

# GlobalPlatform Technology

## TEE Management Framework: Open Trust Protocol (OTrP) Profile

### Version 1.0.0.6

---

**Public Review**

**March 2020**

**Document Reference: GPD\_SPE\_123**

**Copyright © 2017-2020 GlobalPlatform, Inc. All Rights Reserved.**

*Recipients of this document are invited to submit, with their comments, notification of any relevant patents or other intellectual property rights (collectively, "IPR") of which they may be aware which might be necessarily infringed by the implementation of the specification or other work product set forth in this document, and to provide supporting documentation. This document is currently in draft form, and the technology provided or described herein may be subject to updates, revisions, extensions, review, and enhancement by GlobalPlatform or its Committees or Working Groups. Prior to publication of this document by GlobalPlatform, neither Members nor third parties have any right to use this document for anything other than review and study purposes. Use of this information is governed by the GlobalPlatform license agreement and any use inconsistent with that agreement is strictly prohibited.*

THIS SPECIFICATION OR OTHER WORK PRODUCT IS BEING OFFERED WITHOUT ANY WARRANTY WHATSOEVER, AND IN PARTICULAR, ANY WARRANTY OF NON-INFRINGEMENT IS EXPRESSLY DISCLAIMED. ANY IMPLEMENTATION OF THIS SPECIFICATION OR OTHER WORK PRODUCT SHALL BE MADE ENTIRELY AT THE IMPLEMENTER'S OWN RISK, AND NEITHER THE COMPANY, NOR ANY OF ITS MEMBERS OR SUBMITTERS, SHALL HAVE ANY LIABILITY WHATSOEVER TO ANY IMPLEMENTER OR THIRD PARTY FOR ANY DAMAGES OF ANY NATURE WHATSOEVER DIRECTLY OR INDIRECTLY ARISING FROM THE IMPLEMENTATION OF THIS SPECIFICATION OR OTHER WORK PRODUCT.

# Contents

<b>1</b>	<b>Introduction .....</b>	<b>6</b>
1.1	Audience .....	6
1.2	IPR Disclaimer .....	6
1.3	References .....	7
1.4	Terminology and Definitions .....	8
1.5	Abbreviations and Notations .....	10
1.6	Revision History .....	11
<b>2</b>	<b>OTrP Overview.....</b>	<b>12</b>
2.1	Architecture .....	12
2.2	Nomenclature .....	13
2.3	Root Security Domain (rSD) .....	14
2.4	Security Domain (SD) .....	15
2.4.1	UUID Calculation.....	15
2.5	Outside World Entity (OWE) .....	16
2.6	Service Provider (SP).....	16
2.7	Trusted Firmware (TFW).....	16
2.8	OTrP Agent .....	16
2.9	OWE Certificate (OWE-Cert) .....	16
2.10	Security Model .....	17
2.10.1	Security Mechanism.....	17
2.10.2	Cryptographic Requirements .....	17
2.10.3	Cryptographic Recommendations.....	18
2.10.4	Nonce .....	18
2.10.5	Device State Information (DSI) .....	18
2.10.6	Use of Keys .....	18
2.10.7	Key Update .....	19
<b>3</b>	<b>Encoding OTrP Messages Using TEE Client API.....</b>	<b>20</b>
3.1	Reserved Command IDs.....	20
3.2	Encoding OTrP Messages .....	21
3.2.1	Handling Variable Length Return Values.....	21
3.2.2	Atomicity of Operations .....	21
3.2.3	Returning OTrP Errors .....	21
<b>4</b>	<b>JSON Message Formatting.....</b>	<b>22</b>
4.1	COMMAND-TYPE.....	22
4.2	UNPRIVILEGED-COMMAND-TYPE .....	23
4.3	COMMAND-PAYLOAD .....	23
4.4	PROTECTED-HEADER-TYPE .....	24
4.5	HEADER-TYPE.....	24
4.6	COMMAND-PARAMETER-TYPE .....	25
4.7	RESPONSE-PARAMETER-TYPE .....	26
4.8	CONTENT-ENCRYPTION-TYPE .....	27
4.9	KEYWRAP-INFO-TYPE .....	28
4.10	ENCRYPTION-PRIMITIVE-TYPE.....	29
4.11	SIGNATURE-PRIMITIVE-TYPE .....	30
4.12	KEYWRAP-PRIMITIVE-TYPE .....	31
4.13	CERT-PRIMITIVE-TYPE.....	31
4.14	OPERATION-RESPONSE-PRIMITIVE-TYPE.....	32
4.15	DSI-TYPE.....	34

