstanding.
- TSS2_FAPI_RC_TRY_AGAIN: (Finish only) if the asynchronous operation is incomplete and the Finish should be re-executed later to check for the final result.

6.6 Fapi_SetCertificate

Fapi_SetCertificate() stores an x509 certificate in PEM encoding with the key referred to by path.

6.6.1 Prototype

```
TSS2_RC Fapi_SetCertificate(
    FAPI_CONTEXT  *context,
    char    const *path,
    char    const *x509certData);
TSS2_RC Fapi_SetCertificate_Async(
    FAPI_CONTEXT  *context,
    char    const *path,
    char    const *x509certData);
TSS2_RC Fapi_SetCertificate_Finish(
    FAPI_CONTEXT  *context);
```

6.6.2 Parameters

- context is a pointer to the opaque context blob currently being operated on. context MUST NOT be NULL.
- path identifies the entity to be associated with the certificate. path MUST NOT be NULL.
- x509certData is the PEM encoded certificate. x509certData MAY be NULL. If x509certData is NULL then the stored x509 certificate SHALL be removed.

6.6.3 Return Values

- TSS2_RC_SUCCESS: if the function call was a success.
- TSS2_FAPI_RC_BAD_REFERENCE: if context, path or x509certData is NULL.
- TSS2_FAPI_RC_BAD_CONTEXT: if context corruption is detected.
- TSS2_FAPI_RC_KEY_NOT_FOUND: if the path does not map to a FAPI entity.
- TSS2_FAPI_RC_BAD_KEY: if the parent of path is not a key.
- TSS2_FAPI_RC_STORAGE_ERROR: if the FAPI storage cannot be updated.
- TSS2_FAPI_RC_MEMORY: if the FAPI cannot allocate enough memory for internal operations or return parameters.
- TSS2_FAPI_RC_BAD_SEQUENCE: if the synchronous or Async functions are called while the context has another asynchronous operation outstanding, or the Finish function is called while the context does not have an appropriate asynchronous operation outstanding.
- TSS2_FAPI_RC_TRY_AGAIN: (Finish only) if the asynchronous operation is incomplete and the Finish should be re-executed later to check for the final result.

6.7 Fapi_GetCertificate

Fapi_GetCertificate() returns the PEM encoded X.509 certificate associated with the key at path.

6.7.1 Prototype

```
TSS2_RC Fapi_GetCertificate(
    FAPI_CONTEXT *context,
    char   const *path,
    char         **x509certData);
```

```
TSS2_RC Fapi_GetCertificate_Async(
    FAPI_CONTEXT *context,
    char   const *path);
TSS2_RC Fapi_GetCertificate_Finish(
    FAPI_CONTEXT *context,
    char        **x509certData);
```

6.7.2 Parameters

- context is a pointer to the opaque context blob currently being operated on. context MUST NOT be NULL.
- path is the entity whose certificate is requested. path MUST NOT be NULL.
- x509certData returns the PEM encoded certificate. x509certData MUST NOT be NULL. If no certificate is stored, then an empty string is returned.

6.7.3 Return Values

- TSS2_RC_SUCCESS: if the function call was a success.
- TSS2_FAPI_RC_BAD_REFERENCE: if context, path or x509certData is NULL.
- TSS2_FAPI_RC_BAD_CONTEXT: if context corruption is detected.
- TSS2_FAPI_RC_KEY_NOT_FOUND: if path does not map to a FAPI entity.
- TSS2_FAPI_RC_BAD_KEY: if the entity at path is not a key, or is a key that is unsuitable for the requested operation.
- TSS2_FAPI_RC_NO_CERTIFICATE: if there is no certificate associated with the key at path.
- TSS2_FAPI_RC_MEMORY: if the FAPI cannot allocate enough memory for internal operations or return parameters.
- TSS2_FAPI_RC_BAD_SEQUENCE: if the synchronous or Async functions are called while the context has another asynchronous operation outstanding, or the Finish function is called while the context does not have an appropriate asynchronous operation outstanding.
- TSS2_FAPI_RC_TRY_AGAIN: (Finish only) if the asynchronous operation is incomplete and the Finish should be re-executed later to check for the final result.

6.8 Fapi_ExportKey

Fapi_ExportKey() will duplicate a key and encrypt it using the public key of a new parent. The exported data will contain the re-wrapped key pointed to by pathOfKeyToDuplicate and then the JSON encoded policy. The exported data SHALL be encoded according to Section 3.6.

6.8.1 Prototype

```
TSS2_RC Fapi_ExportKey (
    FAPI_CONTEXT *context,
    char   const *pathOfKeyToDuplicate,
    char   const *pathToPublicKeyOfNewParent,
    uint8_t     **exportedData);
TSS2_RC Fapi_ExportKey_Async(
    FAPI_CONTEXT *context,
    char   const *pathOfKeyToDuplicate,
    char   const *pathToPublicKeyOfNewParent);
TSS2_RC Fapi_ExportKey_Finish(
    FAPI_CONTEXT *context,
    uint8_t     **exportedData);
```

6.8.2 Parameters

- context is a pointer to the opaque context blob currently being operated on. context MUST NOT be NULL.

- pathOfKeyToDuplicate is the path to the root of the subtree to export. pathOfKeyToDuplicate MUST NOT be NULL.
- pathToPublicKeyOfNewParent is the path to the public key of the new parent. This path MAY reference external public key paths starting with "/ext". pathToPublicKeyOfNewParent MUST NOT be NULL.
- exportedData returns the exported subtree. exportedData MUST NOT be NULL.

6.8.3 Return Values

- TSS2_RC_SUCCESS: if the function call was a success.
- TSS2_FAPI_RC_BAD_REFERENCE: if context, pathOfKeyToDuplicate, pathToPublicKeyOfParent, or exportedData is NULL.
- TSS2_FAPI_RC_BAD_CONTEXT: if context corruption is detected.
- TSS2_FAPI_RC_KEY_NOT_FOUND: if pathOfKeyToDuplicate or pathToPublicKeyOfNewParent does not map to a FAPI object.
- TSS2_FAPI_RC_BAD_KEY: if the object at pathToPublicKeyOfNewParent is not a key, or is a key that is unsuitable for the requested operation.
- TSS2_FAPI_RC_KEY_NOT_DUPLICABLE: if the key at pathOfKeyToDuplicate is not a duplicable key.
- TSS2_FAPI_RC_IO_ERROR: if internal data cannot be loaded.
- TSS2_FAPI_RC_MEMORY: if the FAPI cannot allocate enough memory for internal operations or return parameters.
- TSS2_FAPI_RC_BAD_SEQUENCE: if the synchronous or Async functions are called while the context has another asynchronous operation outstanding, or the Finish function is called while the context does not have an appropriate asynchronous operation outstanding.
- TSS2_FAPI_RC_TRY_AGAIN: (Finish only) if the asynchronous operation is incomplete and the Finish should be re-executed later to check for the final result.

7 Seal commands

7.1 Fapi_CreateSeal

Fapi_CreateSeal() creates a sealed object and stores it in the FAPI metadata store. If no data is provided (i.e. a NULL-pointer) then the TPM generates random data and fills the sealed object.

7.1.1 Prototype

```
TSS2_RC Fapi_CreateSeal(
    FAPI_CONTEXT  *context,
    char    const *path,
    char    const *type,
    size_t         size,
    char    const *policyPath,
    char    const *authValue,
    uint8_t const *data);
TSS2_RC Fapi_CreateSeal_Async(
    FAPI_CONTEXT  *context,
    char   const  *path,
    char   const  *type,
    size_t         size,
    char    const *policyPath,
    char    const *authValue,
    uint8_t const *data);
TSS2_RC Fapi_CreateSeal_Finish(
    FAPI_CONTEXT  *context);
```

7.1.2 Parameters

- context is a pointer to the opaque context blob currently being operated on. context MUST NOT be NULL.
- path is the path to the new key. path MUST NOT be NULL.
- type identifies the intended usage, see Section 3.3. type MAY be NULL.
- size defines the size of the sealed object. size MUST NOT be zero.
- policyPath identifies the policy to be associated with the new key. policyPath MAY be NULL. If policyPath is NULL then no policy will be associated with the key.
- authValue is the new authorization value for the key. authValue MAY be NULL. If authValue is NULL then the authorization value will be the empty string.
- data is the data to be sealed by the TPM. data MAY be NULL.

7.1.3 Return Values

- TSS2_RC_SUCCESS: if the function call was a success.
- TSS2_FAPI_RC_BAD_REFERENCE: if context or path is NULL.
- TSS2_FAPI_RC_BAD_CONTEXT: if context corruption is detected.
- TSS2_FAPI_RC_KEY_NOT_FOUND: if the parent specified in path does not map to a FAPI key.
- TSS2_FAPI_RC_BAD_PATH: if policyPath is non-NULL and does not map to a FAPI policy.
- TSS2_FAPI_RC_PATH_ALREADY_EXISTS: if path already exists.
- TSS2_FAPI_RC_BAD_VALUE: if type is non-NULL and invalid or size is zero.
- TSS2_FAPI_RC_STORAGE_ERROR: if the FAPI storage cannot be updated.
- TSS2_FAPI_RC_IO_ERROR: if the data cannot be saved.
- TSS2_FAPI_RC_MEMORY: if the FAPI cannot allocate enough memory for internal operations or return parameters.

