GlobalPlatform Technology

TEE Management Framework: Open Trust Protocol (OTrP) Profile

Version 1.0.0.6

Public Review

March 2020

Document Reference: GPD_SPE_123

Contents

Tables

Figures

1 Introduction

The GlobalPlatform TEE Management Framework (TMF) defines standard methods to administer a Trusted Execution Environment (TEE) from outside of the TEE. It is introduced in the GlobalPlatform specification *TEE Management Framework (including ASN.1 Profile)* ([TMF ASN.1]), which describes the security model for the administration of TEEs and of Trusted Applications (TAs) and the corresponding Security Domains (SDs). In particular, [TMF ASN.1] presents the roles and responsibilities of the different stakeholders involved in the administration of TEEs and TAs, the life cycle of administrated entities, and the mechanisms involved in administration operations. In addition, [TMF ASN.1] defines an ASN.1 profile for TMF.

This document specifies an Open Trust Protocol (OTrP) Profile that can be used in the context of TMF for the administration of TEEs, and of TAs and their corresponding SDs. This document also specifies the JSON encoding for OTrP messages.

The companion document *TEE Management Framework: Open Trust Protocol (OTrP) Mapping* ([OTrP Mapping]) shows how OTrP JSON messages map to the ASN.1 format TMF commands and how OTrP Security Domains map to TMF Security Domains. This is an informative mapping that enables a TEE that already exposes an ASN.1 TMF interface to support an OTrP Profile. It is not mandatory that an ASN.1 TMF interface exists; the JSON commands can be used directly for TEE management.

If you are implementing this specification and you think it is not clear on something:

1. **Check with a colleague.**

And if that fails:

2. **Contact GlobalPlatform at TEE-issues-GPD_SPE_123_v1.1@globalplatform.org**

1.1 Audience

This document is suitable for software developers implementing a mechanism for the TEE Management Framework for the Trusted Execution Environment (TEE).

This document is also intended for implementers of the TEE itself, its Trusted OS, Trusted Core Framework, the TEE APIs, and the communications infrastructure required to access Trusted Applications.

1.2 IPR Disclaimer

Attention is drawn to the possibility that some of the elements of this GlobalPlatform specification or other work product may be the subject of intellectual property rights (IPR) held by GlobalPlatform members or others. For additional information regarding any such IPR that have been brought to the attention of GlobalPlatform, please visit https://globalplatform.org/specifications/ip-disclaimers/. GlobalPlatform SHALL NOT be held responsible for identifying any or all such IPR, and takes no position concerning the possible existence or the evidence, validity, or scope of any such IPR.

1.3 References

The tables below list references applicable to this specification. The latest version of each reference applies unless a publication date or version is explicitly stated.

Table 1-1: Normative References

Standard / Specification	Description	Ref
GPD_SPE_010	GlobalPlatform Technology TEE Internal Core API Specification	[TEE Core]
GPD_SPE_120	GlobalPlatform Technology TEE Management Framework (including ASN.1 Profile) [Initially published as TEE Management Framework]	[TMF ASN.1]
GPD_SPE_124	GlobalPlatform Technology TEE Management Framework: Open Trust Protocol (OTrP) Mapping [to be published]	[OTrP Mapping]
GPD_SPE_009	GlobalPlatform Technology TEE System Architecture	[TEE Arch]
GPD_SPE_007	GlobalPlatform Technology TEE Client API Specification	[TEE Client]
RFC 2119	Key words for use in RFCs to Indicate Requirement Levels	[RFC 2119]
RFC 3447	Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications https://www.ietf.org/rfc/rfc3447	[RFC 3447]
RFC 4122	Version 1 UUID https://tools.ietf.org/html/rfc4122	[RFC 4122]
RFC 5280	Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile https://tools.ietf.org/html/rfc5280	[RFC 5280]
RFC 7515	JSON Web Signature (JWS) https://tools.ietf.org/html/rfc7515	[RFC 7515]
RFC 7516	JSON Web Encryption (JWE) https://tools.ietf.org/html/rfc7516	[RFC 7516]
RFC 7517	JSON Web Key (JWK) https://tools.ietf.org/html/rfc7517	[RFC 7517]
RFC 7518	JSON Web Algorithms (JWA) https://tools.ietf.org/html/rfc7518	[RFC 7518]

Table 1-2: Informative References

Standard / Specification	Description	Ref
OTrP from OTPA	The Open Trust Protocol (OTrP) v1.0, developed by the Open Trust Protocol Alliance	[OTPA OTrP]

1.4 Terminology and Definitions

The following meanings apply to SHALL, SHALL NOT, MUST, MUST NOT, SHOULD, SHOULD NOT, and MAY in this document (refer to [RFC 2119]):

- **SHALL** indicates an absolute requirement, as does MUST.
- **SHALL NOT** indicates an absolute prohibition, as does MUST NOT.
- **SHOULD** and **SHOULD NOT** indicate recommendations.
- **MAY** indicates an option.

Selected technical terms used in this document are defined in [TMF ASN.1] and [TEE Core].

Additional terminology is defined in Table 1-3 and in Table 2-1: Document-specific Terminology and Definitions.

Table 1-3: Terminology and Definitions

Term	Definition
Actor	A stakeholder performing a specific role in a GlobalPlatform-compliant environment. These stakeholders may take the form of card issuers, application developers, personalization bureaus, etc.
Authority	An Actor that grants permission to perform a specific set of actions. An Authority is represented in the device by a Security Domain.
Client Application	An application running outside of the Trusted Execution Environment (TEE) making use of the TEE Client API ([TEE Client]) to access facilities provided by Trusted Applications inside the TEE. Contrast *Trusted Application (TA)*.
Device State Information (DSI)	Contains the current configuration information for all Security Domains managed by a particular OWE. (For more information, see section 2.10.5.)
Execution Environment	An environment that hosts and executes software. This could be an REE, with hardware hosting Android, Linux, Windows, an RTOS, or other software; it could be a Secure Element or a TEE.
Nonce	A unique value that SHALL NOT be statistically likely to repeat. (For more information, see section 2.10.4.)
Outside World Entity (OWE)	An entity authorized to manage SDs on devices. (For more information, see section 2.5.) Replaces the Trusted Service Manager (TSM) discussed in The Open Trust Protocol (OTrP) v1.0 ([OTPA OTrP]).

Term	Definition
Regular Execution Environment (REE)	An Execution Environment comprising at least one Regular OS and all other components of the device (SoCs, other discrete components, firmware, and software) which execute, host, and support the Regular OS (excluding any Secure Components included in the device). From the viewpoint of a Secure Component, everything in the REE is considered untrusted, though from the Regular OS point of view there may be internal trust structures. (Formerly referred to as a *Rich Execution Environment (REE)*.) Contrast *Trusted Execution Environment (TEE)*.
Regular OS	An OS executing in a Regular Execution Environment. May be anything from a large OS such as Linux down to a minimal set of statically linked libraries providing services such as a TCP/IP stack. (Formerly referred to as a *Rich OS* or *Device OS*.) Contrast *Trusted OS*.
Root Security Domain (rSD)	A Security Domain over which other Authorities have very limited control; described in detail in [TMF ASN.1] section 4.1.3.3.
Secure Component	GlobalPlatform terminology to represent either a Secure Element or a Trusted Execution Environment.
Security Domain (SD)	An on-device representative of an Authority in the TEE Management Framework security model. Security Domains are responsible for the control of administration operations. SDs are used to perform the provisioning of TEE properties and to manage the life cycle of Trusted Applications and SDs associated with them.
Service Provider (SP)	An entity that issues TAs. (For more information, see section 2.6.)
Session	Logically connects multiple commands invoked on a Trusted Application or a Security Domain. In the context of this specification, logically connects an OWE to a TEE on a device. Begins with a `GetDeviceTEEStateRequest`.
Trusted Application (TA)	An application running inside the Trusted Execution Environment (TEE) that provides security related functionality to Client Applications outside of the TEE or to other Trusted Applications inside the TEE. Contrast *Client Application*.
Trusted Execution Environment (TEE)	An Execution Environment that runs alongside but isolated from an REE. A TEE has security capabilities and meets certain security-related requirements: It protects TEE assets against a set of defined threats which include general software attacks as well as some hardware attacks, and defines rigid safeguards as to data and functions that a program can access. There are multiple technologies that can be used to implement a TEE, and the level of security achieved varies accordingly. Contrast *Regular Execution Environment (REE)*.
Trusted OS	An OS executing in a Secure Component. Contrast *Regular OS*.

Term	Definition
Trusted Storage	Storage that is protected either by the hardware of the TEE or cryptographically by keys held in the TEE, and that is accessible only to the Trusted Application that created the data.

1.5 Abbreviations and Notations

Selected abbreviations and notations used in this document are defined in [TMF ASN.1] and [TEE Core].

Additional abbreviations and notations are included in Table 1-4 and in Table 2-1: Document-specific Terminology and Definitions.

Table 1-4: Abbreviations and Notations

Abbreviation / Notation	Meaning
AAD	Additional Authenticated Data
CBC	Cipher Block Chaining
CEK	Content Encryption Keys
DSI	Device State Information
ECC	Elliptic Curve Cryptography
JWA	JSON Web Algorithms
JWE	JSON Web Encryption
JWK	JSON Web Key
JWS	JSON Web Signature
OCSP	Online Certificate Status Protocol
OTrP	Open Trust Protocol
OWE	Outside World Entity
PKI	Public Key Infrastructure
REE	Regular Execution Environment
rSD	Root Security Domain
SD	Security Domain
SP	Service Provider
TA	Trusted Application
TEE	Trusted Execution Environment
TFW	Trusted Firmware
TMF	TEE Management Framework
TSM	Trusted Service Manager

1.6 Revision History

GlobalPlatform technical documents numbered *n*.0 are major releases. Those numbered *n*.1, *n*.2, etc., are minor releases where changes typically introduce supplementary items that do not impact backward compatibility or interoperability of the specifications. Those numbered *n*.*n*.1, *n*.*n*.2, etc., are maintenance releases that incorporate errata and precisions; all non-trivial changes are indicated, often with revision marks.

Table 1-5: Revision History

Date	Version	Description
May 2019	1.0	Public Release
October 2019	1.0.0.1	Committee Review Added structures to permit SD to support multiple keys with different roles. Use of multiple keys was originally discussed in section 4.8 but no mechanism to support such keys was provided. Clarified that TFW-Cert and TEE-Cert must reflect keys that are unique per instance. Modified `UpdateSDTBSRequest`, deprecating the option to change the Service Provider ID. (SD name is derived from Service Provider ID, so changing this ID would change SD name, and [TMF ASN.1] does not support such a change.) Added section 4.30 regarding version negotiation between host and client. Added section 2.10.7, specifying that implementations SHALL be able to update keys and certificates.
January 2020	1.0.0.5	Member Review
March 2020	1.0.0.6	Public Review
TBD	1.1	Public Release

2 OTrP Overview

2.1 Architecture

Figure 2-1: OTrP Architecture

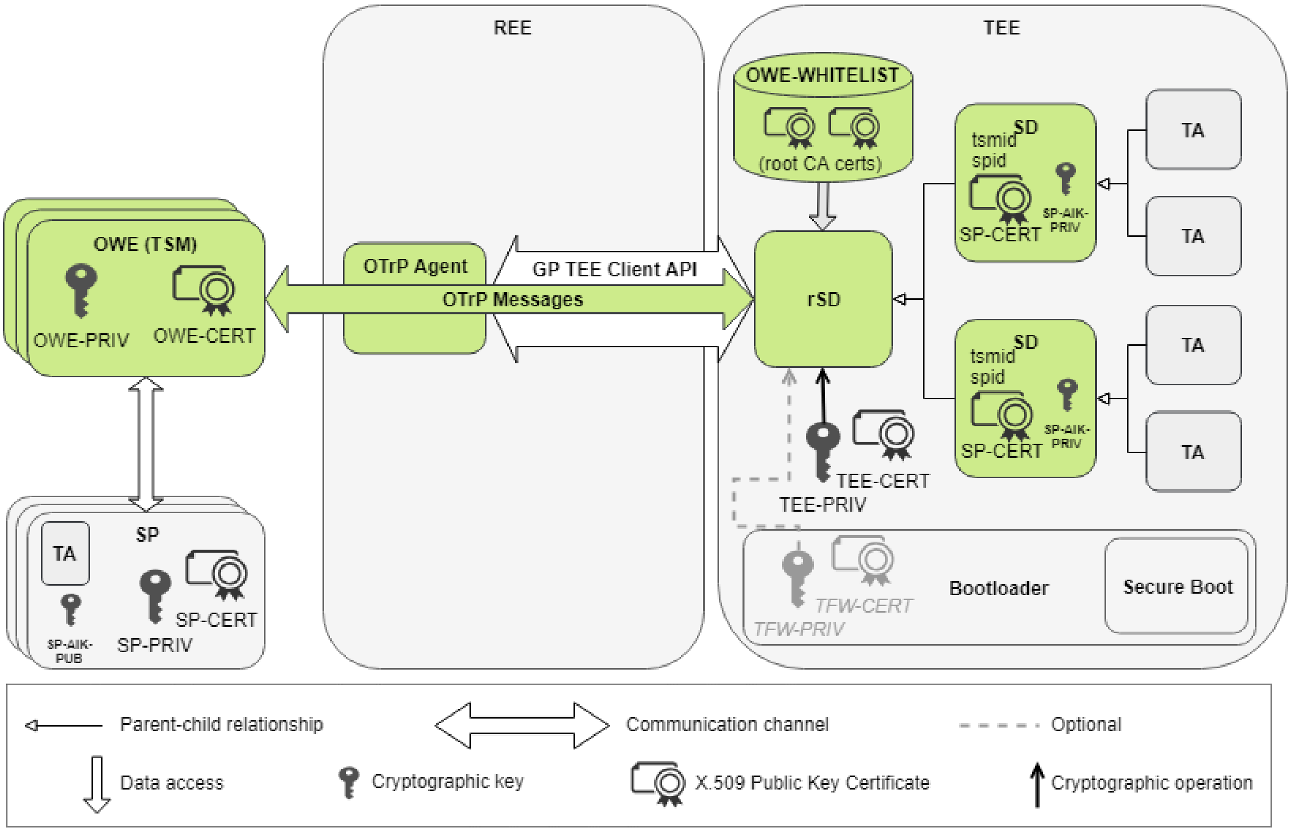

Figure 2-1 shows an architectural overview of OTrP Profile where one or more Outside World Entities (OWEs) interact with an end user's device using OTrP messages. An OWE is similar to a Trusted Service Manager (TSM), which is responsible for the life cycle management of trusted applications (TAs) running on TEEs of devices. Service Providers (SPs) rely on OWEs for distribution and life cycle management of their TAs in their users' devices.

An OTrP Profile compliant TEE SHALL have at least one root Security Domain (rSD), to which OTrP messages are sent through the device REE. If more than one rSD exists, the device SHALL have one rSD as the default rSD to which OTrP messages SHALL be sent unless a target rSD is indicated in the messages. OTrP messages follow a request-response pattern, where an OWE requests an operation and the TEE SHALL respond to the request. An OWE SHALL always initialize an OTrP session with a TEE by requesting the Device State Information (DSI) of the TEE. A TA is always installed in the context of a Security Domain (SD). An SD and a TA essentially have a parent-child relationship; i.e. the TA is a child node of the SD. Furthermore, in OTrP Profile, SDs SHALL be directly associated with the rSD; i.e. the rSD is the immediate parent of its child SDs. An OTrP Agent running on the REE SHALL be responsible for channeling to the OTrP messages between OWEs and relevant rSDs, using GlobalPlatform TEE Client API ([TEE Client]) interfaces.

2.2 Nomenclature

Table 2-1: Document-specific Terminology and Definitions

Term	Definition
OWE-Cert	The public key certificate containing OWE-Pub issued to the OWE by a Certificate Authority. The OWE-Cert SHALL contain an OWE identifier (tsmid).
OWE-Priv	The private portion of a key pair issued to a TEE management entity located outside of the TEE.
OWE-Pub	The public portion of a key pair issued to a TEE management entity located outside of the TEE.
OWE-Whitelist	A set of root Certificate Authority certificates. Each OWE-Cert SHALL chain to a root certificate in the OWE-Whitelist in order to be able to authenticate to an rSD.
rSD_{TA}	An rSD with TA and SD management privileges.
rSD_{TEE}	An rSD with the TEE management privilege.
sdid	A unique value that identifies an SD.
SP-AIK-Priv	The private portion of an anonymous identity key generated by the TEE whenever a first SD for an SP is created. TEE uses the SP-AIK-Priv to decrypt TA binaries and TA personalized data sent by the SP through OWE.
SP-AIK-Pub	The public portion of an anonymous identity key generated by the TEE whenever a first SD for an SP is created. The key pair is used to anonymously identify the device instead of TEE-Pub. The SP uses the SP-AIK-Pub to encrypt TA binaries and TA personalized data during TA life cycle management on a particular TEE.
SP-Cert	The public key certificate containing the SP-Pub issued to the Service Provider by a Certificate Authority or a self-signed certificate.
spid	A unique value that identifies an SP. OWEs SHALL maintain spids for SPs.
SP-Priv	The private portion of a key pair issued to a Service Provider that is used to sign trusted application code.
SP-Pub	The public portion of a key pair issued to a Service Provider that is used to sign trusted application code.
TEE-Cert	The public key certificate containing the TEE-Pub that is signed by the Certificate Authority of the TEE vendor.
TEE-Priv	The private portion of a key pair burned into the device, which is accessible only to the TEE software. TEE uses this key to sign data to attest its validity to a remote entity.
TEE-Pub	The public portion of a key pair burned into the device and accessible by the TEE software.
TFW-Cert	The public key certificate containing the TFW-Pub that is signed by the certificate authority of the TFW issuer.
TFW-Priv	The private portion of a key pair burned into the device, which is accessible only to the device firmware. The device firmware uses this key to sign data to attest its validity to a remote entity.