4.16	DSI-CONTENT-TYPE .....	34
4.17	TRUSTED-FIRMWARE-TYPE .....	35
4.18	TEE-DESCRIPTION-TYPE .....	36
4.19	SD-DEFINITION-TYPE .....	38
4.20	TA-DEFINITION-TYPE .....	38
4.21	ISA-TYPE .....	39
4.22	TEE-PROPERTY-TYPE .....	39
4.23	TEE-AIK-TYPE .....	40
4.24	PUB-KEY-TYPE .....	41
4.24.1	RSA Key .....	41
4.24.2	ECC Key .....	41
4.25	OCSP-ARRAY-TYPE .....	42
4.26	UUID-ARRAY-TYPE .....	42
4.27	GPD-VERSION-TYPE .....	42
4.28	POP-TYPE .....	43
4.29	PUB-KEY-ROLE-TYPE .....	43
4.30	PUB-KEY-ROLE-ARRAY-TYPE .....	43
4.31	Version Negotiation .....	44
<b>5</b>	<b>OTrP Messages .....</b>	<b>45</b>
5.1	Unprivileged Messages .....	45
5.2	Privileged Messages .....	46
5.2.1	Creating a Privileged Message .....	46
5.3	TEE Management Messages .....	47
5.4	GetTAInformationRequest .....	48
5.5	GetTAInformationResponse .....	49
5.6	GetDeviceTEESStateTBSRequest .....	51
5.7	GetDeviceTEESStateTBSResponse .....	53
5.8	CreateSDTBSRequest .....	55
5.9	CreateSDTBSResponse .....	57
5.10	UpdateSDTBSRequest .....	59
5.11	UpdateSDTBSResponse .....	62
5.12	DeleteSDTBSRequest .....	64
5.13	DeleteSDTBSResponse .....	66
5.14	InstallTATBSRequest .....	68
5.15	InstallTATBSResponse .....	71
5.16	UpdateTATBSRequest .....	73
5.17	UpdateTATBSResponse .....	75
5.18	DeleteTATBSRequest .....	77
5.19	DeleteTATBSResponse .....	79
5.20	StoreTEEPPropertyTBSRequest .....	81
5.21	StoreTEEPPropertyTBSResponse .....	83
5.22	FactoryResetTBSRequest .....	85
5.23	FactoryResetTBSResponse .....	87
<b>Annex A</b>	<b>Changes .....</b>	<b>89</b>
A.1	Terminology .....	89
A.2	JSON Elements .....	90
<b>Annex B</b>	<b>String Identifiers for Curves in ECC .....</b>	<b>91</b>
<b>Annex C</b>	<b>Specification Properties .....</b>	<b>92</b>
<b>Annex D</b>	<b>Verification of UUID Version 5 .....</b>	<b>93</b>

## Tables

Table 1-1: Normative References.....	7
Table 1-2: Informative References .....	7
Table 1-3: Terminology and Definitions.....	8
Table 1-4: Abbreviations and Notations .....	10
Table 1-5: Revision History .....	11
Table 2-1: Document-specific Terminology and Definitions .....	13
Table 3-1: Reserved Command IDs.....	20
Table 3-2: Envelope Command Encoding.....	21
Table 4-1: Examples of base64url encoded ENCRYPTION-PRIMITIVE-TYPE .....	29
Table 4-2: Example [TEE Core] Algorithms to Support ENCRYPTION-PRIMITIVE-TYPE .....	29
Table 4-3: Examples of base64url encoded SIGNATURE-PRIMITIVE-TYPE.....	30
Table 4-4: Example [TEE Core] Algorithms to Support SIGNATURE-PRIMITIVE-TYPE.....	30
Table 4-5: Example [TEE Core] Algorithms to Support KEYWRAP-PRIMITIVE-TYPE.....	31
Table 4-6: Internal API Names Strings Definition.....	37
Table 5-1: Request/Response String Naming.....	45
Table A-1: Changes to Terminology.....	89
Table A-2: Changes to JSON Elements.....	90
Table B-1: String Identifiers for Curves in ECC.....	91
Table C-1: Specification Reserved Properties .....	92
Table D-1: Message to be Signed.....	93
Table D-2: Example of Proof of Possession Encoding Values for InstallTARrequest, UpdateTARrequest.....	94

## Figures

Figure 2-1: OTrP Architecture .....	12
Figure 3-1: Single Envelope Command .....	21