- TSS2_FAPI_RC_BAD_SEQUENCE: if the synchronous or Async functions are called while the context has another asynchronous operation outstanding, or the Finish function is called while the context does not have an appropriate asynchronous operation outstanding.
- TSS2_FAPI_RC_TRY_AGAIN: (Finish only) if the asynchronous operation is incomplete and the Finish should be re-executed later to check for the final result.

7.2 Fapi_Unseal

Fapi_Unseal() unseals data from a sealed object created by Fapi_CreateSeal in the FAPI meta data store.

7.2.1 Prototype

```
TSS2_RC Fapi_Unseal(
    FAPI_CONTEXT   *context,
    char     const *path,
    uint8_t       **data,
    size_t         *size);
TSS2_RC Fapi_Unseal_Async(
    FAPI_CONTEXT   *context,
    char     const *path);
TSS2_RC Fapi_Unseal_Finish(
    FAPI_CONTEXT   *context,
    uint8_t       **data,
    size_t         *size);
```

7.2.2 Parameters

- context is a pointer to the opaque context blob currently being operated on. context MUST NOT be NULL.
- path is the path to the sealed data. path MUST NOT be NULL.
- data is the decrypted data after unsealing. data MAY be NULL.
- size is the size of the decrypted data after unsealing. size MAY be NULL.

7.2.3 Return Values

- TSS2_RC_SUCCESS: if the function call was a success.
- TSS2_FAPI_RC_BAD_REFERENCE: if context or path is NULL.
- TSS2_FAPI_RC_BAD_CONTEXT: if context corruption is detected.
- TSS2_FAPI_RC_BAD_PATH: if path does not point to a sealed data object.
- TSS2_FAPI_RC_STORAGE_ERROR: if the FAPI storage cannot be accessed.
- TSS2_FAPI_RC_IO_ERROR: if the data cannot be saved.
- TSS2_FAPI_RC_MEMORY: if the FAPI cannot allocate enough memory for internal operations or return parameters.
- TSS2_FAPI_RC_BAD_SEQUENCE: if the synchronous or Async functions are called while the context has another asynchronous operation outstanding, or the Finish function is called while the context does not have an appropriate asynchronous operation outstanding.
- TSS2_FAPI_RC_TRY_AGAIN: (Finish only) if the asynchronous operation is incomplete and the Finish should be re-executed later to check for the final result.

8 Policy functions

8.1 Fapi_ExportPolicy

Fapi_ExportPolicy() exports a policy referred to by a path in JSON encoding. The exported policy SHALL be encoded according to TCG TSS 2.0 JSON Policy Language Specification.

8.1.1 Prototype

```
TSS2_RC Fapi_ExportPolicy (
    FAPI_CONTEXT *context,
    char   const *path,
    char         **jsonPolicy);
TSS2_RC Fapi_ExportPolicy_Async(
    FAPI_CONTEXT *context,
    char   const *path);
TSS2_RC Fapi_ExportPolicy_Finish(
    FAPI_CONTEXT *context,
    char         **jsonPolicy);
```

8.1.2 Parameters

- context is a pointer to the opaque context blob currently being operated on. context MUST NOT be NULL.
- path is the path of the new policy. path MUST NOT be NULL.
- jsonPolicy returns the JSON-encoded policy. jsonPolicy MUST NOT be NULL.

8.1.3 Return Values

- TSS2_RC_SUCCESS: if the function call was a success.
- TSS2_FAPI_RC_BAD_REFERENCE: if context or path is NULL.
- TSS2_FAPI_RC_BAD_CONTEXT: if context corruption is detected.
- TSS2_FAPI_RC_BAD_PATH: if path does not map to a FAPI policy.
- TSS2_FAPI_RC_MEMORY: if the FAPI cannot allocate enough memory for internal operations or return parameters.
- TSS2_FAPI_RC_BAD_SEQUENCE: if the synchronous or Async functions are called while the context has another asynchronous operation outstanding, or the Finish function is called while the context does not have an appropriate asynchronous operation outstanding.
- TSS2_FAPI_RC_TRY_AGAIN: (Finish only) if the asynchronous operation is incomplete and the Finish should be re-executed later to check for the final result.

8.2 Fapi_AuthorizePolicy

Fapi_AuthorizePolicy() signs a given policy with a given key such that the policy can be referenced from other policies that contain corresponding PolicyAuthorize elements.

8.2.1 Prototype

```
TSS2_RC Fapi_AuthorizePolicy (
    FAPI_CONTEXT  *context,
    char     const *policyPath,
    char     const *keyPath,
    uint8_t const *policyRef,
    size_t          policyRefSize);
TSS2_RC Fapi_AuthorizePolicy_Async(
    FAPI_CONTEXT *context,
    char     const *policyPath,
    char    const *keyPath,
    uint8_t const *policyRef,
    size_t         policyRefSize);
TSS2_RC Fapi_AuthorizePolicy_Finish(
    FAPI_CONTEXT   *context);
```

8.2.2 Parameters

- context is a pointer to the opaque context blob currently being operated on. context MUST NOT be NULL.
- policyPath is the path of the new policy. policyPath MUST NOT be NULL.
- keyPath is the path of the signing key. keyPath MUST NOT be NULL.
- policyRef is a byte buffer to be included in the signature. policyRef MAY be NULL if policyRefSize is 0.
- policyRefSize is the size of policyRef.

8.2.3 Return Values

- TSS2_RC_SUCCESS: if the function call was a success.
- TSS2_FAPI_RC_BAD_REFERENCE: if context or path is NULL.
- TSS2_FAPI_RC_BAD_CONTEXT: if context corruption is detected.
- TSS2_FAPI_RC_BAD_PATH: if policyPath or keyPath do not map to a FAPI policy or key, respectively.
- TSS2_FAPI_RC_MEMORY: if the FAPI cannot allocate enough memory for internal operations or return parameters.
- TSS2_FAPI_RC_BAD_SEQUENCE: if the synchronous or Async functions are called while the context has another asynchronous operation outstanding, or the Finish function is called while the context does not have an appropriate asynchronous operation outstanding.
- TSS2_FAPI_RC_TRY_AGAIN: (Finish only) if the asynchronous operation is incomplete and the Finish should be re-executed later to check for the final result.

8.3 Fapi_WriteAuthorizeNv

FapiWriteAuthorizeNv() writes the digest value of a policy to an NV index such that this policy can be used in other policies containing a corresponding PolicyAuthorizeNv element. The nameAlg property of the NV index defines the digest algorithm for the policy.

8.3.1 Prototype

```
TSS2_RC Fapi_WriteAuthorizeNv(
    FAPI_CONTEXT   *context,
    char    const  *nvPath,
    char    const  *policyPath);
TSS2_RC Fapi_WriteAuthorizeNv_Async(
    FAPI_CONTEXT *context,
    char    const  *nvPath,
    char    const  *policyPath);
TSS2_RC Fapi_WriteAuthorizeNv_Finish(
    FAPI_CONTEXT   *context);
```

8.3.2 Parameters

- context is a pointer to the opaque context blob currently being operated on. context MUST NOT be NULL.
- nvPath is the path of the NV index. nvPath MUST NOT be NULL.
- policyPath is the path of the new policy. policyPath MUST NOT be NULL.

8.3.3 Return Values

- TSS2_RC_SUCCESS: if the function call was a success.
- TSS2_FAPI_RC_BAD_REFERENCE: if context, nvPath, or policyPath is NULL.

- TSS2_FAPI_RC_BAD_CONTEXT: if context corruption is detected.
- TSS2_FAPI_RC_BAD_PATH: if nvPath, or policyPath do not map to a FAPI nv index or policy, respectively.
- TSS2_FAPI_RC_MEMORY: if the FAPI cannot allocate enough memory for internal operations or return parameters.
- TSS2_FAPI_RC_BAD_SEQUENCE: if the synchronous or Async functions are called while the context has another asynchronous operation outstanding, or the Finish function is called while the context does not have an appropriate asynchronous operation outstanding.
- TSS2_FAPI_RC_TRY_AGAIN: (Finish only) if the asynchronous operation is incomplete and the Finish should be re-executed later to check for the final result.

9 Attestation functions

9.1 Fapi_PcrRead

Fapi_PcrRead() provides a PCR value and corresponding Event log. The PCR bank of the provided PCR index is selected in the cryptographic profile.