Term	Definition
TFW-Pub	The public portion of a key pair burned into the device and accessible by the device firmware.
tsmid	A unique value that identifies an OWE. Its value SHALL be the OWE identifier present in the OWE-Cert.

2.3 Root Security Domain (rSD)

A root Security Domain (rSD) is defined in GlobalPlatform TEE Management Framework ([TMF ASN.1]) section 4.1.3.3. In this document, an rSD refers to a root Security Domain in the context of the OTrP Profile. An rSD SHALL NOT have a parent SD that can authorize any OTrP operations on the rSD or its children. An OTrP rSD can be configured with privilege functions as listed in [TMF ASN.1] section 4.1.3.1, except that the rSD SHALL NOT be allowed to create another rSD. OTrP Profile allows a device to be configured with more than one rSD. However, an OTrP Profile compliant device SHALL be configured with at least one rSD with TA and SD management privileges. An rSD with the TEE management privilege SHALL be restricted to authorizing only TEE management operations and SHALL NOT authorize any TA and SD management operations.

In this document, the term rSD_{TA} refers to an rSD with TA and SD management privileges and rSD_{TEE} refers to an rSD with the TEE management privilege.

The UUID of an rSD SHALL be known to OWEs that wish to communicate with the rSD using OTrP messages. Each rSD possesses an OWE-Whitelist, which allows the rSD to determine whether a given OWE is trusted by validating the certificate chain of the OWE-Cert. OTrP Profile SHALL NOT allow an rSD to be installed on a device in the field using OTrP messages.

2.4 Security Domain (SD)

[TMF ASN.1] section 4.1 defines the concept of a Security Domain (SD). In this document, an SD refers to an SD created using OTrP messages. An SD SHALL have only TA Management and TA Personalization privileges. SDs SHALL be uniquely identified using UUIDs. A UUID for an SD SHALL be derived from the tsmid (see section 2.5) and the spid (see section 2.6) associated with the SD as follows:

- Convert *tsmid* to a bitstream.
- Convert *spid* to a bitstream.
- Concatenate the bitstreams as *tsmid* || *spid*
- Calculate the SHA-1 hash of {*tsmid* || *spid*}
- Transform the resulting 20-byte hash into *sdid*, a 16-byte UUID version 1 or a 16-byte UUID version 4, as described in section 2.4.1.

2.4.1 UUID Calculation

Transform the 20-byte hash value into a 16-byte UUID as follows. UUIDs are defined here in big-endian byte order. See [RFC 4122] for field definitions and encodings.

- Set octets 0 through 3 of the *time_low* field to octets 0 through 3 of the hash.
- Set octets 0 and 1 of the *time_mid* field to octets 4 and 5 of the hash.
- Set octets 0 and 1 of the *time_hi_and_version* field to octets 6 and 7 of the hash.
- Set the *clock_seq_hi_and_reserved* field to octet 8 of the hash.
- Set the two most significant bits (bits 6 and 7) of the *clock_seq_hi_and_reserved* field to 01.
- Set the *clock_seq_low* field to octet 9 of the hash.
- Set octets 0 through 5 of the *node* field to octets 10 through 15 of the hash.

To complete a version 1 UUID:

- Set the four most significant bits (bits 12 through 15) of the *time_hi_and_version* field to 0001.

To complete a version 4 UUID:

- Set the four most significant bits (bits 12 through 15) of the *time_hi_and_version* field to 0100.

2.5 Outside World Entity (OWE)

An Outside World Entity (OWE) is usually an entity authorized to manage SDs on devices. Each OWE is identified by a unique identifier called *tsmid*. OWE holds a private key OWE-Priv associated with the OWE-Cert, which it uses to sign OTrP messages. OWE receives the OWE-Cert from an intermediate CA. The OWE-Cert SHALL be chained to a root certificate in the OWE-Whitelist.

2.6 Service Provider (SP)

A Service Provider (SP) is an entity that issues TAs. An SP signs its TAs using the private key associated with its SP-Cert, and establishes a trust relationship with an OWE to deliver TAs. However, the mechanism used to establish a trust relationship is out of scope of the OTrP Profile document.

An SP SHALL be identified by a unique identifier called a *spid*. OWEs are responsible for maintaining the uniqueness of the spids within the context of an OWE. The spids are not required to be globally unique.

2.7 Trusted Firmware (TFW)

A Trusted Firmware (TFW) is a part of the TEE that is a layer outside of the trusted OS. The TFW layer is specific to a TEE architecture and may be unavailable in some TEEs. If available, the TFW is requested to sign a challenge during the beginning of an OTrP session; i.e. while processing the `GetDeviceTEEStateRequest`. The signed output and the TFW information are structured as `TRUSTED-FIRMWARE-TYPE`.

2.8 OTrP Agent

An OTrP Agent is an entity that runs on the REE of the device to facilitate communication between an OWE and TEE. It also provides interfaces for applications to query TAs and trigger OTrP sessions. The OTrP Agent SHALL use TEE Client API ([TEE Client]) to establish an administrative session to a relevant rSD in the TEE. The Agent SHALL channel OTrP messages to an rSD according to the encoding scheme defined in section 3 of this document.

2.9 OWE Certificate (OWE-Cert)

Each OWE that can manage TAs on devices SHALL have an OWE-Cert issued by an intermediate CA whose certificate chains to a root CA present in the OWE-Whitelist on the devices. The OWE-Whitelist is accessible to the rSD, which can validate the OWE-Cert chain during OTrP sessions to authorize OTrP operations requested by OWEs. An OWE SHALL sign every OTrP request message using the private key, which SHALL be verified using the OWE-Cert.

OWE-Cert SHALL identify the OWE. The OWE's identifier SHALL be encoded to the `dNSName` of the `SubjectAltName` extension of the OWE-Cert. The issuer of the OWE-Cert SHALL ensure that the OWE's identifier has not been issued to any other OWE. For example, if the identifier is a fully qualified domain name, then the domain must be owned by the OWE.

The OWE identifier SHALL be used as the tsmid in OTrP messages.

2.10 Security Model

The goals of the security model for OTrP Profile are:

- to provide means to manage the Trusted Execution Environment (TEE), Security Domains (SD), and Trusted Applications (TA)
- to ensure the security and the integrity of these entities
- to enable the confidentiality of the data
- to provide a scalable model allowing deployments involving a unique OWE or multiple OWEs
- to enforce the security policy of each OWE while preserving its assets

To ensure the security and integrity of these entities, the TMF OTrP Profile code implementation on the device is a Trusted OS Component (see TEE System Architecture, [TEE Arch]), or composed from a group of such components. As such it inherits the same security requirements as other Trusted OS Components.

2.10.1 Security Mechanism

OTrP Profile utilizes Public Key Infrastructure (PKI) combined with JSON Web Signature (JWS) ([RFC 7515]) and JSON Web Encryption (JWE) ([RFC 7516]) to allow the OWE to communicate securely with the rSD.

The OWE uses OTrP messages to create and manage Security Domains in the TEE, on behalf of Service Providers, and install, personalize, and manage trusted applications within these Security Domains.

PKI trust is used to enable the TEE to determine which OWEs to trust, and therefore multiple OWEs that meet the trust requirements (OWEs that can prove their identity using an unrevoked OWE-Cert that chains to the OWE-Whitelist) may communicate with the TEE via OTrP messages. Furthermore, the TEE validates the status of the OWE-Cert using the OCSP stapling provided along with the OTrP request messages.

OTrP Profile SHALL enforce the following access control policies on SDs and TAs:

- An OWE SHALL only be authorized to manage SDs that the OWE initially requested to create.
- An OWE SHALL only be authorized to manage TAs that are installed in the SDs that the OWE is authorized to manage.

OTrP Profile uses tsmid to enforce the access control policies on SDs and TAs. The rSD_{TA} associates each SD with the tsmid of the OWE that requested the creation of the SD. The rSD_{TA} validates the tsmid present on the OWE-Cert before authorizing operations on the SD.

2.10.2 Cryptographic Requirements

OTrP Profile SHALL use the JWS scheme for signing and the JWE scheme for encrypting messages. OTrP Profile SHALL use algorithms defined in JSON Web Algorithms (JWA) ([RFC 7518]) for signing, encryption, and key wrap operations. However, the OTrP Profile SHALL select only an algorithm that is supported by the TEE Internal Core API Specification ([TEE Core]).

2.10.3 Cryptographic Recommendations

OTrP Profile SHOULD use the following cryptographic recommendations:

- Symmetric cryptography: Minimum equivalent to AES with 128-bit keys.
- Hash functions: Minimum equivalent to SHA-256.
- Asymmetric cryptography: Minimum equivalent to RSA with key size of at least 3072 bits. However, it is recommended to use Elliptic Curve Cryptography (ECC) with P-256. Other curve values may be used. See Table B-1 for string identifiers of these curves.
- Key management: It is recommended to use Elliptic Curve Diffie–Hellman (ECDH) with key size of at least 256 bits for key agreement / management.
- RSA-based JWE and JWS SHOULD use separate key pairs for signing and encryption.

2.10.4 Nonce

Within all OTrP requests, the nonce plays a critical role in message synchronization. It is a unique value that allows the TEE to verify that it has not authorized any new operations on SDs and TAs belonging to the OWE since the last operation requested by the OWE. An OWE requests a nonce from the TEE at the beginning of an OTrP session; i.e. while requesting the DSI information. The TEE SHALL maintain a nonce per OWE and provide a nonce value to the OWE in every response message. The OWE SHALL use the same nonce value in the next OTrP request. The nonce value changes every time the TEE processes a request. A nonce value SHALL NOT be statistically likely to repeat within a single OTrP session. If the nonce value provided in a request does not match the one provided in the latest response, the TEE SHALL return an error status and the OWE SHALL reinitiate the OTrP session by requesting the DSI information. For more details, see sections 5.5.1 and 5.7.

2.10.5 Device State Information (DSI)

The DSI contains the current configuration information for all Security Domains managed by a particular OWE. The TEE maintains the DSI information for a particular OWE during an OTrP session. The TEE is also responsible for providing DSI information to the OWE at the beginning of the OTrP session. Once a DSI has been obtained by the OWE, further interaction with the TEE contains a hash of the DSI. The TEE provides DSI information in OTrP response messages if indicated by the OWE in the preceding request. The hash of the DSI SHALL be calculated using SHA-256 over the `DSI-CONTENT-TYPE`.

2.10.6 Use of Keys

Version 1.0 of this specification did not specify how to use keys, but simply provided for the Security Domain to return a single key.

Beginning with version 1.1, this specification permits the use of multiple keys.

When creating or updating a Security Domain, the OWE SHOULD provide two keys: one for signing and one for encryption. These SHALL be identified by the `TEE_OperationMode` assigned to each key within the `StoreData` structure. However, the OWE MAY provide a single key for both purposes, in which case it SHALL have both TEE_MODE_ENCRYPT and TEE_MODE_VERIFY. (`TEE_OperationMode` is defined in [TEE Core] section 6.1.)

In response, the Security Domain should create the same number, type, and size of key and return them in a `PUB-KEY-ROLE-ARRAY-TYPE`.

2.10.7 Key Update

Beginning with version 1.1, implementations SHALL be able to update of keys and certificates using `UpdateSDTBSRequest` commands described in section 5.10.

3 Encoding OTrP Messages Using TEE Client API

In a GlobalPlatform TEE that supports OTrP Profile, the TEE Client API ([TEE Client]) SHALL allow the OTrP messages to be sent to the TEE as follows:

- OTrP Agent opens an administrative session to the relevant OTrP root Security Domain.
- Using this session, the OTrP Agent forwards the OTrP requests using [TEE Client] `TEEC_InvokeCommand`.

3.1 Reserved Command IDs

When `TEEC_InvokeCommand` is called to send OTrP messages to a Security Domain, the [TEE Client] Command IDs defined in Table 3-1 are reserved.

Table 3-1: Reserved Command IDs

Range	Description
`0x00000000 – 0x00C1FFFF`	Reserved for GlobalPlatform use
`0x00C20000 – 0x00C2FFFF`	Reserved for TMF ASN.1 Profile
`0x00C30000`	JSON OTrP messages
`0x00C30001 – 0x00C3FFFF`	Reserved for TMF OTrP Profile
`0x00C40000 – 0x3FFFFFFE`	Reserved for GlobalPlatform use
`0x3FFFFFFF`	Defined Error value The Defined Error value is reserved for testing and validation and SHALL be treated as an undefined value when it is provided to an API.
`0x40000000 – 0xFFFFFFFF`	Implementation defined

3.2 Encoding OTrP Messages

The Command ID for forwarding OTrP messages via `TEEC_InvokeCommand` is 0x00C30000.

This command uses a single envelope command with the following parameters:

- The first parameter identifies the input buffer containing the OTrP request message as a UTF-8 encoded string. The byte representation of the OTrP request that is passed to the TEE SHALL NOT be null terminated. The TEE SHALL use the supplied length to determine the length of the OTrP request data and SHALL NOT rely on a null terminator being present.
- The second parameter identifies the output buffer containing the OTrP response message, which will be returned as a UTF-8 encoded string. The OTrP response returned from the TEE SHALL NOT include a null terminator.

Figure 3-1: Single Envelope Command

Table 3-2: Envelope Command Encoding

Parameters	Value	Description
Command ID	`0x00C30000`	OTrP message
Parameter #0	`TEEC_MEMREF_*_INPUT`	Request message including the command payload.
Parameter #1	`TEEC_MEMREF_*_OUTPUT`	Response message including the command response.
Parameter #2	`TEEC_NONE`	Not used
Parameter #3	`TEEC_NONE`	Not used
Status	–	Execution status of the envelope command.

3.2.1 Handling Variable Length Return Values

For handling variable length return values, see [TEE Core] section 3.4.4.

3.2.2 Atomicity of Operations

All operation commands SHALL appear atomic to entities using the GlobalPlatform OTrP Profile. Internally, a TEE may adopt a variety of strategies, including performing garbage collection and applying other required operations in a delayed manner following an OTrP operation command. Some OTrP operations MAY lock out GlobalPlatform TA or SD functionality until the TEE finishes processing the requested OTrP operation.

3.2.3 Returning OTrP Errors

Where possible – even in the event of an error – the status `TEEC_SUCCESS` should be returned, with the response data (Parameter #1) providing the JSON OTrP response message, which may itself indicate that there has been an OTrP error.

In some cases an error may be severe enough that an OTrP message cannot be returned. This might be due to insufficient response buffer allocation (which is described in section 3.2). In these cases, the error codes described in [TMF ASN.1] section 8.1.1, Using the Mandatory TEE Client API, should be used.

4 JSON Message Formatting

Each OTrP message (detailed in section 5) is carried within a JSON message structure and uses the Flattened JWS Serialization Syntax (see [RFC 7515] section 7.2.2).

OTrP messages shown in this document use the following typographic conventions for JSON data types:

- String: Strings in this document are represented as PRINTABLE-STRING-PRIMITIVE-TYPE, enclosed in quotes.
- Integer: Numbers are represented as INTEGER-PRIMITIVE-TYPE.
- Boolean: Booleans are simply represented as BOOLEAN. A Boolean value can either be true or false.
- Array: An array is a collection of values (either values of a single data type or objects). Arrays are enclosed in square brackets ([ ]) with values separated by commas (,).

JSON elements that are marked as OPTIONAL SHALL be ignored by the message receiver if not included in the messages.

4.1 COMMAND-TYPE

The `COMMAND-TYPE` is a JSON structure for signature output. OTrP Profile SHALL use the JWS scheme for signing data and SHALL follow the Flattened JWS JSON Serialization Syntax as:

```
{
        "payload":COMMAND-PAYLOAD,
        "protected":PROTECTED-HEADER-TYPE,
        "header":HEADER-TYPE,
        "signature":"PRINTABLE-STRING-PRIMITIVE-TYPE"
}
```

Where:

- `payload:` The `COMMAND-PAYLOAD` used as a payload to generate a signature.
- `protected:` The JWS protected header element structured as `PROTECTED-HEADER-TYPE`.
- `header:` The JWS header element structured as `HEADER-TYPE`. This element SHALL NOT be used for response messages.
- `signature:` The base64url encoded signature.