# 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](mailto: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.

29 **1.3 References**

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

32 **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 <a href="https://www.ietf.org/rfc/rfc3447">https://www.ietf.org/rfc/rfc3447</a>	[RFC 3447]
RFC 4122	Version 1 UUID <a href="https://tools.ietf.org/html/rfc4122">https://tools.ietf.org/html/rfc4122</a>	[RFC 4122]
RFC 5280	Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile <a href="https://tools.ietf.org/html/rfc5280">https://tools.ietf.org/html/rfc5280</a>	[RFC 5280]
RFC 7515	JSON Web Signature (JWS) <a href="https://tools.ietf.org/html/rfc7515">https://tools.ietf.org/html/rfc7515</a>	[RFC 7515]
RFC 7516	JSON Web Encryption (JWE) <a href="https://tools.ietf.org/html/rfc7516">https://tools.ietf.org/html/rfc7516</a>	[RFC 7516]
RFC 7517	JSON Web Key (JWK) <a href="https://tools.ietf.org/html/rfc7517">https://tools.ietf.org/html/rfc7517</a>	[RFC 7517]
RFC 7518	JSON Web Algorithms (JWA) <a href="https://tools.ietf.org/html/rfc7518">https://tools.ietf.org/html/rfc7518</a>	[RFC 7518]

33 **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]

## 34 1.4 Terminology and Definitions

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

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

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

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

44 **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 <i>Trusted Application (TA)</i> .
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)	<p>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).</p> <p>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.</p> <p>(Formerly referred to as a <i>Rich Execution Environment (REE)</i>.)                      Contrast <i>Trusted Execution Environment (TEE)</i>.</p>
Regular OS	<p>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.</p> <p>(Formerly referred to as a <i>Rich OS</i> or <i>Device OS</i>.)                      Contrast <i>Trusted OS</i>.</p>
Root Security Domain (rSD)	<p>A Security Domain over which other Authorities have very limited control; described in detail in [TMF ASN.1] section 4.1.3.3.</p>
Secure Component	<p>GlobalPlatform terminology to represent either a Secure Element or a Trusted Execution Environment.</p>
Security Domain (SD)	<p>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.</p>
Service Provider (SP)	<p>An entity that issues TAs. (For more information, see section 2.6.)</p>
Session	<p>Logically connects multiple commands invoked on a Trusted Application or a Security Domain.</p> <p>In the context of this specification, logically connects an OWE to a TEE on a device. Begins with a <code>GetDeviceTEESStateRequest</code>.</p>
Trusted Application (TA)	<p>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.</p> <p>Contrast <i>Client Application</i>.</p>
Trusted Execution Environment (TEE)	<p>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.</p> <p>Contrast <i>Regular Execution Environment (REE)</i>.</p>
Trusted OS	<p>An OS executing in a Secure Component.</p> <p>Contrast <i>Regular OS</i>.</p>

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.