9.1.1 Prototype

```
TSS2_RC Fapi_PcrRead(
    FAPI_CONTEXT *context,
    uint32_t      pcrIndex,
    uint8_t     **pcrValue,
    size_t       *pcrValueSize,
    char        **pcrLog);
TSS2_RC Fapi_PcrRead_Async(
    FAPI_CONTEXT *context,
    uint32_t      pcrIndex,);
TSS2_RC Fapi_PcrRead_Finish(
    FAPI_CONTEXT *context,
    uint8_t     **pcrValue,
    size_t       *pcrValueSize,
    char        **pcrLog);
```

9.1.2 Parameters

- context is a pointer to the opaque context blob currently being operated on. context MUST NOT be NULL.
- pcrIndex identifies the PCR to read.
- pcrValue returns PCR digest. pcrValue MAY be NULL.
- pcrValueSize returns the number of bytes in pcrValue. pcrValueSize MAY be NULL.
- pcrLog returns the PCR log for that PCR in the format defined in Section 3.7. pcrLog MAY be NULL.

9.1.3 Return Values

- TSS2_RC_SUCCESS: if the function call was a success.
- TSS2_FAPI_RC_BAD_REFERENCE: if context is NULL.
- TSS2_FAPI_RC_BAD_CONTEXT: if context corruption is detected.
- TSS2_FAPI_RC_BAD_VALUE: if pcrIndex is out of range for the TPM.
- TSS2_FAPI_RC_MEMORY: if the FAPI cannot allocate enough memory for internal operations or return parameters.
- TSS2_FAPI_RC_BAD_SEQUENCE: if the synchronous or Async functions are called while the context has another asynchronous operation outstanding, or the Finish function is called while the context does not have an appropriate asynchronous operation outstanding.
- TSS2_FAPI_RC_TRY_AGAIN: (Finish only) if the asynchronous operation is incomplete and the Finish should be re-executed later to check for the final result.

9.2 Fapi_PcrExtend

Fapi_PcrExtend() extends the data into the PCR listed. The parameter logData is extended into the PCR log. If the logData is NULL, only the PCR extend takes place. All PCRs currently active in the TPM are extended, see TPM2_PCR_Event.

9.2.1 Prototype

```
TSS2_RC Fapi_PcrExtend(
    FAPI_CONTEXT   *context,
    uint32_t        pcr,
    uint8_t  const *data,
    size_t          dataSize,
    char     const *logData);
TSS2_RC Fapi_PcrExtend_Async(
    FAPI_CONTEXT   *context,
    uint32_t        pcr,
    uint8_t  const *data,
    size_t          dataSize,
    char     const *logData);
TSS2_RC Fapi_PcrExtend_Finish(
    FAPI_CONTEXT   *context);
```

9.2.2 Parameters

- context is a pointer to the opaque context blob currently being operated on. context MUST NOT be NULL.
- pcr identifies the PCR to extend.
- data is the event data. This data will be hashed using the respective PCR's hash algorithm. See the TPM2_PCR_Event function of the TPM specification [11]. data MUST NOT be NULL.
- dataSize is the number of bytes in eventData. dataSize MUST NOT be 0.
- logData contains a JSON representation of data to be written to the PCR's event log. logData MAY be NULL.

9.2.3 Return Values

- TSS2_RC_SUCCESS: if the function call was a success.
- TSS2_FAPI_RC_BAD_REFERENCE: if context or data is NULL.
- TSS2_FAPI_RC_BAD_CONTEXT: if context corruption is detected.
- TSS2_FAPI_RC_NO_PCR: if no such PCR exists on this TPM.
- TSS2_FAPI_RC_MEMORY: if the FAPI cannot allocate enough memory for internal operations or return parameters.
- TSS2_FAPI_RC_BAD_SEQUENCE: if the synchronous or Async functions are called while the context has another asynchronous operation outstanding, or the Finish function is called while the context does not have an appropriate asynchronous operation outstanding.
- TSS2_FAPI_RC_TRY_AGAIN: (Finish only) if the asynchronous operation is incomplete and the Finish should be re-executed later to check for the final result.

9.3 Fapi_Quote

Fapi_Quote() performs an attestation using the TPM. The PCR bank of each provided PCR index is set in the cryptographic profile.

9.3.1 Prototype

```
TSS2_RC FAPI_Quote(
    FAPI_CONTEXT   *context,
    uint32_t       *pcrList,
    size_t          pcrListSize,
    char     const *keyPath,
    char     const *quoteType,
    uint8_t  const *qualifyingData,
    size_t          qualifyingDataSize,
    char          **quoteInfo,
    uint8_t       **signature,
    size_t          *signatureSize,
    char           **pcrLog,
    char           **certificate);
TSS2_RC FAPI_Quote_Async(
    FAPI_CONTEXT    *context,
    uint32_t        *pcrList,
    size_t           pcrListSize,
    char     const *keyPath,
    char     const *quoteType,
    uint8_t  const *qualifyingData,
    size_t           qualifyingDataSize);
TSS2_RC FAPI_Quote_Finish(
    FAPI_CONTEXT    *context,
    char           **quoteInfo,
    uint8_t        **signature,
    size_t          *signatureSize,
    char           **pcrLog,
    char           **certificate);
```

9.3.2 Parameters

- context is a pointer to the opaque context blob currently being operated on. context MUST NOT be NULL.
- pcrList is an array holding the PCR indices to quote against. pcrList MUST NOT be NULL.
- pcrListSize is the size of pcrList.
- keyPath identifies the signing key. keyPath MUST NOT be NULL.
- quoteType identifies the type of attestation to be performed. quoteType MUST be NULL (which implies "quote") or "quote" (case insensitive). Note: Future versions may allow other values for other types ofattestations.
- qualifyingData is a nonce provided by the caller to ensure freshness of the signature. qualifyingData MAY be NULL if qualifyingDataSize is 0.
- qualifyingDataSize is the number of bytes in qualifyingData.
- quoteInfo returns a JSON-encoded structure holding the inputs to the quote operation. This includes the digest value and PCR values. quoteInfo MUST NOT be NULL.
- signature returns the signature over the quoted material. signature MUST NOT be NULL.
- signatureSize returns the number of bytes in signature. signatureSize MAY be NULL.
- pcrLog returns the PCR log for the chosen PCR in the format defined in Section 3.7. pcrLog MAY be NULL.
- certificate is the certificate associated with keyPath in PEM format. certificate MAY be NULL.

9.3.3 Return Values

- TSS2_RC_SUCCESS: if the function call was a success.
- TSS2_FAPI_RC_BAD_REFERENCE: if context, pcrList, keyPath, quoteInfo, or signature is NULL.
- TSS2_FAPI_RC_BAD_CONTEXT: if context corruption is detected.
- TSS2_FAPI_RC_KEY_NOT_FOUND: if path does not map to a FAPI entity.
- TSS2_FAPI_RC_BAD_KEY: if the entity at path is not a key, or is a key that is unsuitable for the requested operation.
- TSS2_FAPI_RC_BAD_VALUE: if qualifyingData is invalid or qualifyingDataSize is 0.
- TSS2_FAPI_RC_MEMORY: if the FAPI cannot allocate enough memory for internal operations or return parameters.

- TSS2_FAPI_RC_BAD_SEQUENCE: if the synchronous or Async functions are called while the context has another asynchronous operation outstanding, or the Finish function is called while the context does not have an appropriate asynchronous operation outstanding.
- TSS2_FAPI_RC_TRY_AGAIN: (Finish only) if the asynchronous operation is incomplete and the Finish should be re-executed later to check for the final result.

9.4 Fapi_VerifyQuote

Fapi_VerifyQuote() verifies that the data returned by a quote is valid. This includes

- Reconstructing the quoteInfo's PCR values from the eventLog (if an eventLog was provided)
- Verifying the quoteInfo using the signature and the publicKeyPath

An application using Fapi_VerifyQuote() will further have to

- Assess the publicKey's trustworthiness
- Assess the eventLog entries' trustworthiness

9.4.1 Prototype

```
TSS2_RC Fapi_VerifyQuote(
    FAPI_CONTEXT  *context,
    char    const *publicKeyPath,
    uint8_t const *qualifyingData,
    size_t         qualifyingDataSize,
    char    const *quoteInfo,
    uint8_t const *signature,
    size_t         signatureSize,
    char    const *pcrLog);
TSS2_RC Fapi_VerifyQuote_Async(
    FAPI_CONTEXT  *context,
    char    const *publicKeyPath,
    uint8_t const *qualifyingData,
    size_t         qualifyingDataSize,
    char    const *quoteInfo,
    uint8_t const *signature,
    size_t         signatureSize,
    char    const *pcrLog);
TSS2_RC Fapi_VerifyQuote_Finish(
    FAPI_CONTEXT  *context);
```

9.4.2 Parameters

- context is a pointer to the opaque context blob currently being operated on. context MUST NOT be NULL.
- publicKeyPath identifies the signing key. publicKeyPath MUST NOT be NULL. publicKeyPath MAY be a path to the public key hierarchy /ext.
- qualifyingData is a nonce provided by the caller to ensure freshness of the signature. qualifyingData MAY be NULL if qualifyingDataSize is 0.
- qualifyingDataSize is the number of bytes in qualifyingData.
- quoteInfo the JSON-encoded structure holding the inputs to the quote operation. This includes the digest value and PCR values. quoteInfo MUST NOT be NULL.
- signature the signature over the quoted material. signature MUST NOT be NULL.
- signatureSize is the number of bytes in signature.
- pcrLog returns the PCR log for the chosen PCR in the format defined in Section 3.7. pcrLog MAY be NULL.