4.2 UNPRIVILEGED-COMMAND-TYPE

`UNPRIVILEGED-COMMAND-TYPE` SHALL be one of the following OTrP message types:

```
GET-TA-INFORMATION-REQUEST
GET-TA-INFORMATION-RESPONSE
```

4.3 COMMAND-PAYLOAD

`COMMAND-PAYLOAD` SHALL be the base64url encoding of:

```
COMMAND-TBS
```

Where:

- `COMMAND-TBS:` One of the following OTrP message types:

```
GET-DEVICE-TEE-STATE-TBS-REQUEST
GET-DEVICE-TEE-STATE-TBS-RESPONSE
CREATE-SD-TBS-REQUEST
CREATE-SD-TBS-RESPONSE
UPDATE-SD-TBS-REQUEST
UPDATE-SD-TBS-RESPONSE
DELETE-SD-TBS-REQUEST
DELETE-SD-TBS-RESPONSE
INSTALL-TA-TBS-REQUEST
INSTALL-TA-TBS-RESPONSE
UPDATE-TA-TBS-REQUEST
UPDATE-TA-TBS-RESPONSE
DELETE-TA-TBS-REQUEST
DELETE-TA-TBS-RESPONSE
STORE-TEE-PROPERTY-TBS-REQUEST
STORE-TEE-PROPERTY-TBS-RESPONSE
FACTORY-RESET-TBS-REQUEST
FACTORY-RESET-TBS-RESPONSE
```

4.4 PROTECTED-HEADER-TYPE

The `PROTECTED-HEADER-TYPE` is the JWS protected header. Its value is the base64url encoding of the following elements:

```
{
        "alg":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "rSD":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "tee":"PRINTABLE-STRING-PRIMITIVE-TYPE"
}
```

Where:

- `alg:` A cryptographic algorithm used to sign a message. Its value SHALL be one of the `"alg"` values defined in [RFC 7518]. However, if the selected algorithm in the OTrP request is not supported by [TEE Core] or is not acceptable by the OTrP Profile, then the rSD SHALL return the response with an error message. For more details on `alg`, see section 4.11, `SIGNATURE-PRIMITIVE-TYPE`.
- `rSD:` (OPTIONAL) The UUID of the rSD that is supposed to receive the request message. When this element is not supplied, the OTrP request SHALL be sent to the default rSD$_{TA}$ on the device.
- `tee:` (OPTIONAL) A zero-terminated string that describes the TEE to connect to. Its value matches the parameter `name` used to connect to a TEE while initializing a context using the `TEEC_InitializeContext`. See [TEE Client] section 4.5.2 for details. When this element is not supplied, the OTrP request SHALL be sent to the default TEE on the device.

4.5 HEADER-TYPE

The `HEADER-TYPE` is the JWS header with the following elements:

```
{
        "x5c":["CERT-PRIMITIVE-TYPE"],
        "kid":"PRINTABLE-STRING-PRIMITIVE-TYPE"
}
```

Where:

- `x5c:` An X.509 Certificate Chain (as described in [RFC 5280]) represented as a `CERT-PRIMITIVE-TYPE` array.
- `kid:` (OPTIONAL) A string indicating the key used in the JWS scheme for signing data.

`x5c` for the request message `GetDeviceTEEStateRequest` SHALL contain the entire OWE-Cert chain up to the root CA certificate as the `CERT-PRIMITIVE-TYPE` array. Other request messages may include OWE-Cert alone as the array element.

4.6 COMMAND-PARAMETER-TYPE

OTrP request messages SHALL have the following common elements:

```
{
        "ver":"GPD-VERSION-TYPE",
        "tid":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "rid":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "tee":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "nextdsi":BOOLEAN,
        "dsihash":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "nonce":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "content":CONTENT-ENCRYPTION-TYPE
}
```

Where:

- `ver:` The version of the OTrP request message structured as `GPD-VERSION-TYPE`.
- `tid:` A unique value to identify this transaction. The `tid` SHALL remain unchanged for an OTrP session that begins with `GetDeviceTEEStateRequest`.
- `rid:` A unique value to identify the request. The response SHALL contain the same `rid` value as the corresponding request.
- `tee:` A zero-terminated string that identifies the TEE as defined in [TEE Client] section 4.5.2.
- `nextdsi:` A Boolean value indicating whether a newly calculated `DSI-TYPE` SHALL be returned in the corresponding response message.
- `dsihash:` The base64 encoded SHA-256 hash of the `DSI-TYPE` obtained from the immediate previous response.
- `nonce:` For more information on `nonce`, see section 2.10.4. The `nonce` value SHALL match the value of the `nextnonce` the OWE received in the immediate previous response.
- `content:` Encrypted data structured as a `CONTENT-ENCRYPTION-TYPE`. The input to the encryption function is specific to the request message type as detailed within the request descriptions.

Note: The `COMMAND-PARAMETER-TYPE` may also include additional elements specific to an OTrP request message.

4.7 RESPONSE-PARAMETER-TYPE

In response to a request, the rSD returns a response with the following common elements:

```
{
        "ver":"GPD-VERSION-TYPE",
        "rid":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "tid":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "content":CONTENT-ENCRYPTION-TYPE
}
```

Where:

- `ver:` The version of the OTrP response message structured as `GPD-VERSION-TYPE`.
- `rid:` A unique value identifying the corresponding request.
- `tid:` A unique value identifying the OTrP session.
- `content:` Encrypted data structured as a `CONTENT-ENCRYPTION-TYPE`. The input to the encryption function is specific to the response message type as detailed within the response descriptions.

4.8 CONTENT-ENCRYPTION-TYPE

The `CONTENT-ENCRYPTION-TYPE` is a JSON structure for encrypted data in OTrP messages. `CONTENT-ENCRYPTION-TYPE` uses JWE for encrypting data and follows the Flattened JWE JSON Serialization Syntax. Symmetric keys known as Content Encryption Keys (CEK) are used to encrypt the data. When using RSA, the CEK and authentication HMAC key are encrypted or wrapped by a recipient's public asymmetric key (OWE-Pub or TEE-Pub).

For ECDH, the CEK is agreed using the recipient's public key (OWE-Pub or TEE-Pub) and an ephemeral key is generated by the sender. OTrP Profile does not use JWE AAD (Additional Authenticated Data) as every message is signed after encryption.

The JSON structure for the `CONTENT-ENCRYPTION-TYPE` is as follows:

```
{
        "protected":"ENCRYPTION-PRIMITIVE-TYPE",
        "recipients":[KEYWRAP-INFO-TYPE],
        "iv":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "ciphertext":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "tag":"PRINTABLE-STRING-PRIMITIVE-TYPE"
}
```

Where:

- `protected:` A mandatory JWE header parameter that indicates the cryptographic algorithm used for encryption.
- `recipients:` An array of `KEYWRAP-INFO-TYPE`, each containing information about CEK specific to a recipient.
- `iv:` The base64url encoded initialization vector as defined in [RFC 7516] section A.1.4.
- `ciphertext:` The base64url encoded encrypted data. The input to the encryption function is specific to `COMMAND-TBS`.
- `tag:` The base64url encoded authenticated tag calculated as defined in [RFC 7516] section 5.1.

4.9 KEYWRAP-INFO-TYPE

The `KEYWRAP-INFO-TYPE` is a JSON structure that contains a wrapped key and the information specific to a recipient on unwrapping.

A `KEYWRAP-INFO-TYPE` containing a key wrapped with the recipient's RSA public key is structured as:

```
{
        "header":{
                "alg":"KEYWRAP-PRIMITIVE-TYPE",
                "kid":"PRINTABLE-STRING-PRIMITIVE-TYPE"
        },
        "encrypted_key":"PRINTABLE-STRING-PRIMITIVE-TYPE"
}
```

Where:

- `header:` A mandatory header that contains `alg` element.
- `alg:` The `KEYWRAP-PRIMITIVE-TYPE` value that indicates the algorithm from JSON Web Algorithms ([RFC 7518]) used to encrypt CEK.
- `kid:` (OPTIONAL) A string indicating the key used to encrypt CEK. The value of `kid` SHALL be the base64 encoded value of a SHA-256 of the `PUB-KEY-TYPE`.
- `encrypted_key:` The base64url encoding value of the JWE encrypted CEK.

A `KEYWRAP-INFO-TYPE` containing a key wrapped using ECDH is structured as:

```
{
        "header":{
                "alg":"KEYWRAP-PRIMITIVE-TYPE",
                "kid":"PRINTABLE-STRING-PRIMITIVE-TYPE"
        },
        "encrypted_key":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "epk":PUB-KEY-TYPE,
        "apu":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "apv":"PRINTABLE-STRING-PRIMITIVE-TYPE"
}
```

Where `header`, `alg`, `kid`, and `encrypted_key` are as defined above, and:

- `epk:` An ephemeral ECC public key structured as the ECC-based `PUB-KEY-TYPE`.
- `apu:` The base64url encoded agreement `PartyUInfo` value for key agreement algorithm.
- `apv:` The base64url encoded agreement `PartyVInfo` value for key agreement algorithm.

4.10 ENCRYPTION-PRIMITIVE-TYPE

The `ENCRYPTION-PRIMITIVE-TYPE` indicates the cryptographic algorithm used for encryption in `CONTENT-ENCRYPTION-TYPE`. Its value SHALL be the base64url encoding of one of the `"enc"` values defined in [RFC 7518]. However, if the selected algorithm in the OTrP request is not supported by [TEE Core] or is not acceptable by the OTrP Profile, then the rSD SHALL return the response with an error message.

The following JSON structures are examples of `ENCRYPTION-PRIMITIVE-TYPE` defined using AES-CBC, and HMAC generated using SHA-256 and SHA-512.

```
{"enc":"A128CBC-HS256"}
{"enc":"A256CBC-HS512"}
```

Where:

- `{"enc":"A128CBC-HS256"}`: Represents content encryption with a 128-bit AES key in CBC mode, and an HMAC message authentication code with a 128-bit MAC key and the SHA-256 hash function.
- `{"enc":"A256CBC-HS512"}`: Represents content encryption with a 128-bit AES key in CBC mode, and an HMAC message authentication code with a 256-bit MAC key and the SHA-512 hash function.

Table 4-1 provides the base64url encoded values for these examples of `ENCRYPTION-PRIMITIVE-TYPE`.

Table 4-1: Examples of base64url encoded ENCRYPTION-PRIMITIVE-TYPE

ENCRYPTION-PRIMITIVE-TYPE	base64url encoded
{"enc":"A128CBC-HS256"}	eyJlbmMiOiJBMTI4Q0JDLUhTMjU2In0g
{"enc":"A256CBC-HS512"}	eyJlbmMiOiJBMjU2Q0JDLUhTNTEyIn0g

Table 4-2 lists the corresponding algorithms in [TEE Core] to support the above example encryption algorithms. (These correspondences were true when this specification was published. Confirm the latest information in [TEE Core] and [RFC 7518].)

Table 4-2: Example [TEE Core] Algorithms to Support ENCRYPTION-PRIMITIVE-TYPE

JSON Web Algorithms	[TEE Core] Algorithms
A128CBC-HS256	TEE_ALG_AES_CBC_NOPAD, TEE_ALG_HMAC_SHA256
A128CBC-HS512	TEE_ALG_AES_CBC_NOPAD, TEE_ALG_HMAC_SHA512

A128CBC-HS256 and A256CBC-HS512 use PKCS#7 padding. The padding mechanism should be implemented separately as [TEE Core] does not support it.

Note: See section 2.10 for additional information.

4.11 SIGNATURE-PRIMITIVE-TYPE

The SIGNATURE-PRIMITIVE-TYPE indicates the cryptographic algorithm used to sign a message. Its value SHALL be the base64url encoding of one of the "alg" values defined in [RFC 7518]. However, if the selected algorithm in the OTrP request is not supported by [TEE Core] or is not acceptable by the OTrP Profile, then the rSD SHALL return the response with an error message.

The following JSON structures are examples of SIGNATURE-PRIMITIVE-TYPE defined using RSA and ECC algorithms with key sizes of 256 bits.

```
{"alg":"RS256"}
{"alg":"ES256"}
```

Where:

- {"alg":"RS256"}: Represents signature generated with RSASSA-PKCS1-v1_5 ([RFC 3447]) using SHA-256.
- {"alg":"ES256"}: Represents signature generated with ECDSA using P-256 curve and SHA-256.

Table 4-3 provides the base64url encoded values for these examples of SIGNATURE-PRIMITIVE-TYPE.

Table 4-3: Examples of base64url encoded SIGNATURE-PRIMITIVE-TYPE

SIGNATURE-PRIMITIVE-TYPE	base64url encoded
{"alg":"RS256"}	eyJhbGciOiJSUzI1NiJ9
{"alg":"ES256"}	eyJhbGciOiJFUzI1NiJ9

Table 4-4 lists the corresponding algorithms in [TEE Core] to support the above example "alg". (These correspondences were true when this specification was published. Confirm the latest information in [TEE Core] and [RFC 7518].)

Table 4-4: Example [TEE Core] Algorithms to Support SIGNATURE-PRIMITIVE-TYPE

JSON Web Algorithms	[TEE Core] Algorithms
RS256	TEE_ALG_RSASSA_PKCS1_V1_5_SHA256
ES256	TEE_ALG_ECDSA_SHA256

Note: See section 2.10 for additional information.

4.12 KEYWRAP-PRIMITIVE-TYPE

The `KEYWRAP-PRIMITIVE-TYPE` describes the key management algorithm used to wrap CEK while encrypting data in the `CONTENT-ENCRYPTION-TYPE`. The `KEYWRAP-PRIMITIVE-TYPE` SHALL be one of the key management algorithms defined in [RFC 7518]. However, if the selected algorithm in the OTrP request is not supported by [TEE Core] or is not acceptable by the OTrP Profile, then the rSD SHALL return the response with an error message.

Examples of JSON Web Algorithms for key management that may be used to wrap CEK are as follows:

```
RSA1_5
ECDH-ES+A128KW
ECDH-ES+A256KW
```

Table 4-5 lists the corresponding algorithms in [TEE Core] to support the above example key management algorithms. (These correspondences were true when this specification was published. Confirm the latest information in [TEE Core] and [RFC 7518].)

Table 4-5: Example [TEE Core] Algorithms to Support KEYWRAP-PRIMITIVE-TYPE

Key Management Algorithms	[TEE Core] Algorithms
RSA1_5	TEE_ALG_RSASSA_PKCS1_V1_5_SHA256
ECDH-ES+A128KW	TEE_ALG_ECDH_DERIVE_SHARED_SECRET
ECDH-ES+A256KW	TEE_ALG_ECDH_DERIVE_SHARED_SECRET

Note: See section 2.10 for additional information.

4.13 CERT-PRIMITIVE-TYPE

The `CERT-PRIMITIVE-TYPE` is the base64 encoded representation of an X.509 certificate.

4.14 OPERATION-RESPONSE-PRIMITIVE-TYPE

One of the following strings:

```
OPERATION_SUCCESS
ERR_DEV_STATE_MISMATCH
ERR_INVALID_UUID
ERR_OCSP_INVALID
ERR_OWE_NOT_TRUSTED
ERR_REQUEST_INVALID
ERR_SD_NOT_EMPTY
ERR_SDID_ALREADY_USED
ERR_SPCERT_INVALID
ERR_TA_ALREADY_INSTALLED
ERR_TA_INVALID
ERR_TA_NOT_FOUND
ERR_TEE_BUSY
ERR_TEE_FAIL
ERR_TEE_RESOURCE_FULL
ERR_TEE_UNKNOWN
ERR_TFW_NOT_TRUSTED
ERR_UNSUPPORTED_CRYPTO_ALG
ERR_UNSUPPORTED_MSG_VERSION
ERR_UPDATING_DATA
```

Where the values have the following meanings:

- `OPERATION_SUCCESS:` Returned when the corresponding request message has been processed successfully.
- `ERR_DEV_STATE_MISMATCH:` Returned when the DSI hash value from OWE doesn't match that of the device's current DSI.
- `ERR_INVALID_UUID:` Returned when the given UUID is not supported or cannot be verified.
- `ERR_OCSP_INVALID:` Returned when the OCSP stapling is either invalid, not available, or expired.
- `ERR_OWE_NOT_TRUSTED:` Returned when the OWE-Cert chain cannot be validated using the root CA certificate in the OWE-Whitelist while processing a request message.
- `ERR_REQUEST_INVALID:` Returned when any of the following conditions occurs:
    - Request message is not supported by the rSD.
    - Request message has an invalid message structure; e.g. mandatory element is absent, or undefined elements or structures are included.