9.4.3 Return Values

- TSS2_RC_SUCCESS: if the function call was a success.
- TSS2_FAPI_RC_BAD_REFERENCE: if context, publicKeyPath, quoteInfo or signature is NULL.
- TSS2_FAPI_RC_BAD_CONTEXT: if context corruption is detected.
- TSS2_FAPI_RC_KEY_NOT_FOUND: if path does not map to a FAPI entity.
- TSS2_FAPI_RC_BAD_KEY: if the entity at path is not a key, or is a key that is unsuitable for the requested operation.
- TSS2_FAPI_RC_BAD_VALUE: if qualifyingData is invalid or qualifyingDataSize is 0.
- TSS2_FAPI_RC_MEMORY: if the FAPI cannot allocate enough memory for internal operations or return parameters.
- TSS2_FAPI_RC_BAD_SEQUENCE: if the synchronous or Async functions are called while the context has another asynchronous operation outstanding, or the Finish function is called while the context does not have an appropriate asynchronous operation outstanding.
  TSS2_FAPI_RC_TRY_AGAIN: (Finish only) if the asynchronous operation is incomplete and the Finish should be re-executed later to check for the final result.

10 NV functions

10.1 Fapi_CreateNv

Fapi_CreateNv creates an NV index in the TPM. The path is constructed as described in section 3.2.2. The type field is described in section 3.3.

10.1.1 Prototype

```
TSS2_RC Fapi_CreateNv(
    FAPI_CONTEXT *context,
    char   const *path,
    char   const *type,
    size_t        size,
    char   const *policyPath,
    char   const *authValue);
TSS2_RC Fapi_CreateNv_Async(
    FAPI_CONTEXT *context,
    char   const *path,
    char   const *type,
    size_t        size,
    char   const *policyPath,
    char   const *authValue);
TSS2_RC Fapi_CreateNv_Finish(
    FAPI_CONTEXT *context);
```

10.1.2 Parameters

- context is a pointer to the opaque context blob currently being operated on. context MUST NOT be NULL.
- path is the path to the new key. path MUST NOT be NULL.
- type identifies the intended usage, see Section 3.3. type MAY be NULL.
- size is the size in bytes of the NV index to be created. size MAY be zero if the size is determined by the type; e.g. an NV index of type counter has a size of 8 bytes.
- policyPath identifies the policy to be associated with the new key. policyPath MAY be NULL. If policyPath is NULL then no policy will be associated with the key.
- authValue is the new authorization value for the key. authValue MAY be NULL. If authValue is NULL then the authorization value will be the empty string.

10.1.3 Return Values

- TSS2_RC_SUCCESS: if the function call was a success.
- TSS2_FAPI_RC_BAD_REFERENCE: if context or path is NULL.
- TSS2_FAPI_RC_BAD_CONTEXT: if context corruption is detected.
- TSS2_FAPI_RC_BAD_PATH: if policyPath is non-NULL and does not map to a FAPI policy or if path does not refer to a valid NV index path.
- TSS2_FAPI_RC_PATH_ALREADY_EXISTS: if path already exists.
- TSS2_FAPI_RC_BAD_VALUE: if type is non-NULL and invalid or does not match the size.
- TSS2_FAPI_RC_STORAGE_ERROR: if the FAPI storage cannot be updated.
- TSS2_FAPI_RC_IO_ERROR: if the data cannot be saved.
- TSS2_FAPI_RC_MEMORY: if the FAPI cannot allocate enough memory for internal operations or return parameters.
- TSS2_FAPI_RC_BAD_SEQUENCE: if the synchronous or Async functions are called while the context has another asynchronous operation outstanding, or the Finish function is called while the context does not have an appropriate asynchronous operation outstanding.

- TSS2_FAPI_RC_TRY_AGAIN: (Finish only) if the asynchronous operation is incomplete and the Finish should be re-executed later to check for the final result.

10.2 Fapi_NvRead

Fapi_NvRead() reads the entire data from an NV index of the TPM. The FAPI will automatically perform multiple read operations with the TPM if the NV index is larger than the TPM's TPM2_MAX_NV_BUFFER_SIZE.

10.2.1 Prototype

```
TSS2_RC Fapi_NvRead(
    FAPI_CONTEXT   *context,
    char     const *nvPath,
    uint8_t       **data,
    size_t         *size,
    char          **logData);
TSS2_RC Fapi_NvRead_Async(
    FAPI_CONTEXT   *context,
    char     const *nvPath);
TSS2_RC Fapi_NvRead_Finish(
    FAPI_CONTEXT   *context,
    uint8_t       **data,
    size_t         *size,
    char          **logData);
```

10.2.2 Parameters

- context is a pointer to the opaque context blob currently being operated on. context MUST NOT be NULL.
- nvPath identifies the NV space to read. nvPath MUST NOT be NULL.
- data returns the value read from the NV space. data MUST NOT be NULL.
- size returns the number of bytes read. size MAY be NULL.
- logData returns the JSON encoded log, if the NV index is of type “extend” and an empty string otherwise. logData MAY be NULL.

10.2.3 Return Values

- TSS2_RC_SUCCESS: if the function call was a success.
- TSS2_FAPI_RC_BAD_REFERENCE: if context, nvPath or data is NULL.
- TSS2_FAPI_RC_BAD_CONTEXT: if context corruption is detected.
- TSS2_FAPI_RC_BAD_PATH: if nvPath is not found.
- TSS2_FAPI_RC_AUTHORIZATION_FAILED: if authorization fails.
- TSS2_FAPI_RC_AUTHORIZATION_UNKNOWN: if the authentication method could not be identified.
- TSS2_FAPI_RC_NV_NOT_READABLE: if the NV is not a readable index.
- TSS2_FAPI_RC_MEMORY: if the FAPI cannot allocate enough memory for internal operations or return parameters.
- TSS2_FAPI_RC_BAD_SEQUENCE: if the synchronous or Async functions are called while the context has another asynchronous operation outstanding, or the Finish function is called while the context does not have an appropriate asynchronous operation outstanding.
- TSS2_FAPI_RC_TRY_AGAIN: (Finish only) if the asynchronous operation is incomplete and the Finish should be re-executed later to check for the final result.

10.3 Fapi_NvWrite

Fapi_NvWrite() writes data to a “regular” (not pin, extend or counter) NV index. Only the full index can be written. Partial writes are not supported. If the provided data is smaller than the NV index’s size, then it is padded up with zero bytes at the end. The FAPI will automatically perform multiple write operations with the TPM if the input buffer is larger than the TPM's TPM2_MAX_NV_BUFFER_SIZE.

10.3.1 Prototype

```
TSS2_RC Fapi_NvWrite(
    FAPI_CONTEXT  *context,
    char    const *nvPath,
    uint8_t const *data,
    size_t         size);
TSS2_RC Fapi_NvWrite_Async(
    FAPI_CONTEXT  *context,
    char    const *nvPath,
    uint8_t const *data,
    size_t         size);
TSS2_RC Fapi_NvWrite_Finish(
    FAPI_CONTEXT  *context);
```

10.3.2 Parameters

- context is a pointer to the opaque context blob currently being operated on. context MUST NOT be NULL.
- nvPath identifies the NV space to write. nvPath MUST NOT be NULL.
- data is the data to write to the NV space. data MUST NOT be NULL.
- size is the size of the data buffer in bytes.

10.3.3 Return Values

- TSS2_RC_SUCCESS: if the function call was a success.
- TSS2_FAPI_RC_BAD_REFERENCE: if context, nvPath, or data is NULL.
- TSS2_FAPI_RC_BAD_CONTEXT: if context corruption is detected.
- TSS2_FAPI_RC_BAD_PATH: if nvPath is not found.
- TSS2_FAPI_RC_NV_EXCEEDED: if the NV index is not large enough for the data to be written.
- TSS2_FAPI_RC_NV_WRONG_TYPE: if the NV index is an extendable index.
- TSS2_FAPI_RC_NV_NOT_WRITEABLE: if the NV index is not a writeable index.
- TSS2_FAPI_RC_POLICY_UNKNOWN: if the policy is unknown.
- TSS2_FAPI_RC_MEMORY: if the FAPI cannot allocate enough memory for internal operations or return parameters.
- TSS2_FAPI_RC_BAD_SEQUENCE: if the synchronous or Async functions are called while the context has another asynchronous operation outstanding, or the Finish function is called while the context does not have an appropriate asynchronous operation outstanding.
- TSS2_FAPI_RC_TRY_AGAIN: (Finish only) if the asynchronous operation is incomplete and the Finish should be re-executed later to check for the final result.

10.4 Fapi_NvExtend

Fapi_NvExtend() performs an extend options on an NV index of type extend (i.e. an NV index that behaves similarly to a PCR).

10.4.1 Prototype

```
TSS2_RC Fapi_NvExtend(
    FAPI_CONTEXT  *context,
    char    const *nvPath,
    uint8_t const *data,
    size_t         dataSize,
    char    const *logData);
TSS2_RC Fapi_NvExtend_Async(
    FAPI_CONTEXT  *context,
    char    const *nvPath,
    uint8_t const *data,
    size_t         dataSize,
    char    const *logData);
TSS2_RC Fapi_NvExtend_Finish(
    FAPI_CONTEXT  *context);
```

10.4.2 Parameters

- context is a pointer to the opaque context blob currently being operated on. context MUST NOT be NULL.
- nvPath identifies the NV space to write. nvPath MUST NOT be NULL.
- data is the data to be extended into the NV space. data MUST NOT be NULL.
- dataSize is the size of the data buffer in bytes. dataSize MUST be smaller or equal to 1024 bytes.
- logData contains a JSON representation of data to be written to the PCR's event log. logData MAY be NULL.

10.4.3 Return Values

- TSS2_RC_SUCCESS: if the function call was a success.
- TSS2_FAPI_RC_BAD_REFERENCE: if context, nvPath, or data is NULL.
- TSS2_FAPI_RC_BAD_CONTEXT: if context corruption is detected.
- TSS2_FAPI_RC_BAD_PATH: if nvPath is not found.
- TSS2_FAPI_RC_NV_WRONG_TYPE: if the NV is not an extendable index. The NV index type must be pcr (which maps to TPM_NT_EXTEND).
- TSS2_FAPI_RC_POLICY_UNKNOWN: if the policy is unknown.
- TSS2_FAPI_RC_MEMORY: if the FAPI cannot allocate enough memory for internal operations or return parameters.
- TSS2_FAPI_RC_BAD_SEQUENCE: if the synchronous or Async functions are called while the context has another asynchronous operation outstanding, or the Finish function is called while the context does not have an appropriate asynchronous operation outstanding.
- TSS2_FAPI_RC_TRY_AGAIN: (Finish only) if the asynchronous operation is incomplete and the Finish should be re-executed later to check for the final result.

10.5 Fapi_NvIncrement

Fapi_NvIncrement() increments by 1 an NV index that is of type counter.

10.5.1 Prototype

```
TSS2_RC Fapi_NvIncrement(
    FAPI_CONTEXT *context,
    char   const *nvPath);
TSS2_RC Fapi_NvIncrement_Async(
    FAPI_CONTEXT *context,
    char   const *nvPath);
TSS2_RC Fapi_NvIncrement_Finish(
    FAPI_CONTEXT *context);
```

10.5.2 Parameters

- context is a pointer to the opaque context blob currently being operated on. context MUST NOT be NULL.
- nvPath identifies the NV space to increment. nvPath MUST NOT be NULL.

10.5.3 Return Values

- TSS2_RC_SUCCESS: if the function call was a success.
- TSS2_FAPI_RC_BAD_REFERENCE: if context or nvPath is NULL.
- TSS2_FAPI_RC_BAD_CONTEXT: if context corruption is detected.
- TSS2_FAPI_RC_BAD_PATH: if nvPath is not found.
- TSS2_FAPI_RC_NV_WRONG_TYPE: if the NV is not a counter index. The NV index type MUST be of type "counter" (which maps to TPM_NT_COUNTER).
- TSS2_FAPI_RC_NV_NOT_WRITEABLE: if the NV is not a writeable index.
- TSS2_FAPI_RC_POLICY_UNKNOWN: if the policy is unknown.
- TSS2_FAPI_RC_MEMORY: if the FAPI cannot allocate enough memory for internal operations or return parameters.
- TSS2_FAPI_RC_BAD_SEQUENCE: if the synchronous or Async functions are called while the context has another asynchronous operation outstanding, or the Finish function is called while the context does not have an appropriate asynchronous operation outstanding.
- TSS2_FAPI_RC_TRY_AGAIN: (Finish only) if the asynchronous operation is incomplete and the Finish should be re-executed later to check for the final result.

10.6 Fapi_NvSetBits

Fapi_NvSetBits() sets bits in an NV Index that was created as a bit field. Any number of bits from 0 to 64 may be SET. The contents of bitmap are ORed with the current contents of the NV Index.

10.6.1 Prototype

```
TSS2_RC Fapi_NvSetBits(
    FAPI_CONTEXT *context,
    char   const *nvPath,
    uint64_t      bitmap);
TSS2_RC Fapi_NvSetBits_Async(
    FAPI_CONTEXT *context,
    char   const *nvPath,
    uint64_t      bitmap);
TSS2_RC Fapi_NvSetBits_Finish(
    FAPI_CONTEXT *context);
```

10.6.2 Parameters

- context is a pointer to the opaque context blob currently being operated on. context MUST NOT be NULL.
- nvPath identifies the NV space to write. nvPath MUST NOT be NULL.
- bitmap is a mask indicating which bits to set in the NV space.

10.6.3 Return Values

- TSS2_RC_SUCCESS: if the function call was a success.
- TSS2_FAPI_RC_BAD_REFERENCE: if context or nvPath is NULL.
- TSS2_FAPI_RC_BAD_CONTEXT: if context corruption is detected.
- TSS2_FAPI_RC_BAD_PATH: if nvPath is not found.
- TSS2_FAPI_RC_NV_WRONG_TYPE: if the NV is not a bitmap index. The NV index type must be TPM2_NT_BITS.
- TSS2_FAPI_RC_NV_NOT_WRITEABLE: if the NV is not a writeable index.
- TSS2_FAPI_RC_POLICY_UNKNOWN: if the policy is unknown.
- TSS2_FAPI_RC_MEMORY: if the FAPI cannot allocate enough memory for internal operations or return parameters.

- TSS2_FAPI_RC_BAD_SEQUENCE: if the synchronous or Async functions are called while the context has another asynchronous operation outstanding, or the Finish function is called while the context does not have an appropriate asynchronous operation outstanding.
- TSS2_FAPI_RC_TRY_AGAIN: (Finish only) if the asynchronous operation is incomplete and the Finish should be re-executed later to check for the final result.

11 FAPI Callbacks

11.1 Fapi_SetAuthCB

Fapi_SetAuthCB() registers an application-defined function as a callback to allow the TSS to get authorization values from the application. The callback and user data pointers are saved within the context and the callback is invoked whenever an authorization value is needed. The userData parameter is a pointer to application-defined data that will be passed to the callback each time it is invoked. The userData is intended to hold application-specific state as needed, and may be NULL if no such state is required. The callback is cleared if the callback function pointer is NULL, and any attempt to use a policy that requires user-supplied authorization will fail.

11.1.1 Prototype

```
TSS2_RC Fapi_SetAuthCB(
    FAPI_CONTEXT *context,
    Fapi_CB_Auth  callback,
    void         *userData);
```

11.1.2 Parameters

- context is a pointer to the opaque context blob currently being operated on. context MUST NOT be NULL.
- callback is the pointer to the callback function for auth values. callback MUST NOT be NULL.
- userData is a pointer that is provided to all callback invocations. userData MAY be NULL.

11.1.3 Return Values

- TSS2_RC_SUCCESS: if the function call was a success.
- TSS2_FAPI_RC_BAD_REFERENCE: if context or callback is NULL.
- TSS2_FAPI_RC_BAD_CONTEXT: if context corruption is detected.
- TSS2_FAPI_RC_MEMORY: if the FAPI cannot allocate enough memory for internal operations or return parameters.
- TSS2_FAPI_RC_BAD_SEQUENCE: if this function is called while the context has another asynchronous operation outstanding.

11.1.4 Fapi_CB_Auth

The Fapi_CB_Auth type describes a callback function prototype that will return an authValue from the application to FAPI when invoced. The programmer is responsible for allocating memory for the auth buffer, but it is the responsibility of FAPI to free it. The FAPI is responsible for creating the HMAC value from the authValue to provide authentication to the TPM.

11.1.4.1 Prototype

```
typedef TSS2_RC (*Fapi_CB_Auth)(
    FAPI_CONTEXT *context,
    char   const *description,
    char        **auth,
    void         *userData);
```

11.1.4.2 Parameters

- context is a pointer to the opaque context blob currently being operated on. context MUST NOT be NULL.
- description is a user readable description of the authorization value requested. description MUST NOT be NULL.
- auth is the authorization value. auth MUST NOT be NULL.
- userData is the same pointer passed in the userData parameter during Fapi_SetAuthCB.