    - Failure to verify message signature.
    - Failure to decrypt `CONTENT-ENCRYPTION-TYPE` value.
    - Insufficient privilege to perform an operation (e.g. deletion of a TA from an SD that the OWE is not allowed to access).
- `ERR_SD_NOT_EMPTY:` Returned when an OWE tries to delete an SD that contains one or more TAs.
- `ERR_SDID_ALREADY_USED:` Returned when an OWE requests creation of an SD with a UUID that already exists in the namespace of the OWE in the TEE.
- `ERR_SPCERT_INVALID:` Returned when the new SP-Cert provided while updating an SD is not valid.
- `ERR_TA_ALREADY_INSTALLED:` Returned when an OWE requests installation of a TA with a given UUID and a version that already exists.
- `ERR_TA_INVALID:` Returned when any of the following conditions occurs while checking validity of a TA:
    - TA binary has a format that the TEE doesn't recognize.
    - TEE fails to decrypt the encoding of TA binary and personalization data.
    - If the SP isn't registered with the SD where a TA is to be installed.
    - During an update, if the version of the TA is lower than the current version installed.
    - If the TA version information provided in the request message is different than the TA version associated with the TA binary.
- `ERR_TA_NOT_FOUND:` Returned when the target TA doesn't exist in the SD.
- `ERR_TEE_BUSY:` Returned when the device TEE is currently busy.
- `ERR_TEE_FAIL:` Returned when any of the following conditions occurs:
    - TEE fails to respond to an OWE request. The OTrP Agent will construct an error message in responding to the OWE's request.
    - TEE fails to process a request because of its internal error.
- `ERR_TEE_RESOURCE_FULL:` Returned when a device resource is no longer available, such as storage space is full.
- `ERR_TEE_UNKNOWN:` Returned when the TEE is not supposed to receive the request, as determined by checking the TEE name or device identifier (`did`) in the request message.
- `ERR_TFW_NOT_TRUSTED:` Returned when the TEE determines that the underlying device firmware is not trustworthy.
- `ERR_UNSUPPORTED_CRYPTO_ALG:` Returned when a request message contains `CONTENT-ENCRYPTION-TYPE` value encrypted with a cryptographic algorithm that the TEE doesn't support.
- `ERR_UNSUPPORTED_MSG_VERSION:` Returned when the OTrP version of the request message is not supported by the TEE.
- `ERR_UPDATING_DATA:` Returned when updating a data parameter (`sd_data` or `encrypted_ta_data`) during an `UpdateSDRequest` or an `UpdateTARequest` is unsuccessful.

4.15 DSI-TYPE

The JSON structure that contains the DSI value. This structure SHALL be used to calculate the `dsihash` value.

```
{
        "dsi":DSI-CONTENT-TYPE
}
```

Where:

- `dsi:` The device state information value structured as `DSI-CONTENT-TYPE`.

4.16 DSI-CONTENT-TYPE

The JSON structure that describes the current DSI is as follows:

```
{
        "tfwdata":TRUSTED-FIRMWARE-TYPE,
        "tee":TEE-DESCRIPTION-TYPE
}
```

Where:

- `tfwdata:` (OPTIONAL) The trusted firmware information structured as `TRUSTED-FIRMWARE-TYPE`.
- `tee:` The underlying TEE information structured as `TEE-DESCRIPTION-TYPE`.

4.17 TRUSTED-FIRMWARE-TYPE

A `TRUSTED-FIRMWARE-TYPE` provides information regarding the trusted firmware on the device. It is structured according to the JWS scheme and the TFW key is used to generate the signature.

```
{
        "payload":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "protected":PROTECTED-HEADER-TYPE,
        "header":HEADER-TYPE,
        "signature":"PRINTABLE-STRING-PRIMITIVE-TYPE"
}
```

Where:

- `payload:` The string representing a challenge that the TFW SHALL sign. The `tid` value from the corresponding `GetDeviceTEEStateRequest` is used as the challenge.
- `protected:` The JWS protected header element structured as `PROTECTED-HEADER-TYPE`. The `PROTECTED-HEADER-TYPE` SHALL include only the `"alg"` element that indicates the cryptographic algorithm used to sign the payload.
- `header:` The JWS header element structured as `HEADER-TYPE`. The `x5c` element of the `HEADER-TYPE` SHALL contain the TFW-Cert represented as `CERT-PRIMITIVE-TYPE`. Storage of TFW-Cert is implementation defined.

  Note: Version 1.0 of this specification incorrectly suggested that `gpd.tee.firmware.implementation.binaryversion` could be used to extract TFW-Cert from the TFW.
- `signature:` The base64url encoded signature calculated according to the JWS scheme.

Note: The interface for the TEE to request the signature over a challenge from the trusted firmware is implementation specific.

4.18 TEE-DESCRIPTION-TYPE

A JSON structure that describes the TEE available on the device.

```
{
        "name":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "teever":"GPD-VERSION-TYPE",
        "cert":"CERT-PRIMITIVE-TYPE",
        "cacert":["CERT-PRIMITIVE-TYPE"],
        "sdlist":[SD-DEFINITION-TYPE],
        "teeaiklist":[TEE-AIK-TYPE],
        "isaset":ISA-TYPE,
        "teeImplementationProperty":[TEE-PROPERTY-TYPE]
}
```

Where:

- `name:` A zero-terminated string that describes the TEE to connect to. Its value matches the parameter `name` used to connect to a TEE while initializing a context using the `TEEC_InitializeContext`. For details, see [TEE Client] section 4.5.2.
- `teever:` The TEE version `gpd.tee.trustedos.implementation.version` structured as `GPD-VERSION-TYPE`.
- `cert:` The TEE-Cert represented as `CERT-PRIMITIVE-TYPE`. The certified key must be unique to the TEE instance.

  Note: Storage of the TEE-Cert and associated key is out of scope.

  Note: Version 1.0 of this specification incorrectly suggested that `gpd.tee.trustedos.implementation.binaryversion` could be used to extract TEE-Cert from the TFW.
- `cacert:` The X.509 certificate chain starting with the CA certificate that issued the TEE-Cert up to the root CA certificate structured as the `CERT-PRIMITIVE-TYPE` array.
- `sdlist:` An array of `SD-DEFINITION-TYPE`, where each element of the array provides the metadata of an SD that a given OWE has access to. This element SHALL be excluded if the `rSD` that prepares this JSON object is an rSD_{TEE}.
- `teeaiklist:` An array of `TEE-AIK-TYPE`, where each element of the array provides information related to an SP-AIK. This element SHALL be excluded if the `rSD` that prepares this JSON object is an rSD_{TEE}.
- `isaset:` (OPTIONAL) Instruction set and architecture definition based on ISA Type defined in [TMF ASN.1] section 9.1.4.
- `teeImplementationProperty:` (OPTIONAL) Lists the TEE properties. For more information on TEE properties, see [TMF ASN.1] section A.5. This element SHALL be included only if the `rSD` that prepares this JSON structure is an rSD_{TEE}. However, if the TEE has TMF ASN.1 audit SD capabilities, then OTrP SHALL provide the following valid API name string to be used with the `optionalApis` attribute of TEE Type, defined in [TMF ASN.1] section 9.1.6.

Table 4-6: Internal API Names Strings Definition

Strings	Description
TMF-OTrP-Profile	OTrP Profile of TEE Management Framework

4.19 SD-DEFINITION-TYPE

A JSON structure that describes the metadata information of an SD.

```
{
        "sdid":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "spid":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "protocol":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "talist":[TA-DEFINITION-TYPE]
}
```

Where:

- `sdid:` The base64 encoded UUID of the SD. See section 2.4 for the SD UUID generation.
- `spid:` The base64 encoded Service Provider identifier that is associated with the SD. See section 2.6 for the `spid` value generation.
- `protocol:` (OPTIONAL) The base64 encoded data that informs the OWE that the SD supports TMF commands in addition to OTrP Profile. The format of the protocol SHALL be a `SecureLayerAuditInfo` as defined in [TMF ASN.1] section 9.1.1.
- `talist:` An array of `TA-DEFINITION-TYPE`, where each element of the array provides information about a TA installed within the context of the SD.

4.20 TA-DEFINITION-TYPE

A JSON structure that provides the version number information and the UUID of a TA.

```
{
        "taid":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "taver":"PRINTABLE-STRING-PRIMITIVE-TYPE"
}
```

Where:

- `taid:` The base64 encoded UUID of a TA.
- `taver:` The string containing the TA version information. The TA version information SHALL use the `gpd.ta.version` property defined in [TEE Core].

4.21 ISA-TYPE

A JSON structure that describes the details of an instructional set and architecture that may be used by Trusted Applications. For details, see [TMF ASN.1] section 9.1.4, ISA Type.

```
{
        "isaName":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "processorType":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "instructionSet":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "addressSize":INTEGER-PRIMITIVE-TYPE,
        "abi":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "endianness":INTEGER-PRIMITIVE-TYPE
}
```

Where:

- `isaName:` Specifies a human readable description of the instruction set architecture.
- `processorType:` Indicates the type of the processor as a string.
- `instructionSet:` The instruction set as a string.
- `addressSize:` The size of addresses in bits as a number.
- `abi:` The Application Binary Interface in use.
- `endianness:` How values greater than 1 byte in length are stored.

4.22 TEE-PROPERTY-TYPE

A JSON structure that provides the TEE property information to be stored. TEE properties are detailed in [TMF ASN.1] section A.5.

```
{
        "PROPERTY-NAME":PROPERTY-VALUE
}
```

Where:

- `PROPERTY-NAME:` A string that identifies the TEE property as described in [TMF ASN.1] section A.5.
- `PROPERTY-VALUE:` The value of the TEE property.

4.23 TEE-AIK-TYPE

A JSON structure that describes the SP-AIK-Pub information associated with an SP.

Version 1.0 specified that there could be only one key per Service Provider. Version 1.1 permits multiple keys, with separate keys for encryption and signature.

Version 1.1

```
{
        "spaik":[PUB-KEY-ROLE-TYPE],
        "spid":"PRINTABLE-STRING-PRIMITIVE-TYPE"
}
```

Where:

- `spaik:` An array of SP-AIK-Pub keys structured as `PUB-KEY-ROLE-TYPE`.
- `spid:` The Service Provider identifier associated with the SP-AIK key.

Version 1.0

```
{
        "spaik":PUB-KEY-TYPE,
        "spid":"PRINTABLE-STRING-PRIMITIVE-TYPE"
}
```

Where:

- `spaik:` SP-AIK-Pub key structured as `PUB-KEY-TYPE`.
- `spid:` The Service Provider identifier associated with the SP-AIK key.

4.24 PUB-KEY-TYPE

The `PUB-KEY-TYPE` is a public key structured according to JSON Web Key (JWK) ([RFC 7517]).

4.24.1 RSA Key

An RSA-based `PUB-KEY-TYPE` is structured as:

```
{
        "kty":"RSA",
        "n":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "e":"PRINTABLE-STRING-PRIMITIVE-TYPE"
}
```

Where:

- `kty:` The JWK Key Type parameter indicating the cryptographic algorithm used with the key. The `kty` value for RSA public keys is fixed to the string `"RSA"`.
- `n:` The base64urlUInt encoded RSA public key modulus n.
- `e:` The base64urlUInt encoded RSA public key exponent e.

4.24.2 ECC Key

An ECC-based `PUB-KEY-TYPE` is structured as:

```
{
        "kty":"EC",
        "crv":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "x":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "y":"PRINTABLE-STRING-PRIMITIVE-TYPE"
}
```

Where:

- `kty:` The `kty` value for ECC keys is fixed to the string `"EC"`.
- `crv:` A string defining the curve type used with the ECC key.
- `x:` The base64url encoded x component of the ECC key.
- `y:` The base64url encoded y component of the ECC key.

Note: The curve values are listed in [RFC 7518]. However, other curve values may be used, as discussed in section 2.10. See Table B-1 for the string identifiers of these curves.

4.25 OCSP-ARRAY-TYPE

```
["PRINTABLE-STRING-PRIMITIVE-TYPE"]
```

A JSON array of OCSP stapling. Each element is a base64 encoded string. Multiple elements SHALL be represented using comma separation.

4.26 UUID-ARRAY-TYPE

```
["PRINTABLE-STRING-PRIMITIVE-TYPE"]
```

A JSON array containing a base64 encoded UUID string. Multiple elements SHALL be represented using comma separation.

4.27 GPD-VERSION-TYPE

The version number SHALL be represented as a string of the following form:

```
"GPD.TEE.[Major].[Minor].[Maintenance].[RFU]"
```

Where:

- `Major:` The major version number of the specification.
- `Minor:` The minor version number of the specification.
- `Maintenance:` The maintenance version number of the specification. If the version is not a maintenance release, this SHALL be zero.
- `RFU:` Reserved for future use. Currently this byte SHALL be zero.

There SHALL be no leading zeros and the string may contain only digits and `"."`.

A zero value SHALL be represented by a `"0"` and not an empty position.

For example, an OTrP message based on the initial version of this specification would indicate the version as the string `"GPD.TEE.1.0.0.0"`.

4.28 POP-TYPE

A JSON structure used as proof of possession to validate UUID version 5. `POP-TYPE` is structured as follows:

```
{
        "popkey":PUB-KEY-TYPE,
        "alg":SIGNATURE-PRIMITIVE-TYPE,
        "pop":"PRINTABLE-STRING-PRIMITIVE-TYPE"
}
```

Where:

- `popkey:` Specifies a public key structured as `PUB-KEY-TYPE` whose hash matches the specified UUID according to the rules set out in [TMF ASN.1] section 5.6.2.
- `alg:` The algorithm used to calculate proof of possession signature.
- `pop:` The base64 encoded proof of possession signature constructed as defined in [TMF ASN.1] section 5.6.2.

4.29 PUB-KEY-ROLE-TYPE

A JSON structure used to hold a public key and its roles.

```
{
        "role":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "key":PUB-KEY-TYPE
}
```

Where:

- role: specifies the key role and is either:
    - `"Enc"`: A public key that can be used to encrypt data sent to this Security Domain.
    - `"Ver"`: A public key that can be used to verify signatures created by this Security Domain.
    - `"All"`: A public key that can be used both for encrypting data and for verifying signatures.
- `key`: The public key encoded as a `PUB-KEY-TYPE`.

Note: "All" should be used only for backwards compatibility with Security Domains provisioned with version 1.0 of this specification, which only supported a single key.

4.30 PUB-KEY-ROLE-ARRAY-TYPE

```
[PUB-KEY-ROLE-ARRAY-TYPE]
```

An JSON array containing `PUB-KEY-ROLE-TYPE`. Multiple elements SHALL be represented using comma separation.

4.31 Version Negotiation

As there are multiple versions of this protocol, it is not guaranteed that clients and servers will support the same version.

Clients SHOULD be created using the most recent version of this specification. They SHOULD support earlier versions but MAY support a single version.

Servers SHOULD support all versions that have not been deprecated for security reasons. Currently no versions have been deprecated.

In order to negotiate which version to use, the server SHOULD always initially submit the command with the highest version the server supports.

If the client can support that version, it responds with `OPERATION-RESPONSE-PRIMITIVE-TYPE` set to `OPERATION_SUCCESS` or to a suitable error.

If the client cannot support the requested version, it responds with the `OPERATION-RESPONSE-PRIMITIVE-TYPE` set to `ERR_UNSUPPORTED_MSG_VERSION` and the `ver` field set to either:

- The highest supported version that is lower than the version requested. The server SHOULD retry the message with this or, if that version has been deprecated, with a lower version.
- If no such version exists, the lowest supported version. This indicates that the client and server do not have a common version. In this case, the server would need to be upgraded before it can support that client.

5 OTrP Messages

OTrP messages follows a request-response pattern. The OTrP messages are categorized into three types: unprivileged messages, privileged messages, and TEE management messages. OTrP messages SHALL use the following naming structure for request and response strings, where xyz is the message name:

Table 5-1: Request/Response String Naming

OTrP Message	Request/Response String Naming
A request message that is not yet signed	`xyzTBSRequest`
A response message that is not yet signed	`xyzTBSResponse`
A request message sent to a TEE	`xyzRequest`
A response message returned from a TEE	`xyzResponse`

Important: TEE management messages are OPTIONAL and may not be supported by all OTrP Profile implementations.

5.1 Unprivileged Messages

Unprivileged messages SHALL NOT be signed. They SHALL be formatted as follows:

```
{
    "NAME":UNPRIVILEGED-COMMAND-TYPE
}
```

Where:

- `NAME` SHALL be one of the following strings:

```
GetTAInformationRequest
GetTAInformationResponse
```

5.2 Privileged Messages

Privileged messages SHALL always be signed by the sender. Every privileged message SHALL be formatted as follows:

```
{
     "NAME":COMMAND-TYPE
}
```

Where:

- NAME SHALL be one of the following strings:

```
GetDeviceTEEStateRequest
GetDeviceTEEStateResponse
CreateSDRequest
CreateSDResponse
UpdateSDRequest
UpdateSDResponse
DeleteSDRequest
DeleteSDResponse
InstallTARequest
InstallTAResponse
UpdateTARequest
UpdateTAResponse
DeleteTARequest
DeleteTAResponse
```

- COMMAND-TYPE: Contains the corresponding signed message.

5.2.1 Creating a Privileged Message

A privileged message SHALL be created as follows:

- The sender produces a COMMAND-TBS-TYPE JSON object appropriate for the message type.
- The sender uses its private key to calculate a signature over the base64 encoded value of the COMMAND-TBS-TYPE. A privileged message is signed according to the JWS scheme.
- The signature value and the COMMAND-TBS-TYPE are enclosed into a COMMAND-TYPE.
- The COMMAND-TYPE is finally enclosed into the OTrP message, with the command NAME string inserted.

Only OTrP request messages SHALL include the OWE-Cert as a part of the HEADER-TYPE element in the COMMAND-TYPE. The public key associated with the OWE-Cert SHALL be used to verify the signature.

For privacy reasons, an OTrP response message SHALL NOT include its TEE-Cert as a part of the `HEADER-TYPE` element, but it SHALL include its TEE-Cert as a part of the `GetDeviceTEEStateResponse` message.

5.3 TEE Management Messages

TEE management messages are a set of **optional** OTrP messages intended for managing TEEs. Only the OWE whose OWE-Cert chains to the root CA certificate in the OWE-Whitelist of the rSD_{TEE} SHALL be allowed to issue TEE management messages. TEE management messages SHALL always be sent to and processed by an rSD_{TEE}. Prior to sending TEE management messages, the OWE SHALL initiate an OTrP session with rSD_{TEE} by sending a `GetDeviceTEEStateRequest` message.

TEE management messages SHALL be formatted as follows:

```
{
        "NAME":COMMAND-TYPE
}
```

Where:

- `NAME` SHALL be one of the following strings:

```
StoreTEEPropertyRequest
StoreTEEPropertyResponse
FactoryResetRequest
FactoryResetResponse
```

- `COMMAND-TYPE:` Contains the corresponding signed message.

TEE management messages SHALL be created as described in section 5.2.1 for privileged messages.

5.4 GetTAInformationRequest

The `GetTAInformationRequest` is an unprivileged message that SHALL NOT be signed by the sender. It is intended for an REE application to query the status of a TA and the TA metadata from the TEE. This message SHALL always be sent to an rSD$_{TA}$.

```
{
	"GetTAInformationRequest":{
		"ver":"GPD-VERSION-TYPE",
		"taid":"PRINTABLE-STRING-PRIMITIVE-TYPE",
		"spid":"PRINTABLE-STRING-PRIMITIVE-TYPE"
	}
}
```

Where:

- `ver:` The version of the OTrP message structured as `GPD-VERSION-TYPE`.
- `taid:` The base64 encoded UUID representing the TA identifier.
- `spid:` The Service Provider identifier that signs the TA.

The rSD$_{TA}$ SHALL return `GetTAInformationResponse` with the TA metadata only if the given TA is installed using OTrP messages. TEE SHALL return `GetTAInformationResponse` with a failure status for a given TA installed on the device using any method outside the scope of OTrP Profile.

5.4.1 Processing Requirements

Upon receiving the `GetTAInformationRequest` message, the rSD$_{TA}$ SHALL:

- Search all SDs to determine whether the given TA exists.
- Ensure that the `spid` associated with the TA matches the given `spid`.

Upon successfully completing the above steps, the rSD$_{TA}$ prepares a response with the TA metadata. A response message `GetTAInformationResponse` SHALL always be returned regardless of the status of the operation.

5.5 GetTAInformationResponse

In response to a `GetTAInformationRequest`, the rSD SHALL return `GetTAInformationResponse` with the TA metadata for the given TA. The JSON structure for the `GetTAInformationResponse` SHALL be as follows:

```
{
        "GetTAInformationResponse":{
                "ver":"GPD-VERSION-TYPE",
                "status":"OPERATION-RESPONSE-PRIMITIVE-TYPE",
                "taid":"PRINTABLE-STRING-PRIMITIVE-TYPE",
                "taver":"PRINTABLE-STRING-PRIMITIVE-TYPE",
                "sdid":"PRINTABLE-STRING-PRIMITIVE-TYPE",
                "spid":"PRINTABLE-STRING-PRIMITIVE-TYPE",
                "tsmid":"PRINTABLE-STRING-PRIMITIVE-TYPE"
        }
}
```

Where:

- `ver:` The version of the OTrP message structured as `GPD-VERSION-TYPE`.
- `status:` An `OPERATION-RESPONSE-PRIMITIVE-TYPE` indicating the status of the `GetTAInformationRequest` operation. If successful, the value of `status` SHALL be `OPERATION_SUCCESS`; otherwise its value SHALL be an error string listed in section 5.5.1.
- `taid:` The base64 encoded UUID representing the TA identifier.
- `taver:` The string containing the TA version information. In case of failure, the value may be set to `null` or the element may be omitted.
- `sdid:` The base64 encoded UUID of the parent SD of the TA. In case of failure, the value may be set to `null` or the element may be omitted.
- `spid:` The Service Provider identifier that signs the TA. Matches the corresponding `GetTAInformationRequest`.
- `tsmid:` The identifier of the OWE that is authorized to request management operations on the SD. In case of failure, the value may be set to `null` or the element may be omitted.

5.5.1 Error Conditions

If any validation listed in section 5.4.1 fails or if a TEE error occurs, the rSD_{TA} SHALL use an appropriate `OPERATION-RESPONSE-PRIMITIVE-TYPE` (listed below) as the `status` value in the corresponding response message.

- `ERR_TA_NOT_FOUND`
- `ERR_TEE_BUSY`
- `ERR_TEE_FAIL`
- `ERR_TEE_RESOURCE_FULL`
- `ERR_TEE_UNKNOWN`
- `ERR_UNSUPPORTED_MSG_VERSION`

See section 4.14 for details on error strings.

5.6 GetDeviceTEEStateTBSRequest

An OWE SHALL issue a `GetDeviceTEEStateTBSRequest` message to query the DSI of a target device. An OTrP session begins with this message. The message SHALL be signed using the JWS scheme and encapsulated in a `GetDeviceTEEStateRequest` message. However, this message SHALL NOT contain any encrypted content. The JSON structure for the `GetDeviceTEEStateTBSRequest` SHALL be as follows:

```
{
        "GetDeviceTEEStateTBSRequest":{
                "ver":"GPD-VERSION-TYPE",
                "tid":"PRINTABLE-STRING-PRIMITIVE-TYPE",
                "rid":"PRINTABLE-STRING-PRIMITIVE-TYPE",
                "ocspdat":OCSP-ARRAY-TYPE,
                "supportedsigalgs":[SIGNATURE-PRIMITIVE-TYPE]
        }
}
```

Where:

- `ver:` The version of the OTrP message structured as `GPD-VERSION-TYPE`.
- `tid:` A unique value for the ongoing transaction.
- `rid:` A unique value for this message.
- `ocspdat:` `OCSP-ARRAY-TYPE` as described in section 4.25. The first element of the array is the OCSP stapling for validating the OWE-Cert, followed by OCSP stapling for verifying each subsequent intermediate CA in the certificate chain.
- `supportedsigalgs:` (OPTIONAL) A list of signature algorithms supported by the OWE. Its value is an array of `SIGNATURE-PRIMITIVE-TYPE`. If this element is absent, the TEE SHALL use any signature algorithm defined by the `SIGNATURE-PRIMITIVE-TYPE`.

5.6.1 Processing Requirements

Upon receiving the `GetDeviceTEEStateRequest` message, the rSD SHALL:

- Validate the JSON web signature associated with the request, using the OWE-Pub associated with the OWE-Cert.
- Determine whether the OWE-Cert chains to a root CA certificate in the OWE-Whitelist.
- Check the revocation status of the OWE-Cert and its intermediate CA certificates in the chain, using the OCSP stapling.
- Cache the OCSP stapling for subsequent command checking. The TEE MAY use its own clock for OCSP stapling validation.
- Challenge the TFW (if available on the device) to sign a UTF-8 encoded `tid` value. The signed value is included in the `GetDeviceTEEStateResponse` message as a part of `DSI-TYPE`.

Upon successfully completing the above steps, the rSD gathers DSI to prepare a response. A response message `GetDeviceTEEStateResponse` SHALL always be returned regardless of the status of the operation.

5.7 GetDeviceTEEStateTBSResponse

In response to a `GetDeviceTEEStateRequest`, the rSD SHALL return a `GetDeviceTEEStateResponse` that encapsulates a `GetDeviceTEEStateTBSResponse` message. The JSON structure for the `GetDeviceTEEStateTBSResponse` SHALL be as follows:

```
{
        "GetDeviceTEEStateTBSResponse":{
                "ver":"GPD-VERSION-TYPE",
                "status":"OPERATION-RESPONSE-PRIMITIVE-TYPE",
                "rid":"PRINTABLE-STRING-PRIMITIVE-TYPE",
                "tid":"PRINTABLE-STRING-PRIMITIVE-TYPE",
                "signerreq":BOOLEAN,
                "content":CONTENT-ENCRYPTION-TYPE
        }
}
```

Where:

- `ver:` The version of the OTrP message structured as `GPD-VERSION-TYPE`.
- `status:` An `OPERATION-RESPONSE-PRIMITIVE-TYPE` indicating the status of the `GetDeviceTEEStateRequest` operation. If successful, the value of `status` SHALL be `OPERATION_SUCCESS`; otherwise its value SHALL be an error string listed in section 5.7.1.
- `rid:` A unique value identifying the `GetDeviceTEEStateRequest` message.
- `tid:` A unique value identifying the OTrP session. Matches the `tid` value in `GetDeviceTEEStateRequest` message.
- `signerreq:` A Boolean value that indicates whether the OWE should send its signer certificate and OCSP stapling again in the subsequent messages. It is recommended that the `signerreq` value is set to `false`. If the value is set to `false`, the TEE SHALL cache the OWE signer certificate and OCSP stapling.
- `content:` JWE encrypted data as a `CONTENT-ENCRYPTION-TYPE`.

The following JSON structure SHALL be used as an input to the JWE while generating `CONTENT-ENCRYPTION-TYPE`.

```
{
        "dsi":DSI-CONTENT-TYPE,
        "nextnonce":"PRINTABLE-STRING-PRIMITIVE-TYPE"
}
```

Where:

- `dsi:` The `DSI-CONTENT-TYPE` that represents the current device state.

- `nextnonce:` A unique value that the OWE SHALL use as nonce in the next request. See section 2.10.4 for details.

5.7.1 Error Conditions

If any validation listed in section 5.6.1 fails or if a TEE error occurs, the rSD SHALL use an appropriate `OPERATION-RESPONSE-PRIMITIVE-TYPE` (listed below) as the `status` value in the corresponding response message.

- `ERR_OCSP_INVALID`
- `ERR_OWE_NOT_TRUSTED`
- `ERR_REQUEST_INVALID`
- `ERR_TEE_BUSY`
- `ERR_TEE_FAIL`
- `ERR_TEE_RESOURCE_FULL`
- `ERR_TEE_UNKNOWN`
- `ERR_TFW_NOT_TRUSTED`
- `ERR_UNSUPPORTED_CRYPTO_ALG`
- `ERR_UNSUPPORTED_MSG_VERSION`

See section 4.14 for details on error strings.

5.8 CreateSDTBSRequest

An OWE SHALL issue a `CreateSDTBSRequest` message to create a new Security Domain with the given parameters. The message SHALL be signed using the JWS scheme and encapsulated in a `CreateSDRequest` message. This message SHALL always be sent to the rSD_{TA}. The JSON structure for the `CreateSDTBSRequest` SHALL be as follows:

```
{
        "CreateSDTBSRequest":COMMAND-PARAMETER-TYPE
}
```

Within the `COMMAND-PARAMETER-TYPE`, the following JSON structure is used as an input to the JWE while generating `CONTENT-ENCRYPTION-TYPE`.

```
{
        "spid":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "sdid":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "spcert":"CERT-PRIMITIVE-TYPE",
        "tsmid":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "did":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "sd_data":"PRINTABLE-STRING-PRIMITIVE-TYPE"
}
```

Where:

- `spid:` The base64 encoded Service Provider identifier that is to be associated with the SD. See section 2.6 for the `spid` generation.
- `sdid:` The base64 encoded UUID of the SD to be created. The `sdid` SHALL remain unchangeable throughout its life cycle.
- `spcert:` SP-Cert formatted as `CERT-PRIMITIVE-TYPE` that is to be associated with the SD. Only TAs that are signed using a key associated with the SP-Cert SHALL be allowed to be installed in the SD.
- `tsmid:` The identifier of the OWE that issued the request.
- `did:` The base64 encoded SHA-256 hash of the TEE-Cert binary. `did` is used as the device identifier to which the request is issued.
- `sd_data:` (OPTIONAL) The base64url encoded SD personalization data. This element may be used to equip the SD with credentials required to support TMF commands. The format of the SD personalization data SHALL be a DER-encoded `StoredDataObject` as defined in [TMF ASN.1] section 8.3.3.6.

See section 2.10.6 for more information on key use.

5.8.1 Processing Requirements

Before authorizing SD creation, the rSD_{TA} SHALL:

- Validate the JSON web signature associated with this request.
- Determine whether the OWE-Cert chains to a root CA certificate in OWE-Whitelist.
- Check the revocation status of the OWE-Cert chain, using the cached OCSP stapling. If the cache is unavailable or expired, the rSD SHALL return the corresponding response with an error string along with an indication (`signerreq` set to `true`) to provide OCSP stapling in the next request. The OWE may reissue the request with OCSP stapling.
- Compare the `dsihash` value to the SHA-256 hash of the internal `DSI-TYPE` to ensure that the DSI has not changed since the last changes requested by the OWE.
- Compare `nonce` to the last `nextnonce` sent to the OWE to ensure that no new operation has been authorized on SDs and TAs associated with the OWE since the last operation requested by the OWE.
- Decrypt the `ciphertext` element of the `CONTENT-ENCRYPTION-TYPE` to obtain the SD information.
- Validate the format of the `spcert`.
- Verify the `did` to ensure that the request is intended for the correct device.
- Verify that the given `sdid` is valid for the SP, using the process defined in section 2.4.
- Verify that the SD with the given `sdid` does not already exist.
- Verify that the `tsmid` matches the OWE identifier in the OWE-Cert to ensure that the OWE issuing the request has access to the SD. See section 2.9 for details.

Upon successfully completing the above processing, the SD with the given parameters SHALL be created. If the `spid` associated with the SD is not assigned to any SDs on the device, then the TEE SHALL also generate a key pair called SP-AIK. A response message `CreateSDResponse` SHALL always be returned regardless of the status of the operation.

5.9 CreateSDTBSResponse

In response to a `CreateSDRequest`, the rSD_{TA} SHALL return a `CreateSDResponse`, encapsulating the `CreateSDTBSResponse` message. The JSON structure for the `CreateSDTBSResponse` is as follows:

```
{
        "CreateSDTBSResponse":RESPONSE-PARAMETER-TYPE
}
```

Within the `RESPONSE-PARAMETER-TYPE`, the following JSON structure is used as an input to the JWE while generating `CONTENT-ENCRYPTION-TYPE`.

Version 1.1

```
{
        "status":"OPERATION-RESPONSE-PRIMITIVE-TYPE",
        "did":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "sdid":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "spaik":PUB-KEY-ROLE-ARRAY-TYPE,
        "dsi":DSI-CONTENT-TYPE,
        "nextnonce":"PRINTABLE-STRING-PRIMITIVE-TYPE"
}
```

Version 1.0

```
{
        "status":"OPERATION-RESPONSE-PRIMITIVE-TYPE",
        "did":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "sdid":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "spaik":PUB-KEY-TYPE,
        "dsi":DSI-CONTENT-TYPE,
        "nextnonce":"PRINTABLE-STRING-PRIMITIVE-TYPE"
}
```

Where:

- `status:` An `OPERATION-RESPONSE-PRIMITIVE-TYPE` indicating the status of the `CreateSDRequest` operation. If the SD is created successfully, the value of `status` SHALL be `OPERATION_SUCCESS`; otherwise its value SHALL be one of the error strings listed in section 5.9.1.
- `did:` The value of `did` from the `CreateSDRequest`.
- `sdid:` The value of the `sdid` from the `CreateSDRequest`.

- `spaik:`

  Version 1.1

  The SP-AIK-Pub keys structured as a `PUB-KEY-ROLE-ARRAY-TYPE`.

  Version 1.0

  A single SP-AIK-Pub key returned as a `PUB-KEY-TYPE`.

  This element is returned only if the request caused a new SP-AIK to be generated.

- `dsi:` The `DSI-CONTENT-TYPE` for the new device state. This element is returned only when the `nextdsi` is set to `true` in the `CreateSDRequest`.
- `nextnonce:` A unique value that the OWE SHALL use as nonce in the next request. See section 2.10.4 for details.