11.1.4.3 Return Values

- TSS2_RC_SUCCESS: if the function call was a success.

- TSS2_FAPI_RC_TRY_AGAIN: if the function needs to be invoked again.
- TSS2_FAPI_RC_CB_FAILURE: if the authorization failed.

11.2 Fapi_SetBranchCB

Fapi_SetBranchCB() registers a callback that will be invoked whenever the FAPI has to decide which branch of a Policy-OR policy to use to authorize a particular FAPI operation. Since the FAPI does not know which branch is appropriate, the application-defined callback is used to make the choice for the FAPI. The callback and user data pointers are associated with the context. The userData parameter is a pointer to application-defined data that will be passed to the callback each time it is invoked. The userData is intended to hold application-specific state as needed, and may be NULL if no such state is required. The callback is cleared if the callback function pointer is NULL, and any attempt to use a policy that includes an OR branch MAY fail.

11.2.1 Prototype

```
TSS2_RC Fapi_SetBranchCB(
    FAPI_CONTEXT   *context,
    Fapi_CB_Branch  callback,
    void           *userData);
```

11.2.2 Parameters

- context is a pointer to the opaque context blob currently being operated on. context MUST NOT be NULL
- callback is the pointer to the callback function for branch selection. callback MUST NOT be NULL.
- userData is a pointer that is provided to all callback invocations. userData MAY be NULL.

11.2.3 Return Values

- TSS2_RC_SUCCESS: if the function call was a success.
- TSS2_FAPI_RC_BAD_REFERENCE: if context or callback is NULL.
- TSS2_FAPI_RC_BAD_CONTEXT: if context corruption is detected.
- TSS2_FAPI_RC_MEMORY: if the FAPI cannot allocate enough memory for internal operations or return parameters.
- TSS2_FAPI_RC_BAD_SEQUENCE: if this function is called while the context has another asynchronous operation outstanding.

11.2.4 Fapi_CB_Branch

The Fapi_CB_Branch type describes a callback function prototype that returns a branch choice during policy evaluation. Such choices take place when a policy contains a PolicyOR (with more than one branch), or a PolicyAuthorize (which has more than one approved policy).

FAPI will invoke the callback with the following information:

- An arbitrary pointer supplied by the application when the callback was registered
- The number of policies/branches to choose from
- The names associated with those policies/branches
- The description of the entity being authorized

The selectedBranch returned is the index within the branchName array and starts with 0.

11.2.4.1 Prototype

```
typedef TSS2_RC (*Fapi_CB_Branch)(
    FAPI_CONTEXT *context,
    char   const *description,
    char  const **branchNames,
    size_t        numBranches,
    size_t        *selectedBranch,
    void          *userData);
```

11.2.4.2 Parameters

- context is a pointer to the opaque context blob currently being operated on. context MUST NOT be NULL.
- description is a human readable description from the PolicyOR statement. description MUST NOT be NULL.
- branchName is a list of pointers to human readable names for the branches as from the PolicyOR statement. branchName MUST NOT be NULL.
- numBranches is the number of branches.
- selectedBranch returns the index of the selected branch. selectedBranch MUST NOT be NULL.
- userData is the same pointer passed in the userData parameter during Fapi_SetAuthCB.

11.2.4.3 Return Values

- TSS2_RC_SUCCESS: if the function call was a success.
- TSS2_FAPI_RC_TRY_AGAIN: if the function needs to be invoked again.
- TSS2_FAPI_RC_CB_FAILURE: if the branch selection failed.

11.3 Fapi_SetSignCB

Fapi_SetSignCB() registers an application-defined function as a callback to allow the FAPI to get signatures authorizing use of TPM objects. The callback and user data pointers are saved within the context and the callback is invoked whenever a signature-based policy is used to authorize a TPM command. The userData parameter is a pointer to application-defined data that will be passed to the callback each time it is invoked. The userData is intended to hold application-specific state as needed, and may be NULL if no such state is required. The callback is cleared if the callback function pointer is NULL, and any attempt to use a policy that requires a signature-based authorization will fail.

11.3.1 Prototype

```
TSS2_RC Fapi_SetSignCB(
    FAPI_CONTEXT *context,
    Fapi_CB_Sign  callback,
    void         *userData);
```

11.3.2 Parameters

- context is a pointer to the opaque context blob currently being operated on. context MUST NOT be NULL.
- callback is the pointer to the callback function for signature based authentication. callback MUST NOT be NULL.
- userData is a pointer that is provided to all callback invocations. userData MAY be NULL.

11.3.3 Return Values

- TSS2_RC_SUCCESS: if the function call was a success.
- TSS2_FAPI_RC_BAD_REFERENCE: if context or callback is NULL.
- TSS2_FAPI_RC_BAD_CONTEXT: if context corruption is detected.
- TSS2_FAPI_RC_MEMORY: if the FAPI cannot allocate enough memory for internal operations or return parameters.
- TSS2_FAPI_RC_BAD_SEQUENCE: if this function is called while the context has another asynchronous operation outstanding.

11.3.4 Fapi_CB_Sign

The Fapi_CB_Branch type describes a callback function prototype that returns a signature from the application to the FAPI. The purpose of this signature is to authorize a policy execution containing a PolicySigned element.

The publicKey is the name of the public key. The publicKeyHint is a human readable string from the policy, which helps the user to identify the correct key to be used.

11.3.4.1 Prototype

```
typedef TSS2_RC (*Fapi_CB_Sign)(
    FAPI_CONTEXT  *context,
    char    const *description,
    uint8_t const *publicKey,
    char    const *publicKeyHint,
    uint32_t       hashAlg,
    uint8_t const *dataToSign,
    size_t         dataToSignSize,
    uint8_t      **signature,
    size_t        *signatureSize,
    void          *userData);
```

11.3.4.2 Parameters

- context is a pointer to the opaque context blob currently being operated on. context MUST NOT be NULL
- publicKey is the public key that will be used by the TPM to verify the signature in PEM encoding. publicKey MUST NOT be NULL.
- publicKeyHint is human readable information, regarding the public key to be used. publicKeyHint MAY be NULL.
- hashAlg is the hash algorithm to be used during signing.
- dataToSign is the data to be hashed and signed by the application. dataToSign MUST NOT be NULL.
- dataToSignSize is the size of dataToSign.
- signature returns the signature over dataToSign. signature MUST NOT be NULL.
- signatureSize returns the size of signature. signatureSize MUST NOT be NULL.
- userData is the same pointer passed in the userData parameter during Fapi_SetAuthCB.

11.3.4.3 Return Values

- TSS2_RC_SUCCESS: if the function call was a success.
- TSS2_FAPI_RC_TRY_AGAIN: if the function needs to be invoked again.
- TSS2_FAPI_RC_CB_FAILURE: if the signature operation failed.