5.9.1 Error Conditions

If any validation listed in section 5.8.1 fails or if a TEE error occurs, the rSD_{TA} SHALL use an appropriate `OPERATION-RESPONSE-PRIMITIVE-TYPE` (listed below) as the `status` value in the corresponding response message.

- `ERR_DEV_STATE_MISMATCH`
- `ERR_INVALID_UUID`
- `ERR_OCSP_INVALID`
- `ERR_OWE_NOT_TRUSTED`
- `ERR_REQUEST_INVALID`
- `ERR_REVERT_OPERATION`
- `ERR_SDID_ALREADY_USED`
- `ERR_SPCERT_INVALID`
- `ERR_TEE_BUSY`
- `ERR_TEE_FAIL`
- `ERR_TEE_RESOURCE_FULL`
- `ERR_TEE_UNKNOWN`
- `ERR_UNSUPPORTED_CRYPTO_ALG`
- `ERR_UNSUPPORTED_MSG_VERSION`

See section 4.14 for details on error strings.

Note: If the OWE receives `ERR_REVERT_OPERATION`, it is recommended that the OWE remove the recently created SD; otherwise the DSI value will be inconsistent.

5.10 UpdateSDTBSRequest

An OWE SHALL issue an `UpdateSDTBSRequest` message to update SD metadata with the given parameters. The message SHALL be signed using the JWS scheme and encapsulated in an `UpdateSDRequest` message. This message SHALL always be sent to rSD_{TA}. A JSON structure for the `UpdateSDTBSRequest` SHALL be as follows:

```
{
        "UpdateSDTBSRequest":COMMAND-PARAMETER-TYPE
}
```

Within the `COMMAND-PARAMETER-TYPE`, the following JSON structure is used as an input to the JWE while generating `CONTENT-ENCRYPTION-TYPE`:

Version 1.1

```
{
        "tsmid":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "spid":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "did":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "sdid":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "changes":{
                "spcert":["CERT-PRIMITIVE-TYPE"],
                "deloldspcert":["PRINTABLE-STRING-PRIMITIVE-TYPE"],
                "sd_data":"PRINTABLE-STRING-PRIMITIVE-TYPE"
        }
}
```

Version 1.0

```
{
        "tsmid":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "spid":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "did":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "sdid":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "changes":{
                "newspid":"PRINTABLE-STRING-PRIMITIVE-TYPE",
                "spcert":["CERT-PRIMITIVE-TYPE"],
                "deloldspcert":["PRINTABLE-STRING-PRIMITIVE-TYPE"],
                "sd_data":"PRINTABLE-STRING-PRIMITIVE-TYPE"
```

```
        }
}
```

Where:

- `tsmid:` The identifier of the OWE that issued the request.
- `spid:` The base64 encoded service provide identifier that is associated with the SD.
- `did:` The base64 encoded SHA-256 hash of the TEE-Cert binary. `did` is used as the device identifier to which the request is issued.
- `sdid:` The base64 encoded UUID representing the SD to be updated.
- `changes:` A JSON structure containing parameters to be updated. All elements within this JSON structure are **optional**.
- `newspid:`

  Version 1.1

  Deprecated in v1.1.

  > The SD name is derived from the `spid`, so changing the `spid` effectively changes the SD name. In a pure OTrP system this is permissible, but not in TMF.

  Version 1.0

  The new base64 encoded service provide identifier that is to be associated with the SD. If the `newspid` is not associated with any existing SDs in the device, the rSD SHALL generate a new SP-AIK key pair for the `newspid`.
- `spcert:` An array of SP-Certs formatted as `CERT-PRIMITIVE-TYPE` that is to be associated with the SD.
- `deloldspcert:` The base64 encoded SHA-256 hash value of SP-Certs previously assigned to the SD that are to be deleted.

  Note: Deleting the certificate without supplying a new certificate would make it impossible to verify new OTrP sessions.
- `sd_data:` The base64 encoded SD personalization data. This element may be used to equip the SD with credentials required to support TMF commands. The format of the SD personalization data SHALL be a DER-encoded `StoredDataObject` as defined in [TMF ASN.1] section 8.3.3.6. See section 2.10.6 for more information on key use. If the OWE updates keys, the SD SHOULD generate a new SP-AIK-Pub.

  Important: Beginning with version 1.1, implementations SHALL be able to update keys and certificates.

5.10.1 Processing Requirements

Before authorizing the SD update, the rSD$_{TA}$ SHALL:

- Validate the JSON web signature associated with the request.
- Determine whether the OWE-Cert chains to a root CA certificate in OWE-Whitelist.
- Check the revocation status of the OWE-Cert chain, using the cached OCSP stapling. If the cache is unavailable or expired, the rSD SHALL return the corresponding response with an error string along with an indication (`signerreq` set to `true`) to provide OCSP stapling in the next request. The OWE may reissue the request with OCSP stapling.

- Compare the `dsihash` value to the SHA-256 hash of the internal `DSI-TYPE` to ensure that the DSI has not changed since the last changes requested by the OWE.
- Compare `nonce` to the last `nextnonce` sent to the OWE to ensure that no new operation has been authorized on SDs and TAs associated with the OWE since the last operation requested by the OWE.
- Decrypt the `ciphertext` element of the `CONTENT-ENCRYPTION-TYPE` to obtain the update parameters.
- Verify the `did` to ensure that the request is intended for the correct device.
- Verify that the given `sdid` is valid for the SP, using the process defined in section 2.4.
- Verify that the SD with the given `sdid` exists.
- Verify that the `tsmid` matches the OWE identifier in the OWE-Cert to ensure that the OWE issuing the request has access to the SD. See section 2.9 for details.

Upon successfully completing the above requirements, the specified SD is updated with the given parameters. If the update operation results in the generation of a new SP-AIK, the newly generated SP-AIK SHALL replace the existing SP-AIK. A response message `UpdateSDResponse` SHALL always be returned regardless of the status of the operation.

5.11 UpdateSDTBSResponse

In response to an `UpdateSDRequest`, the rSD_{TA} SHALL return an `UpdateSDResponse`, encapsulating the `UpdateSDTBSResponse` message. The JSON structure for the `UpdateSDTBSResponse` SHALL be as follows:

```
{
        "UpdateSDTBSResponse":RESPONSE-PARAMETER-TYPE
}
```

Within the `RESPONSE-PARAMETER-TYPE`, the following JSON structure is used as an input to the JWE while generating `CONTENT-ENCRYPTION-TYPE`:

Version 1.1

```
{
        "status":"OPERATION-RESPONSE-PRIMITIVE-TYPE",
        "did":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "spaik":PUB-KEY-ROLE-ARRAY-TYPE,
        "dsi":DSI-CONTENT-TYPE,
        "nextnonce":"PRINTABLE-STRING-PRIMITIVE-TYPE"
}
```

Version 1.0

```
{
        "status":"OPERATION-RESPONSE-PRIMITIVE-TYPE",
        "did":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "spaik":PUB-KEY-TYPE,
        "dsi":DSI-CONTENT-TYPE,
        "nextnonce":"PRINTABLE-STRING-PRIMITIVE-TYPE"
}
```

Where:

- `status:` An `OPERATION-RESPONSE-PRIMITIVE-TYPE` indicating the status of the `UpdateSDRequest` operation. If the SD is updated successfully, the value of `status` SHALL be `OPERATION_SUCCESS`; otherwise its value SHALL be an error string listed in section 5.11.1.
- `did:` The value of `did` from the `UpdateSDRequest`.

- `spaik:`

  Version 1.1

  The SP-AIK-Pub keys structured as a `PUB-KEY-ROLE-ARRAY-TYPE`.

  Version 1.0

  A single SP-AIK-Pub key returned as a `PUB-KEY-TYPE`.

  This element is returned only if the `UpdateSDRequest` causes the rSD_{TA} to generate a new SP-AIK.

- `dsi:` The `DSI-CONTENT-TYPE` for the new device state. This element is returned only when the `nextdsi` is set to `true` in the `UpdateSDRequest`.
- `nextnonce:` A unique value that the OWE SHALL use as nonce in the next request. See section 2.10.4 for details.

5.11.1 Error Conditions

If any validation listed in section 5.10.1 fails or if a TEE error occurs, the rSD_{TA} SHALL use an appropriate `OPERATION-RESPONSE-PRIMITIVE-TYPE` (listed below) as the `status` value in the corresponding response message.

- `ERR_DEV_STATE_MISMATCH`
- `ERR_INVALID_UUID`
- `ERR_OCSP_INVALID`
- `ERR_OWE_NOT_TRUSTED`
- `ERR_REQUEST_INVALID`
- `ERR_SPCERT_INVALID`
- `ERR_TEE_BUSY`
- `ERR_TEE_FAIL`
- `ERR_TEE_RESOURCE_FULL`
- `ERR_TEE_UNKNOWN`
- `ERR_UNSUPPORTED_CRYPTO_ALG`
- `ERR_UNSUPPORTED_MSG_VERSION`
- `ERR_UPDATING_DATA`

See section 4.14 for details on error strings.

Note: If the OWE receives `ERR_REVERT_OPERATION`, it is recommended that the OWE remove the recently created SD; otherwise the DSI value will be inconsistent.

5.12 DeleteSDTBSRequest

An OWE SHALL issue a `DeleteSDTBSRequest` message to delete a specified SD and optionally delete all TAs contained within the SD. The message SHALL be signed using the JWS scheme and encapsulated in a `DeleteSDRequest` message. This message SHALL always be sent to the rSD_{TA}. The JSON structure for the `DeleteSDTBSRequest` SHALL be as follows:

```
{
        "DeleteSDTBSRequest":COMMAND-PARAMETER-TYPE
}
```

Within the `COMMAND-PARAMETER-TYPE`, the following JSON structure is used as an input to the JWE while generating `CONTENT-ENCRYPTION-TYPE`:

```
{
        "tsmid":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "did":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "sdid":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "deletetas":BOOLEAN
}
```

Where:

- `tsmid:` The identifier of the OWE that issued the request.
- `did:` The base64 encoded SHA-256 hash of the TEE-Cert binary. `did` is used as the device identifier to which the request is issued.
- `sdid:` The base64 encoded UUID representing the SD to be deleted.
- `deletetas:` A Boolean value indicating whether to delete all TAs within the SD. If set to `false`, deleting an SD with one or more TAs installed SHALL cause a failure.

5.12.1 Processing Requirements

Before authorizing the deletion of the SD, the rSD_{TA} SHALL:

- Validate the JSON web signature associated with the request.
- Determine whether the OWE-Cert chains to a root CA certificate in the OWE-Whitelist.
- Check the revocation status of the OWE-Cert chain, using the cached OCSP stapling. If the cache is unavailable or expired, the rSD SHALL return the corresponding response with an error string along with an indication (`signerreq` set to `true`) to provide OCSP stapling in the next request. The OWE may reissue the request with OCSP stapling.
- Compare the `dsihash` value to the SHA-256 hash of the internal `DSI-TYPE` to ensure that the DSI has not changed since the last changes requested by the OWE.
- Compare `nonce` to the last `nextnonce` sent to the OWE to ensure that no new operation has been authorized on SDs and TAs associated with the OWE since the last operation requested by the OWE.
- Decrypt the `ciphertext` element of the `CONTENT-ENCRYPTION-TYPE` to obtain the deletion parameters.
- Verify the `did` to ensure that the request is intended for the correct device.
- Verify that the `tsmid` matches the OWE identifier in the OWE-Cert to ensure that the OWE issuing the request has access to the SD. See section 2.9 for details.
- Ensure that, if `deletetas` is set to `false`, the SD contains no TAs; otherwise the deletion SHALL be aborted, resulting in a failure.

Upon successfully completing the above requirements, the specified SD SHALL be deleted. A response message `DeleteSDResponse` SHALL always be returned regardless of the status of the operation.

5.13 DeleteSDTBSResponse

In response to a `DeleteSDRequest` command, the rSD_{TA} SHALL return a `DeleteSDResponse`, encapsulating the `DeleteSDTBSResponse` message. The JSON structure for the `DeleteSDTBSResponse` is as follows:

```
{
        "DeleteSDTBSResponse":RESPONSE-PARAMETER-TYPE
}
```

Within the `RESPONSE-PARAMETER-TYPE`, the following JSON structure is used as an input to the JWE while generating `CONTENT-ENCRYPTION-TYPE`.

```
{
        "status":"OPERATION-RESPONSE-PRIMITIVE-TYPE",
        "did":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "dsi":DSI-CONTENT-TYPE,
        "nextnonce":"PRINTABLE-STRING-PRIMITIVE-TYPE"
}
```

Where:

- `status:` An `OPERATION-RESPONSE-PRIMITIVE-TYPE` indicating the status of the `DeleteSDRequest` operation. If the SD is deleted successfully, the value of `status` SHALL be `OPERATION_SUCCESS`; otherwise its value SHALL be an error string listed in section 5.13.1.
- `did:` The value of `did` from the `DeleteSDRequest`.
- `dsi:` The `DSI-CONTENT-TYPE` for the new device state. This element is returned only when the `nextdsi` is set to `true` in the `DeleteSDRequest`.
- `nextnonce:` A unique value that the OWE SHALL use as nonce in the next request. See section 2.10.4 for details.

5.13.1 Error Conditions

If any validation listed in section 5.12.1 fails or if a TEE error occurs, the rSD_{TA} SHALL use an appropriate `OPERATION-RESPONSE-PRIMITIVE-TYPE` (listed below) as the `status` value in the corresponding response message.

- `ERR_DEV_STATE_MISMATCH`
- `ERR_OCSP_INVALID`
- `ERR_OWE_NOT_TRUSTED`
- `ERR_REQUEST_INVALID`
- `ERR_SD_NOT_EMPTY`
- `ERR_SPCERT_INVALID`
- `ERR_TEE_BUSY`
- `ERR_TEE_FAIL`
- `ERR_TEE_RESOURCE_FULL`
- `ERR_TEE_UNKNOWN`
- `ERR_UNSUPPORTED_CRYPTO_ALG`
- `ERR_UNSUPPORTED_MSG_VERSION`

See section 4.14 for details on error strings.

5.14 InstallTATBSRequest

An OWE SHALL issue an `InstallTATBSRequest` message to install a specified TA into a specified Security Domain. The message SHALL be signed using the JWS scheme and encapsulated in an `InstallTARequest` message. This message SHALL always be sent to the rSD_{TA}. The JSON structure for `InstallTATBSRequest` is as follows:

```
{
	"InstallTATBSRequest":COMMAND-PARAMETER-TYPE
}
```

Within the `PROTECTED-HEADER-TYPE` (section 4.4), the following JSON structure is used as an input to the JWE while generating `CONTENT-ENCRYPTION-TYPE`.

```
{
	"tsmid":"PRINTABLE-STRING-PRIMITIVE-TYPE",
	"did":"PRINTABLE-STRING-PRIMITIVE-TYPE",
	"spid":"PRINTABLE-STRING-PRIMITIVE-TYPE",
	"sdid":"PRINTABLE-STRING-PRIMITIVE-TYPE",
	"spcert":"CERT-PRIMITIVE-TYPE",
	"taid":"PRINTABLE-STRING-PRIMITIVE-TYPE",
	"taver":"PRINTABLE-STRING-PRIMITIVE-TYPE",
	"pop_data":POP-TYPE
}
```

Where:

- `tsmid:` The identifier of the OWE that issued the request.
- `did:` The base64 encoded SHA-256 hash of the TEE-Cert binary. `did` is used as the device identifier to which the request is issued.
- `spid:` The base64 encoded Service Provider identifier that is associated with the SD.
- `sdid:` The base64 encoded UUID of the SD where the TA is to be installed.
- `spcert:` (OPTIONAL) The SP-Cert formatted as `CERT-PRIMITIVE-TYPE` that signed the TA. This element is provided when the TA is signed with an SP-Cert that was not previously associated with the SD.
- `taid:` The base64 encoded UUID of the TA to be installed.
- `taver:` The string containing the TA version information.
- `pop_data`: (OPTIONAL) `POP-TYPE` value SHALL be included when the given `taid` is a UUID version 5. It is used to perform a verification of proof of possession of a UUID version 5 as defined in [TMF ASN.1] section 8.3.3.7. (See details in Annex D.)

Additionally, the `InstallTATBSRequest` SHALL include the following JSON elements in the `COMMAND-PARAMETER-TYPE`.

```
"encrypted_ta_bin":CONTENT-ENCRYPTION-TYPE
"encrypted_ta_data":CONTENT-ENCRYPTION-TYPE
```

- `encrypted_ta_bin:` An encrypted TA binary structured as a `CONTENT-ENCRYPTION-TYPE` where the CEK is wrapped using the SP-AIK-Pub.
- `encrypted_ta_data:` (OPTIONAL) An encrypted TA personalization data structured as a `CONTENT-ENCRYPTION-TYPE` where the CEK is wrapped using the SP-AIK-Pub. The TA should be able to access the personalization data via interfaces defined in [TEE Core]. The format of the TA personalization data SHALL be a DER-encoded `StoredDataObject` as defined in [TMF ASN.1] section 8.3.3.6.

5.14.1 Processing Requirements

Before authorizing the deletion of the SD, the rSD$_{TA}$ SHALL:

- Validate the JSON web signature associated with the request.
- Determine whether the OWE-Cert chains to a root CA certificate in the OWE-Whitelist.
- Check the revocation status of the OWE-Cert chain, using the cached OCSP stapling. If the cache is unavailable or expired, the rSD SHALL return the corresponding response with an error string along with an indication (`signerreq` set to `true`) to provide OCSP stapling in the next request. The OWE may reissue the request with OCSP stapling.
- Compare the `dsihash` value to the SHA-256 hash of the internal `DSI-TYPE` to ensure that the DSI has not changed since the last changes requested by the OWE.
- Compare `nonce` to the last `nextnonce` sent to the OWE to ensure that no new operation has been authorized on SDs and TAs associated with the OWE since the last operation requested by the OWE.
- Decrypt the `ciphertext` element of the `CONTENT-ENCRYPTION-TYPE` to obtain the TA information.
- Validate the format of the `spcert`.
- Verify the `did` to ensure that the request is intended for the correct device.
- Verify that the `tsmid` matches the OWE identifier in the OWE-Cert to ensure that the OWE issuing the request has access to the SD. See section 2.9 for details.
- If `taid` is a UUID version 5, validate the proof of possession of the TA UUID as defined in [TMF ASN.1] section 8.3.3.7. (See details in Annex D.)
- Verify that, if the TA already exists in the device, the version of the TA to be installed is higher than the existing TA.
- Use SP-AIK-Priv key to decrypt TA binary and personalization data. If the version of the TA associated with the TA binary is different than the `taver` element, the rSD$_{TA}$ SHALL abort the update process.
- Validate the TA signature using an SP-Cert associated with the SD. The TA signing mechanism may be specific to the TEE OS.

Upon successfully completing the above requirements, the given TA SHALL be installed into the specified SD. A response message `InstallTAResponse` SHALL always be returned regardless of the status of the operation.

5.15 InstallTATBSResponse

In response to an `InstallTARequest`, the rSD_{TA} SHALL return an `InstallTAResponse`, encapsulating the `InstallTATBSResponse` message. The JSON structure for the `InstallTATBSResponse` SHALL be as follows:

```
{
        "InstallTATBSResponse":RESPONSE-PARAMETER-TYPE
}
```

Within the `RESPONSE-PARAMETER-TYPE`, the following JSON structure SHALL be used as an input to the JWE while generating `CONTENT-ENCRYPTION-TYPE`.

```
{
        "status":"OPERATION-RESPONSE-PRIMITIVE-TYPE",
        "did":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "dsi":DSI-CONTENT-TYPE,
        "nextnonce":"PRINTABLE-STRING-PRIMITIVE-TYPE"
}
```

Where:

- `status:` An `OPERATION-RESPONSE-PRIMITIVE-TYPE` indicating the status of the `InstallTARequest` operation. If the TA is installed successfully, the value of `status` SHALL be `OPERATION_SUCCESS`; otherwise its value SHALL be an error string listed in section 5.15.1.
- `did:` The value of `did` from previous messages.
- `dsi:` The `DSI-CONTENT-TYPE` for the new device state. This element is returned only when the `nextdsi` is set to `true` in the `InstallTARequest`.
- `nextnonce:` A unique value that the OWE SHALL use as nonce in the next request. See section 2.10.4 for details.

5.15.1 Error Conditions

If any validation listed in section 5.14.1 fails or if a TEE error occurs, the rSD_{TA} SHALL use an appropriate `OPERATION-RESPONSE-PRIMITIVE-TYPE` (listed below) as the `status` value in the corresponding response message.

- `ERR_DEV_STATE_MISMATCH`
- `ERR_INVALID_UUID`
- `ERR_OCSP_INVALID`
- `ERR_OWE_NOT_TRUSTED`
- `ERR_REQUEST_INVALID`
- `ERR_REVERT_OPERATION`
- `ERR_SPCERT_INVALID`
- `ERR_TA_ALREADY_INSTALLED`
- `ERR_TA_INVALID`
- `ERR_TEE_BUSY`
- `ERR_TEE_FAIL`
- `ERR_TEE_RESOURCE_FULL`
- `ERR_TEE_UNKNOWN`
- `ERR_UNSUPPORTED_CRYPTO_ALG`
- `ERR_UNSUPPORTED_MSG_VERSION`

See section 4.14 for details on error strings.

Note: If the OWE receives `ERR_REVERT_OPERATION`, it is recommended that the OWE remove the recently created SD; otherwise the DSI value will be inconsistent.

5.16 UpdateTATBSRequest

An OWE SHALL issue an `UpdateTATBSRequest` message to update TA binary and/or TA personalization data. The message SHALL be signed using the JWS scheme and encapsulated in an `UpdateTARequest` message. This message SHALL always be sent to rSD_{TA}. The JSON structure for the `UpdateTATBSRequest` SHALL be as follows:

```
{
        "UpdateTATBSRequest":COMMAND-PARAMETER-TYPE
}
```

Within the `COMMAND-PARAMETER-TYPE`, the following JSON structure SHALL be used as an input to the JWE while generating `CONTENT-ENCRYPTION-TYPE`:

```
{
        "tsmid":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "did":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "spid":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "sdid":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "spcert":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "taid":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "newtaver":"PRINTABLE-STRING-PRIMITIVE-TYPE"",
        "pop_data":POP-TYPE
}
```

Where:

- `tsmid:` The identifier of the OWE that issued the request.
- `did:` The base64 encoded SHA-256 hash of the TEE-Cert binary. `did` is used as the device identifier to which the request is issued.
- `spid:` The base64 encoded Service Provider identifier that is associated with the SD.
- `sdid:` The base64 encoded UUID of the SD where the TA is to be installed.
- `spcert:` (OPTIONAL) The SP-Cert formatted as `CERT-PRIMITIVE-TYPE` that signed the TA. This element is provided when the TA is signed with a SP-Cert that was not previously associated with the SD.
- `taid:` The base64 encoded UUID of the TA to be installed.
- `newtaver:` (OPTIONAL) The string containing the TA version information that is to be updated.
- `pop_data`: (OPTIONAL) The value of `POP-TYPE` SHALL be included when the given `taid` is a UUID version 5. It is used to perform a verification of proof of possession of a UUID version 5 as defined in [TMF ASN.1] section 8.3.3.7. (See details in Annex D.)

Additionally, the `UpdateTATBSRequest` SHALL include at least one of the following JSON elements in the `COMMAND-PARAMETER-TYPE`.

```
"encrypted_ta_bin":CONTENT-ENCRYPTION-TYPE
"encrypted_ta_data":CONTENT-ENCRYPTION-TYPE
```

- `encrypted_ta_bin:` An encrypted TA binary that replaces the existing TA binary, structured as a `CONTENT-ENCRYPTION-TYPE` where the CEK is wrapped using the SP-AIK-Pub.
- `encrypted_ta_data:` (OPTIONAL) An encrypted TA personalization data to replace the existing TA personalization data, structured as a `CONTENT-ENCRYPTION-TYPE` where the CEK is wrapped using the SP-AIK-Pub. The TA should be able to access the personalization data via interfaces defined in [TEE Core]. The format of the TA personalization data SHALL be a DER-encoded `StoredDataObject` as defined in [TMF ASN.1] section 8.3.3.6.

5.16.1 Processing Requirements

Before authorizing the update on the TA, the rSD_{TA} SHALL:

- Validate the JSON web signature associated with the request.
- Determine whether the OWE-Cert chains to a root CA certificate in OWE-Whitelist.
- Check the revocation status of the OWE-Cert chain, using the cached OCSP stapling. If the cache is unavailable or expired, the `rSD` SHALL return the corresponding response with an error string along with an indication (`signerreq` set to `true`) to provide OCSP stapling in the next request. The OWE may reissue the request with OCSP stapling.
- Compare the `dsihash` value to the SHA-256 hash of the internal `DSI-TYPE` to ensure that the DSI has not changed since the last changes requested by the OWE.
- Compare `nonce` to the last `nextnonce` sent to the OWE to ensure that no new operation has been authorized on SDs and TAs associated with the OWE since the last operation requested by the OWE.
- Decrypt the `ciphertext` element of the `CONTENT-ENCRYPTION-TYPE` to obtain the TA information.
- Verify the `did` to ensure that the request is intended for the correct device.
- Verify that the `tsmid` matches the OWE identifier in the OWE-Cert to ensure that the OWE issuing the request has access to the SD. See section 2.9 for details.
- If `taid` is a UUID version 5, validate the proof of possession of the TA UUID as defined in [TMF ASN.1] section 8.3.3.7. (See details in Annex D.)
- Verify that the version of the TA to be updated is higher than the one that is currently installed. If the update command does not contain the `encrypted_ta_bin` element, the rSD_{TA} SHALL ignore the `newtaver` element.
- Use the SP-AIK-Priv key to decrypt TA binary and personalization data. If the version of the TA associated with the TA binary is different than the `newtaver` element, the rSD_{TA} SHALL abort the update process.
- Validate the TA signature using an SP-Cert associated with the SD. The TA signing mechanism may be specific to the TEE OS.

Upon successfully completing the above steps, the given TA SHALL be updated. Prior to an update, the TA SHALL be forcefully shut down as defined in [TMF ASN.1] section 11. A response message `UpdateTAResponse` SHALL always be returned regardless of the status of the operation.

5.17 UpdateTATBSResponse

In response to an `UpdateTARequest`, the TEE SHALL return an `UpdateTAResponse`, encapsulating the `UpdateTATBSResponse` message. The JSON structure for the `UpdateTATBSResponse` SHALL be as follows:

```
{
        "UpdateTATBSResponse":RESPONSE-PARAMETER-TYPE
}
```

Within the `RESPONSE-PARAMETER-TYPE`, the following JSON structure is used as an input to the JWE while generating `CONTENT-ENCRYPTION-TYPE`.

```
{
        "status":"OPERATION-RESPONSE-PRIMITIVE-TYPE",
        "did":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "dsi":DSI-CONTENT-TYPE,
        "nextnonce":"PRINTABLE-STRING-PRIMITIVE-TYPE"
}
```

Where:

- `status:` An `OPERATION-RESPONSE-PRIMITIVE-TYPE` indicating the status of the `UpdateTARequest` operation. If the TA is successfully updated, the value of `status` SHALL be `OPERATION_SUCCESS`; otherwise its value SHALL be an error string listed in section 5.17.1.
- `did:` The value of `did` from previous messages.
- `dsi:` The `DSI-CONTENT-TYPE` for the new device state. This element is returned only when the `nextdsi` is set to `true` in the `UpdateTARequest`.
- `nextnonce:` A unique value that the OWE SHALL use as nonce in the next request. See section 2.10.4 for details.

5.17.1 Error Conditions

If any validation listed in section 5.16.1 fails or if a TEE error occurs, the rSD_{TA} SHALL use an appropriate `OPERATION-RESPONSE-PRIMITIVE-TYPE` (listed below) as the `status` value in the corresponding response message.

- `ERR_DEV_STATE_MISMATCH`
- `ERR_INVALID_UUID`
- `ERR_OCSP_INVALID`
- `ERR_OWE_NOT_TRUSTED`
- `ERR_REQUEST_INVALID`
- `ERR_SPCERT_INVALID`
- `ERR_TA_ALREADY_INSTALLED`
- `ERR_TA_INVALID`
- `ERR_TEE_BUSY`
- `ERR_TEE_FAIL`
- `ERR_TEE_RESOURCE_FULL`
- `ERR_TEE_UNKNOWN`
- `ERR_UNSUPPORTED_CRYPTO_ALG`
- `ERR_UNSUPPORTED_MSG_VERSION`
- `ERR_UPDATING_DATA`

See section 4.14 for details on error strings.

Note: If the OWE receives `ERR_REVERT_OPERATION`, it is recommended that the OWE remove the recently created SD; otherwise the DSI value will be inconsistent.

5.18 DeleteTATBSRequest

An OWE SHALL issue a `DeleteTATBSRequest` message to delete a specific TA from a specified SD. The message SHALL be signed using the JWS scheme and encapsulated in a `DeleteTARequest` message. This message SHALL always be sent to rSD_{TA}. The JSON structure for the `DeleteTATBSRequest` is as follows:

```
{
        "DeleteTATBSRequest":COMMAND-PARAMETER-TYPE
}
```

Within the `COMMAND-PARAMETER-TYPE`, the following JSON structure SHALL be used as an input to the JWE while generating `CONTENT-ENCRYPTION-TYPE`.

```
{
        "tsmid":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "did":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "sdid":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "taid":"PRINTABLE-STRING-PRIMITIVE-TYPE"
}
```

Where:

- `tsmid:` The identifier of the OWE that issued the request.
- `did:` The base64 encoded SHA-256 hash of the TEE-Cert binary. `did` is used as the device identifier to which the request is issued.
- `sdid:` The base64 encoded UUID of the SD where the TA is installed.
- `taid:` The base64 encoded UUID of a TA that is to be deleted.

5.18.1 Processing Requirements

Before authorizing the deletion of the SD, the rSD_{TA} SHALL:

- Validate the JSON web signature associated with the request.
- Determine whether the OWE-Cert chains to a root CA certificate in OWE-Whitelist.
- Check the revocation status of the OWE-Cert chain, using the cached OCSP stapling. If the cache is unavailable or expired, the rSD SHALL return the corresponding response with an error string along with an indication (`signerreq` set to `true`) to provide OCSP stapling in the next request. The OWE may reissue the request with OCSP stapling.
- Compare the `dsihash` value to the SHA-256 hash of the internal `DSI-TYPE` to ensure that the DSI has not changed since the last changes requested by the OWE.
- Compare `nonce` to the last `nextnonce` sent to the OWE to ensure that no new operation has been authorized on SDs and TAs associated with the OWE since the last operation requested by the OWE.
- Decrypt the `ciphertext` element of the `CONTENT-ENCRYPTION-TYPE` to obtain the TA information for deletion.

- Verify the `did` to ensure that the request is intended for the correct device.
- Verify that the `tsmid` matches the OWE identifier in the OWE-Cert to ensure that the OWE issuing the request has access to the SD. See section 2.9 for details.

Upon successfully completing the above requirements, the given TA SHALL be deleted. Prior to deletion, the TA SHALL be forcefully shut down as defined in [TMF ASN.1] section 11. A response message `DeleteTAResponse` SHALL always be returned regardless of the status of the operation.

5.19 DeleteTATBSResponse

In response to a `DeleteTARequest` command, the rSD_{TA} SHALL return a `DeleteTAResponse`, encapsulating the `DeleteTATBSResponse` message. The JSON structure for the `DeleteTATBSResponse` SHALL be as follows:

```
{
        "DeleteTATBSResponse":RESPONSE-PARAMETER-TYPE
}
```

Within the `RESPONSE-PARAMETER-TYPE`, the following JSON structure is used as an input to the JWE while generating `CONTENT-ENCRYPTION-TYPE`.

```
{
        "status":"OPERATION-RESPONSE-PRIMITIVE-TYPE",
        "did":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "dsi":DSI-CONTENT-TYPE,
        "nextnonce":"PRINTABLE-STRING-PRIMITIVE-TYPE"
}
```

Where:

- `status:` An `OPERATION-RESPONSE-PRIMITIVE-TYPE` indicating the status of the `DeleteSDRequest` operation. If the TA is deleted successfully, the value of `status` SHALL be `OPERATION_SUCCESS`; otherwise its value SHALL be an error string listed in section 5.19.1.
- `did:` The value of `did` from the `DeleteSDRequest`.
- `dsi:` The `DSI-CONTENT-TYPE` for the new device state. This element is returned only when the `nextdsi` is set to `true` in the `DeleteSDRequest`.
- `nextnonce:` A unique value that the OWE SHALL use as nonce in the next request. See section 2.10.4 for details.

5.19.1 Error Conditions

If any validation listed in section 5.18.1 fails or if a TEE error occurs, the rSD_{TA} SHALL use an appropriate `OPERATION-RESPONSE-PRIMITIVE-TYPE` (listed below) as the `status` value in the corresponding response message.

- `ERR_DEV_STATE_MISMATCH`
- `ERR_OCSP_INVALID`
- `ERR_OWE_NOT_TRUSTED`
- `ERR_REQUEST_INVALID`
- `ERR_TA_NOT_FOUND`
- `ERR_TEE_BUSY`
- `ERR_TEE_FAIL`

- `ERR_TEE_RESOURCE_FULL`
- `ERR_TEE_UNKNOWN`
- `ERR_UNSUPPORTED_CRYPTO_ALG`
- `ERR_UNSUPPORTED_MSG_VERSION`

See section 4.14 for details on error strings.