Appendix: HEADER FILE

```
/* SPDX-License-Identifier: BSD-2 */
/***********************************************************************
 * Copyright 2017-2018, Fraunhofer SIT sponsored by Infineon Technologies AG
 * All rights reserved.
 ***********************************************************************/
#ifndef TSS2_FAPI_H
#define TSS2_FAPI_H

#include "tss2_esys.h"

#ifdef __cplusplus
extern "C" {
#endif

/* Type definitions */

typedef struct FAPI_CONTEXT FAPI_CONTEXT;

/* Context functions */

TSS2_RC Fapi_Initialize(
    FAPI_CONTEXT  **context,
    char     const *uri);

TSS2_RC Fapi_Initialize_Async(
    FAPI_CONTEXT  **context,
    char     const *uri);

TSS2_RC Fapi_Initialize_Finish(
    FAPI_CONTEXT  **context);

void Fapi_Finalize(
    FAPI_CONTEXT  **context);

void Fapi_Free(
    void           *ptr);

TSS2_RC Fapi_GetInfo(
    FAPI_CONTEXT   *context,
    char          **info);

TSS2_RC Fapi_GetInfo_Async(
    FAPI_CONTEXT   *context);

TSS2_RC Fapi_GetInfo_Finish(
    FAPI_CONTEXT   *context,
    char          **info);
```

```
/* General functions */

TSS2_RC Fapi_Provision(
    FAPI_CONTEXT   *context,
    char     const *policyPathEh,
    char     const *authValueEh,
    char     const *policyPathSh,
    char     const *authValueSh,
    char     const *authValueLockout);

TSS2_RC Fapi_Provision_Async(
    FAPI_CONTEXT   *context,
    char     const *policyPathEh,
    char     const *authValueEh,
    char     const *policyPathSh,
    char     const *authValueSh,
    char     const *authValueLockout);

TSS2_RC Fapi_Provision_Finish(
    FAPI_CONTEXT   *context);

TSS2_RC Fapi_GetPlatformCertificates(
    FAPI_CONTEXT   *context,
    uint8_t       **certificates,
    size_t         *certificatesSize);

TSS2_RC Fapi_GetPlatformCertificates_Async(
    FAPI_CONTEXT   *context);

TSS2_RC Fapi_GetPlatformCertificates_Finish(
    FAPI_CONTEXT   *context,
    uint8_t       **certificates,
    size_t         *certificatesSize);

TSS2_RC Fapi_TPM_GetRandom(
    FAPI_CONTEXT   *context,
    size_t          numBytes,
    uint8_t       **data);

TSS2_RC Fapi_TPM_GetRandom_Async(
    FAPI_CONTEXT   *context,
    size_t          numBytes);

TSS2_RC Fapi_TPM_GetRandom_Finish(
    FAPI_CONTEXT   *context,
    uint8_t       **data);

TSS2_RC Fapi_Import(
    FAPI_CONTEXT   *context,
    char     const *path,
    char     const *importData);

TSS2_RC Fapi_Import_Async(
    FAPI_CONTEXT   *context,
    char     const *path,
    char     const *importData);

TSS2_RC Fapi_Import_Finish(
    FAPI_CONTEXT   *context);

TSS2_RC Fapi_List(
    FAPI_CONTEXT   *context,
    char     const *searchPath,
    char          **pathList);

TSS2_RC Fapi_List_Async(
    FAPI_CONTEXT   *context,
    char     const *searchPath);

TSS2_RC Fapi_List_Finish(
    FAPI_CONTEXT   *context,
    char          **pathlist);

TSS2_RC Fapi_Delete(
    FAPI_CONTEXT   *context,
    char     const *path);

TSS2_RC Fapi_Delete_Async(
    FAPI_CONTEXT   *context,
    char     const *path);

TSS2_RC Fapi_Delete_Finish(
    FAPI_CONTEXT   *context);

TSS2_RC Fapi_ChangeAuth(
    FAPI_CONTEXT   *context,
    char     const *entityPath,
    char     const *authValue);

TSS2_RC Fapi_ChangeAuth_Async(
    FAPI_CONTEXT   *context,
    char     const *entityPath,
    char     const *authValue);

TSS2_RC Fapi_ChangeAuth_Finish(
    FAPI_CONTEXT   *context);

TSS2_RC Fapi_SetDescription(
    FAPI_CONTEXT   *context,
    char     const *path,
    char     const *description);

TSS2_RC Fapi_SetDescription_Async(
    FAPI_CONTEXT   *context,
    char     const *path,
    char     const *description);

TSS2_RC Fapi_SetDescription_Finish(
    FAPI_CONTEXT   *context);

TSS2_RC Fapi_GetDescription(
    FAPI_CONTEXT   *context,
    char     const *path,
    char          **description);

TSS2_RC Fapi_GetDescription_Async(
    FAPI_CONTEXT   *context,
    char     const *path);

TSS2_RC Fapi_GetDescription_Finish(
    FAPI_CONTEXT   *context,
    char          **description);

TSS2_RC Fapi_SetAppData(
    FAPI_CONTEXT   *context,
    char     const *path,
    uint8_t  const *appData,
    size_t          appDataSize);

TSS2_RC Fapi_SetAppData_Async(
    FAPI_CONTEXT   *context,
    char     const *path,
    uint8_t  const *appData,
    size_t          appDataSize);

TSS2_RC Fapi_SetAppData_Finish(
    FAPI_CONTEXT   *context);

TSS2_RC Fapi_GetAppData(
    FAPI_CONTEXT   *context,
    char     const *path,
    uint8_t       **appData,
    size_t         *appDataSize);

TSS2_RC Fapi_GetAppData_Async(
    FAPI_CONTEXT   *context,
    char     const *path);

TSS2_RC Fapi_GetAppData_Finish(
    FAPI_CONTEXT   *context,
    uint8_t       **appData,
    size_t         *appDataSize);

TSS2_RC Fapi_GetTPMBlobs(
    FAPI_CONTEXT   *context,
    char     const *path,
    uint8_t       **tpm2bPublic,
    size_t         *tpm2bPublicSize,
    uint8_t       **tpm2bPrivate,
    size_t         *tpm2bPrivateSize
    char          **policy);

TSS2_RC Fapi_GetTPMBlobs_Async(
    FAPI_CONTEXT   *context,
    char     const *path);

TSS2_RC Fapi_GetTPMBlobs_Finish(
    FAPI_CONTEXT   *context,
    uint8_t       **tpm2bPublic,
    size_t         *tpm2bPublicSize,
    uint8_t       **tpm2bPrivate,
    size_t         *tpm2bPrivateSize,
    char          **policy);

/* Key functions */

TSS2_RC Fapi_CreateKey(
    FAPI_CONTEXT   *context,
    char     const *keyPath,
    char     const *type,
    char     const *policyPath,
    char     const *authvalue);

TSS2_RC Fapi_CreateKey_Async(
    FAPI_CONTEXT   *context,
    char     const *keyPath,
    char     const *type,
    char     const *policyPath,
    char     const *authvalue);

TSS2_RC Fapi_CreateKey_Finish(
    FAPI_CONTEXT   *context);

TSS2_RC Fapi_Sign(
    FAPI_CONTEXT   *context,
    char     const *keyPath,
    char     const *padding,
    uint8_t  const *digest,
    size_t          digestSize,
    uint8_t       **signature,
    size_t          *signatureSize,
    char           **publicKey,
    char           **certificate);

TSS2_RC Fapi_Sign_Async(
    FAPI_CONTEXT    *context,
    char      const *keyPath,
    uint8_t   const *digest,
    size_t           digestSize);

TSS2_RC Fapi_Sign_Finish(
    FAPI_CONTEXT    *context,
    uint8_t        **signature,
    size_t          *signatureSize,
    char           **publicKey,
    char           **certificate);

TSS2_RC Fapi_VerifySignature(
    FAPI_CONTEXT    *context,
    char      const *keyPath,
    uint8_t   const *signature,
    size_t           signatureSize,
    uint8_t   const *digest,
    size_t           digestSize);

TSS2_RC Fapi_VerifySignature_Async(
    FAPI_CONTEXT    *context,
    char      const *keyPath,
    uint8_t   const *signature,
    size_t           signatureSize,
    uint8_t   const *digest,
    size_t           digestSize);

TSS2_RC Fapi_VerifySignature_Finish(
    FAPI_CONTEXT    *context);

TSS2_RC Fapi_Encrypt(
    FAPI_CONTEXT    *context,
    char      const *keyPath,
    char      const *policyPath,
    uint8_t   const *plaintext,
    size_t           plaintextSize,
    char           **cipherText);

TSS2_RC Fapi_Encrypt_Async(
    FAPI_CONTEXT    *context,
    char      const *keyPath,
    char      const *policyPath,
    uint8_t   const *plaintext,
    size_t           plaintextSize);
```

```
TSS2_RC Fapi_Encrypt_Finish(
    FAPI_CONTEXT   *context,
    char          **cipherText);

TSS2_RC Fapi_Decrypt(
    FAPI_CONTEXT   *context,
    char     const *cipherText,
    uint8_t       **plainText,
    size_t         *plainTextSize);

TSS2_RC Fapi_Decrypt_Async(
    FAPI_CONTEXT   *context,
    char     const *cipherText);

TSS2_RC Fapi_Decrypt_Finish(
    FAPI_CONTEXT   *context,
    uint8_t       **plainText,
    size_t         *plainTextSize);

TSS2_RC Fapi_SetCertificate(
    FAPI_CONTEXT   *context,
    char     const *path,
    char     const *x509certData);

TSS2_RC Fapi_SetCertificate_Async(
    FAPI_CONTEXT   *context,
    char     const *path,
    char     const *x509certData);

TSS2_RC Fapi_SetCertificate_Finish(
    FAPI_CONTEXT   *context);

TSS2_RC Fapi_GetCertificate(
    FAPI_CONTEXT   *context,
    char     const *path,
    char          **x509certData);

TSS2_RC Fapi_GetCertificate_Async(
    FAPI_CONTEXT   *context,
    char     const *path);

TSS2_RC Fapi_GetCertificate_Finish(
    FAPI_CONTEXT   *context,
    char          **x509certData);

TSS2_RC Fapi_ExportKey(
    FAPI_CONTEXT   *context,
    char     const *pathOfKeyToDuplicate,
    char     const *pathToPublicKeyOfNewParent,
    char          **exportedData);

TSS2_RC Fapi_ExportKey_Async(
    FAPI_CONTEXT   *context,
    char     const *pathOfKeyToDuplicate,
    char     const *pathToPublicKeyOfNewParent);

TSS2_RC Fapi_ExportKey_Finish(
    FAPI_CONTEXT   *context,
    char          **exportedData);

/* Seal functions */

TSS2_RC Fapi_CreateSeal(
    FAPI_CONTEXT   *context,
    char     const *path,
    char     const *type,
    size_t          size,
    char     const *policyPath,
    char     const *authValue,
    uint8_t  const *data);

TSS2_RC Fapi_CreateSeal_Async(
    FAPI_CONTEXT   *context,
    char     const *path,
    char     const *type,
    size_t          size,
    char     const *policyPath,
    char     const *authValue,
    uint8_t  const *data);

TSS2_RC Fapi_CreateSeal_Finish(
    FAPI_CONTEXT   *context);

TSS2_RC Fapi_Unseal(
    FAPI_CONTEXT   *context,
    char     const *path,
    uint8_t       **data,
    size_t         *size);

TSS2_RC Fapi_Unseal_Async(
    FAPI_CONTEXT   *context,
    char     const *path);

TSS2_RC Fapi_Unseal_Finish(
    FAPI_CONTEXT   *context,
    uint8_t       **data,
    size_t         *size);

/* Policy functions */
```

```
TSS2_RC Fapi_PolicyExport(
    FAPI_CONTEXT   *context,
    char     const *path,
    char          **jsonPolicy);

TSS2_RC Fapi_PolicyExport_Async(
    FAPI_CONTEXT   *context,
    char     const *path);

TSS2_RC Fapi_PolicyExport_Finish(
    FAPI_CONTEXT   *context,
    char          **jsonPolicy);

TSS2_RC Fapi_Policy_AuthorizePolicy(
    FAPI_CONTEXT   *context,
    char     const *policyPath,
    char     const *keyPath,
    uint8_t  const *policyRef,
    size_t          policyRefSize);

TSS2_RC Fapi_Policy_AuthorizePolicy_Async(
    FAPI_CONTEXT   *context,
    char     const *policyPath,
    char     const *keyPath,
    uint8_t  const *policyRef,
    size_t          policyRefSize);

TSS2_RC Fapi_Policy_AuthorizePolicy_Finish(
    FAPI_CONTEXT   *context);

TSS2_RC Fapi_WriteAuthorizeNv(
    FAPI_CONTEXT   *context,
    char     const *nvPath,
    char     const *policyPath);

TSS2_RC Fapi_WriteAuthorizeNv_Async(
    FAPI_CONTEXT   *context,
    char     const *nvPath,
    char     const *policyPath);

TSS2_RC Fapi_WriteAuthorizeNv_Finish(
    FAPI_CONTEXT   *context);

/* Attestation functions */

TSS2_RC Fapi_PcrRead(
    FAPI_CONTEXT   *context,
    uint32_t        pcrIndex,
    uint8_t       **pcrValue,
    size_t         *pcrValueSize,
    char          **pcrLog);

TSS2_RC Fapi_PcrRead_Async(
    FAPI_CONTEXT   *context,
    uint32_t        pcrIndex);

TSS2_RC Fapi_PcrRead_Finish(
    FAPI_CONTEXT   *context,
    uint8_t       **pcrValue,
    size_t         *pcrValueSize,
    char          **pcrLog);

TSS2_RC Fapi_PcrExtend(
    FAPI_CONTEXT   *context,
    uint32_t        pcrIndex,
    uint8_t  const *data,
    size_t          dataSize,
    char     const *logData);

TSS2_RC Fapi_PcrExtend_Async(
    FAPI_CONTEXT   *context,
    uint32_t        pcrIndex,
    uint8_t  const *data,
    size_t          dataSize,
    char     const *logData);

TSS2_RC Fapi_PcrExtend_Finish(
    FAPI_CONTEXT   *context);

TSS2_RC Fapi_Quote(
    FAPI_CONTEXT   *context,
    uint32_t       *pcrList,
    size_t          pcrListSize,
    char     const *keyPath,
    char     const *quoteType,
    uint8_t  const *qualifyingData,
    size_t          qualifyingDataSize,
    char          **quoteInfo,
    uint8_t       **signature,
    size_t         *signatureSize,
    char          **pcrLog,
    char           *certificate);

TSS2_RC Fapi_Quote_Async(
    FAPI_CONTEXT   *context,
    uint32_t       *pcrList,
    size_t          pcrListSize,
    char     const *keyPath,
    char     const *quoteType,
    uint8_t  const *qualifyingData,
    size_t          qualifyingDataSize);

TSS2_RC Fapi_Quote_Finish(
    FAPI_CONTEXT   *context,
    char          **quoteInfo,
    uint8_t       **signature,
    size_t         *signatureSize,
    char          **pcrEventLog,
    char           *certificate);

TSS2_RC Fapi_VerifyQuote(
    FAPI_CONTEXT   *context,
    char     const *publicKeyPath,
    uint8_t  const *qualifyingData,
    size_t          qualifyingDataSize,
    char     const *quoteInfo,
    uint8_t  const *signature,
    size_t          signatureSize,
    char     const *pcrLog);

TSS2_RC Fapi_VerifyQuote_Async(
    FAPI_CONTEXT   *context,
    char     const *publicKeyPath,
    uint8_t  const *qualifyingData,
    size_t          qualifyingDataSize,
    char     const *quoteInfo,
    uint8_t  const *signature,
    size_t          signatureSize,
    char     const *pcrLog);

TSS2_RC Fapi_VerifyQuote_Finish(
    FAPI_CONTEXT   *context);

/* NV functions */

TSS2_RC Fapi_CreateNv(
    FAPI_CONTEXT   *context,
    char     const *path,
    char     const *type,
    size_t          size,
    char     const *policyPath,
    char     const *authValue);

TSS2_RC Fapi_CreateNv_Async(
    FAPI_CONTEXT   *context,
    char     const *path,
    size_t          size,
    char     const *policyPath,
    char     const *authValue);
```

```
TSS2_RC Fapi_CreateNv_Finish(
    FAPI_CONTEXT   *context);

TSS2_RC Fapi_NvRead(
    FAPI_CONTEXT   *context,
    char     const *path,
    uint8_t       **data,
    size_t         *size,
    char          **logData);

TSS2_RC Fapi_NvRead_Async(
    FAPI_CONTEXT   *context,
    char     const *path);

TSS2_RC Fapi_NvRead_Finish(
    FAPI_CONTEXT   *context,
    uint8_t       **data,
    size_t         *size,
    char          **logData);

TSS2_RC Fapi_NvWrite(
    FAPI_CONTEXT   *context,
    char     const *path,
    uint8_t  const *data,
    size_t          size);

TSS2_RC Fapi_NvWrite_Async(
    FAPI_CONTEXT   *context,
    char     const *path,
    uint8_t  const *data,
    size_t          size);

TSS2_RC Fapi_NvWrite_Finish(
    FAPI_CONTEXT   *context);

TSS2_RC Fapi_NvExtend(
    FAPI_CONTEXT   *context,
    char     const *path,
    uint8_t  const *data,
    size_t          size,
    char     const *logData);

TSS2_RC Fapi_NvExtend_Async(
    FAPI_CONTEXT   *context,
    char     const *path,
    uint8_t  const *data,
    size_t          size,
    char     const *logData);
```

```
TSS2_RC Fapi_NvExtend_Finish(
    FAPI_CONTEXT   *context);

TSS2_RC Fapi_NvIncrement(
    FAPI_CONTEXT   *context,
    char     const *path);

TSS2_RC Fapi_NvIncrement_Async(
    FAPI_CONTEXT   *context,
    char     const *path);

TSS2_RC Fapi_NvIncrement_Finish(
    FAPI_CONTEXT   *context);

TSS2_RC Fapi_NvSetBits(
    FAPI_CONTEXT   *context,
    char     const *path,
    uint64_t        bitmap);

TSS2_RC Fapi_NvSetBits_Async(
    FAPI_CONTEXT   *context,
    char     const *path,
    uint64_t        bitmap);

TSS2_RC Fapi_NvSetBits_Finish(
    FAPI_CONTEXT   *context);

typedef TSS2_RC (*Fapi_CB_Auth)(
    FAPI_CONTEXT   *context,
    char     const *description,
    char          **auth,
    void           *userData);

TSS2_RC Fapi_SetAuthCB(
    FAPI_CONTEXT   *context,
    Fapi_CB_Auth    callback,
    void           *userData);

typedef TSS2_RC (*Fapi_CB_Branch)(
    FAPI_CONTEXT   *context,
    char     const *description,
    char    const **branchNames,
    size_t          numBranches,
    size_t         *selectedBranch,
    void           *userData);

TSS2_RC Fapi_SetBranchCB(
    FAPI_CONTEXT   *context,
    Fapi_CB_Branch  callback,
    void           *userData);
```

```
typedef TSS2_RC (*Fapi_CB_Sign)(
    FAPI_CONTEXT   *context,
    char      const *description,
    char      const *publicKey,
    char      const *publicKeyHint,
    uint32_t         hashAlg,
    uint8_t   const *dataToSign,
    size_t           dataToSignSize,
    uint8_t        **signature,
    size_t          *signatureSize,
    void            *userData);

TSS2_RC Fapi_SetSignCB(
    FAPI_CONTEXT   *context,
    Fapi_CB_Sign    callback,
    void           *userData);

#ifdef __cplusplus
}
#endif

#endif /* TSS2_FAPI_H */
```