5.20 StoreTEEPropertyTBSRequest

An OWE SHALL issue a `StoreTEEPropertyTBSRequest` message to store, update, or delete TEE properties. The message SHALL be signed using the JWS scheme and encapsulated in a `StoreTEEPropertyRequest` message. This message SHALL always be issued to an rSD_{TEE}.

TEE properties are described in [TMF ASN.1] section A.5.

The only property that can be updated is `gpd.tee.tmf.resetpreserved.entities`, which is used to indicate entities as UUIDs to be preserved across a Factory Reset operation on the TEE.

The JSON structure for the `StoreTEEPropertyTBSRequest` is as follows:

```
{
        "StoreTEEPropertyTBSRequest":COMMAND-PARAMETER-TYPE
}
```

The following JSON elements will be used as input to the `CONTENT-ENCRYPTION-TYPE` within `COMMAND-TYPE`.

```
{
        "tsmid":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "did":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "property":"gpd.tee.tmf.resetpreserved.entities",
        "value":{
                        "taids":UUID-ARRAY-TYPE,
                        "sdids":UUID-ARRAY-TYPE
                }
}
```

Where:

- `tsmid:` The identifier of the OWE that issued the request.
- `did:` The base64 encoded SHA-256 hash of the TEE-Cert binary. `did` is used as the device identifier to which the request is issued.
- `property:` OTrP Profile SHALL support only the TEE property `gpd.tee.tmf.resetpreserved.entities`.
- `value:` The value of the TEE property.
- `taids:` UUIDS of TAs structured as `UUID-ARRAY-TYPE` that SHALL be preserved across a Factory Reset operation on TEE.
- `sdids:` UUIDS of SDs structured as `UUID-ARRAY-TYPE` that SHALL be preserved across a Factory Reset operation on TEE.

The `StoreTEEPropertyTBSRequest` SHALL always replace the previous value of the TEE property.

5.20.1 Processing Requirements

Upon receiving the `StoreTEEPropertyRequest` message, the rSD_{TEE} SHALL:

- Validate the JSON web signature associated with the request, using the OWE-Pub associated with the OWE-Cert.
- Determine whether the OWE-Cert chains to a root CA certificate in its OWE-Whitelist.
- Check the revocation status of the OWE-Cert chain, using the cached OCSP stapling. If the cache is unavailable or expired, the rSD SHALL return the corresponding response with an error string along with an indication (`signerreq` set to `true`) to provide OCSP stapling in the next request. The OWE may reissue the request with OCSP stapling.

Upon successfully completing the above requirements, the rSD_{TEE} SHALL replace the TEE property with the given value. A response message `StoreTEEPropertyResponse` SHALL always be returned regardless of the status of the operation.

5.21 StoreTEEPropertyTBSResponse

In response to a `StoreTEEPropertyRequest` command, the rSD$_{TEE}$ SHALL return a `StoreTEEPropertyResponse`, encapsulating the `StoreTEEPropertyTBSResponse` message. The JSON structure for the `StoreTEEPropertyTBSResponse` is as follows:

```
{
        "StoreTEEPropertyTBSResponse":RESPONSE-PARAMETER-TYPE
}
```

Within the `RESPONSE-PARAMETER-TYPE`, the following JSON structure is used as an input to the JWE while generating `CONTENT-ENCRYPTION-TYPE`.

```
{
        "status":"OPERATION-RESPONSE-PRIMITIVE-TYPE",
        "did":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "dsi":DSI-CONTENT-TYPE,
        "nextnonce":"PRINTABLE-STRING-PRIMITIVE-TYPE"
}
```

Where:

- `status:` An `OPERATION-RESPONSE-PRIMITIVE-TYPE` indicating the status of the `StoreTEEPropertyRequest` operation. If the TEE property is stored successfully, the value of `status` SHALL be `OPERATION_SUCCESS`; otherwise its value SHALL be an error string listed in section 5.21.1.
- `did:` The value of `did` from the `StoreTEEPropertyRequest`.
- `dsi:` The `DSI-CONTENT-TYPE` for the new device state. This element is returned only when the `nextdsi` is set to `true` in the `StoreTEEPropertyRequest`.
- `nextnonce:` A unique value that the OWE SHALL use as nonce in the next request. See section 2.10.4 for details.

5.21.1 Error Conditions

If any validation listed in section 5.20.1 fails or if a TEE error occurs, the rSD$_{TEE}$ SHALL use an appropriate `OPERATION-RESPONSE-PRIMITIVE-TYPE` (listed below) as the `status` value in the corresponding response message.

- `ERR_OCSP_INVALID`
- `ERR_OWE_NOT_TRUSTED`
- `ERR_REQUEST_INVALID`
- `ERR_TEE_BUSY`
- `ERR_TEE_FAIL`
- `ERR_TEE_RESOURCE_FULL`

- `ERR_TEE_UNKNOWN`
- `ERR_UNSUPPORTED_CRYPTO_ALG`
- `ERR_UNSUPPORTED_MSG_VERSION`

See section 4.14 for details on error strings.

5.22 FactoryResetTBSRequest

An OWE issues a `FactoryResetTBSRequest` message to move the TEE to a notional “factory” state. This message SHALL be signed using the JWS scheme and encapsulated in a `FactoryResetRequest` message. This message SHALL always be issued to an rSD_{TEE}.

The JSON structure for the `FactoryResetTBSRequest` is as follows:

```
{
        "FactoryResetTBSRequest":COMMAND-PARAMETER-TYPE
}
```

The following JSON element will be used as input to the `CONTENT-ENCRYPTION-TYPE` within `COMMAND-TYPE`.

```
{
        "tsmid":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "did":"PRINTABLE-STRING-PRIMITIVE-TYPE"
}
```

Where:

- `tsmid:` The identifier of the OWE that issued the request.
- `did:` The base64 encoded SHA-256 hash of the TEE-Cert binary. `did` is used as the device identifier to which the request is issued.

5.22.1 Processing Requirements

Upon receiving the `FactoryResetRequest` message, the rSD_{TEE} SHALL:

- Validate the JSON web signature associated with the request, using the OWE-Pub associated with the OWE-Cert.
- Determine whether the OWE-Cert chains to a root CA certificate in its OWE-Whitelist.
- Check the revocation status of the OWE-Cert chain, using the cached OCSP stapling. If the cache is unavailable or expired, the rSD SHALL return the corresponding response with an error string along with an indication (`signerreq` set to `true`) to provide OCSP stapling in the next request. The OWE may reissue the request with OCSP stapling.

Upon successfully completing the above steps, the rSD_{TEE} SHALL perform a factory reset on the device. All SDs and TAs created or installed using OTrP Profile or [TMF ASN.1] that are not listed in `gpd.tee.tmf.resetpreserved.entities` SHALL be removed. All TAs that are listed in `gpd.tee.tmf.resetpreserved.entities` SHALL be reset as follows:

- All active TEE Client or TEE Internal sessions are terminated. If the administration session used to perform the Factory Reset operation is terminated, then the factory reset SHALL continue.
- All data (if any) in the `TEE_STORAGE_PERSO` storage space is retained unmodified.
- All data (if any) in the `TEE_STORAGE_PRIVATE` storage space is removed atomically.

  WARNING: Future TEE specifications may add new storage IDs that are not mentioned in this document. Consult those specifications to determine how the new storage IDs react to factory reset.

A response message `StoreTEEPropertyResponse` SHALL always be returned regardless of the status of the operation.

5.23 FactoryResetTBSResponse

In response to a `FactoryResetRequest` command, the rSD$_{TEE}$ SHALL return a `FactoryResetResponse`, encapsulating the `FactoryResetTBSResponse` message. The JSON structure for the `FactoryResetTBSResponse` is as follows:

```
{
        "FactoryResetTBSResponse":RESPONSE-PARAMETER-TYPE
}
```

Within the `RESPONSE-PARAMETER-TYPE`, the following JSON structure is used as an input to the JWE while generating `CONTENT-ENCRYPTION-TYPE`.

```
{
        "status":"OPERATION-RESPONSE-PRIMITIVE-TYPE",
        "did":"PRINTABLE-STRING-PRIMITIVE-TYPE",
        "dsi":DSI-CONTENT-TYPE,
        "nextnonce":"PRINTABLE-STRING-PRIMITIVE-TYPE"
}
```

Where:

- `status:` An `OPERATION-RESPONSE-PRIMITIVE-TYPE` indicating the status of the `FactoryResetRequest` operation. If the TEE property is stored successfully, the value of `status` SHALL be `OPERATION_SUCCESS`; otherwise its value SHALL be an error string in section 5.23.1.
- `did:` The value of `did` from the `FactoryResetRequest`.
- `dsi:` The `DSI-CONTENT-TYPE` for the new device state. This element is returned only when the `nextdsi` is set to `true` in the `FactoryResetRequest`.
- `nextnonce:` A unique value that the OWE SHALL use as nonce in the next request. See section 2.10.4 for details.

5.23.1 Error Conditions

If any validation listed in section 5.22.1 fails or if a TEE error occurs, the rSD_{TEE} SHALL use an appropriate `OPERATION-RESPONSE-PRIMITIVE-TYPE` (listed below) as the `status` value in the corresponding response message.

- `ERR_OCSP_INVALID`
- `ERR_OWE_NOT_TRUSTED`
- `ERR_REQUEST_INVALID`
- `ERR_TEE_BUSY`
- `ERR_TEE_FAIL`
- `ERR_TEE_RESOURCE_FULL`
- `ERR_TEE_UNKNOWN`
- `ERR_UNSUPPORTED_CRYPTO_ALG`
- `ERR_UNSUPPORTED_MSG_VERSION`

See section 4.14 for details on error strings.

Annex A Changes

This annex describes changes between the original Open Trust Protocol (OTrP) v1.0 ([OTPA OTrP]) and the GlobalPlatform OTrP Profile described in this document.

A.1 Terminology

Table A-1: Changes to Terminology

Original Terminology	Terminology Used	Notes
Trusted Service Manager (TSM)	Outside World Entity (OWE)	OWE replaces TSM. OWEs are responsible for the life cycle management of TAs running on TEEs of devices.

A.2 JSON Elements

Table A-2: Changes to JSON Elements

Original JSON Element Name	JSON Element Name Used	Status	Notes
`sdname`	`sdid`	Updated	`sdname` represented the name of the Security Domain to be created. `sdname` has been changed to `sdid`, a UUID that identifies a Security Domain. Furthermore, `sdid` SHALL not be changeable.
`taname`		Removed	`taname` represented the TA application friendly name. A TA SHALL be represented only using `tid`, a UUID that identifies a TA.
`teespaik`	`spaik`	Updated	`teespaik` and `spaik` both represented SP-AIK-Pub. For consistency, only `spaik` is used.
`newsdname`		Removed	UUID of an SD SHALL not be changed.
`teespaiktype`		Removed	`spaik` is structured according to JWK, which includes the key type definition within the JWK structure.
`reason`		Removed	`reason` described the failure reason detail. This document incorporates reason for failure (`OPERATION-RESPONSE-PRIMITIVE-TYPE`) in the `status` element.
`cnt`		Removed	`cnt` represented the number of SDs owned by a TSM. The `cnt` element is redundant as the number of SDs owned by a TSM can be represented by an array of JSON object `SD-DEFINITION-TYPE`.
`encrypted_ta`	`encrypted_ta_bin`	Updated	`encrypted_ta` used an ad hoc JSON structure to represent encrypted TA binary, and included TA personalization data. The `encrypted_ta_bin` SHALL follow `CONTENT-ENCRYPTION-TYPE`, which is based on the JWE format for representing encrypted data. TA personalization data SHALL be represented using a separate JSON element, `encrypted_ta_data`.
n/a	`encrypted_ta_data`	Added	TA personalization data previously included in `encrypted_ta`.

Annex B String Identifiers for Curves in ECC

JWA defines string identifiers for NIST curves. GlobalPlatform uses a wider set of curves and so defines additional identifiers to cover those other cases.

Table B-1: String Identifiers for Curves in ECC

Curve Type	String Identifiers for Curves	[TEE Core] Algorithms	Notes
NIST Curves	P-224	TEE_ECC_CURVE_NIST_P224	
	P-256	TEE_ECC_CURVE_NIST_P256	
	P-384	TEE_ECC_CURVE_NIST_P384	
	P-521	TEE_ECC_CURVE_NIST_P521	
Brainpool Curves	BR-224	TEE_ECC_CURVE_BSI_P224r1	
	BR-256	TEE_ECC_CURVE_BSI_P256r1	
	BR-320	TEE_ECC_CURVE_BSI_P320r1	
	BR-384	TEE_ECC_CURVE_BSI_P384r1	
	BR-512	TEE_ECC_CURVE_BSI_P512r1	
Brainpool Twisted	BT-224	TEE_ECC_CURVE_BSI_P224t1	
	BT-256	TEE_ECC_CURVE_BSI_P256t1	
	BT-320	TEE_ECC_CURVE_BSI_P320t1	
	BT-384	TEE_ECC_CURVE_BSI_P384t1	
	BT-512	TEE_ECC_CURVE_BSI_P512t1	
Edwards Curves	Ed25519	TEE_ECC_CURVE_25519	Signature
	X25519		Key exchange
Chinese Curves	S-256	TEE_ECC_CURVE_SM2	

Annex C Specification Properties

Most properties used in this specification can be retrieved by the generic Property Access Functions described in [TEE Core], using the pseudo-handle specified in Table C-1.

The `gpd.sd.isRootSD` property of an SD is flagged internally by the TEE at SD installation time and SHOULD NOT be retrieved using the generic Property Access Functions.

Table C-1: Specification Reserved Properties

Property	Type	Comment	Can Be Retrieved by	Pseudo-handle Used in Retrieval
`gpd.client.parentSD`	UUID	The UUID of the direct parent SD of a TA. (See [TMF ASN.1] section 4.1.2.)	TA called by a client TA	`TEE_PROPSET_CURRENT_CLIENT`
`gpd.sd.isRootSD`	boolean	Property that is set internally by the TEE when successfully installing a new rSD.	n/a	n/a
`gpd.ta.parentSD`	UUID	The UUID of the direct parent SD of a TA. (See [TMF ASN.1] section 4.1.2.)	TA	`TEE_PROPSET_CURRENT_TA`
`gpd.tee.tmf.otrp.version`	uint32_t	The version of this specification, encoded as specified in [TMF ASN.1] section A.4.	TEE	`TEE_PROPSET_TEE_IMPLEMENTATION`

Annex D Verification of UUID Version 5

Verification of the UUID version 5 proof of possession is defined in [TMF ASN.1] section 8.3.3.7 and section 5.6.1. The proof of possession is a signature calculated over the sequence of bytes resulting from the concatenation of the tag-length-value octets of the TA UUID and the TA Binary File.

Table D-1: Message to be Signed

Tag	Length	Value (in hex)	Description
0x43	0x10	ab cd ef 01 23 45 67 89 ab cd ef 01 23 45 67 89	TA UUID "abcdef01-2345-6789-abcd-ef0123456789"
0x04	<L>	54 41 20 62 69 6e 61 72 79	The plain text of the TA binary: Here substituted with the dummy value "TA binary"

Where:

L: The length of the TA binary encoded as defined in [TMF ASN.1] section 7.6.

The proof of possession then has the following form:

Table D-2: Example of Proof of Possession Encoding Values for InstallTARequest, UpdateTARequest

Tag	Length	Value (in hex)	Description
0x68	0x6a		uuidVerificationParams structure of length 106 octets
0x43	0x10	6b c2 de 43 50 12 48 55 9c 8e ea af 0c b9 fd e7	Protocol (UUID version 5 verification)
0x02	0x01	01	Version of protocol
0xa0	0x53		uuidV5Params structure of length 83 octets
0x02	0x05	00 A0 00 00 30	Key type: TEE_TYPE_RSA_PUBLIC_KEY
0x02	0x02	08 00	Key size: 2048
0x30	0x25		SEQUENCE of attributes structure of length 37 octets
0x62	0x10		Attribute structure of length 16 octets
0x02	0x05	00 D0 00 01 30	Attribute id: TEE_ATTR_RSA_MODULUS
0x04	0x07	6d 6f 64 75 6c 75 73	Modulus attribute: Here substituted with the dummy value "modulus"
0x62	0x11		Attribute structure of length 17 octets
0x02	0x05	00 D0 00 02 30	Attribute id: TEE_ATTR_RSA_PUBLIC_EXPONENT
0x04	0x08	65 78 70 6f 6e 65 6e 74	Exponent attribute: Here substituted with the dummy value "exponent"
0x65	0x09		signatureParams structure of length 9 octets
0x02	0x04	70 41 49 30	Algorithm identifier: TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA256
0x02	0x01	03	Operation mode: TEE_MODE_VERIFY
0x04	0x14	73 6f 6d 65 20 73 69 67 6e 61 74 75 72 65 20 76 61 6c 75 65	Signature: Here substituted with the dummy value "some signature value"

WARNING: Please check that a new release of [TMF ASN.1] has not changed the underlying definition.