## 45 1.5 Abbreviations and Notations

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

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

49 **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

50 **1.6 Revision History**

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

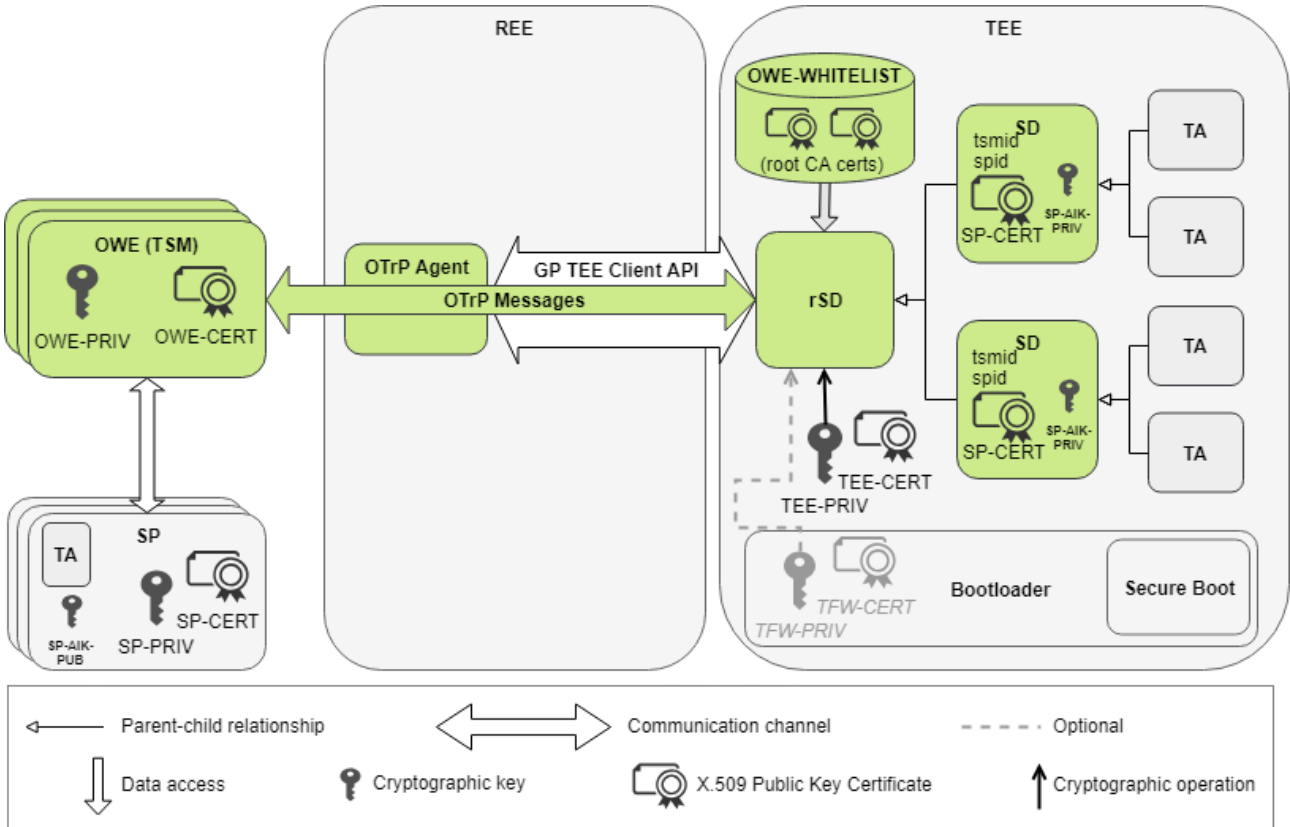
55 **Table 1-5: Revision History**

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

56 **2 OTrP Overview**

57 **2.1 Architecture**

58 **Figure 2-1: OTrP Architecture**



60  
61

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

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

77 **2.2 Nomenclature**

78 **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 <sub>TA</sub>	An rSD with TA and SD management privileges.
rSD <sub>TEE</sub>	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.

## 79 2.3 Root Security Domain (rSD)

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

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

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

## 95 2.4 Security Domain (SD)

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

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

### 106 2.4.1 UUID Calculation

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

- 109 • Set octets 0 through 3 of the *time\_low* field to octets 0 through 3 of the hash.
- 110 • Set octets 0 and 1 of the *time\_mid* field to octets 4 and 5 of the hash.
- 111 • Set octets 0 and 1 of the *time\_hi\_and\_version* field to octets 6 and 7 of the hash.
- 112 • Set the *clock\_seq\_hi\_and\_reserved* field to octet 8 of the hash.
- 113 • Set the two most significant bits (bits 6 and 7) of the *clock\_seq\_hi\_and\_reserved* field to 01.
- 114 • Set the *clock\_seq\_low* field to octet 9 of the hash.
- 115 • Set octets 0 through 5 of the *node* field to octets 10 through 15 of the hash.

116 To complete a version 1 UUID:

- 117 • Set the four most significant bits (bits 12 through 15) of the *time\_hi\_and\_version* field to 0001.

118 To complete a version 4 UUID:

- 119 • Set the four most significant bits (bits 12 through 15) of the *time\_hi\_and\_version* field to 0100.

## 120 2.5 Outside World Entity (OWE)

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

## 125 2.6 Service Provider (SP)

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

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

## 131 2.7 Trusted Firmware (TFW)

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

## 137 2.8 OTrP Agent

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

## 143 2.9 OWE Certificate (OWE-Cert)

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

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

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



## 154 2.10 Security Model

155 The goals of the security model for OTrP Profile are:

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

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

### 165 2.10.1 Security Mechanism

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

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

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

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

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

178 OTrP Profile uses tsmid to enforce the access control policies on SDs and TAs. The rSD<sub>TA</sub> associates each  
179 SD with the tsmid of the OWE that requested the creation of the SD. The rSD<sub>TA</sub> validates the tsmid present on  
180 the OWE-Cert before authorizing operations on the SD.

### 181 2.10.2 Cryptographic Requirements

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

### 186 2.10.3 Cryptographic Recommendations

187 OTrP Profile SHOULD use the following cryptographic recommendations:

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

### 196 2.10.4 Nonce

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

### 207 2.10.5 Device State Information (DSI)

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

### 214 2.10.6 Use of Keys

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

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

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

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

225 **2.10.7 Key Update**

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

## 228 3 Encoding OTrP Messages Using TEE Client API

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

- 231 • OTrP Agent opens an administrative session to the relevant OTrP root Security Domain.
- 232 • Using this session, the OTrP Agent forwards the OTrP requests using [TEE Client]  
233 TEEC\_InvokeCommand.

### 234 3.1 Reserved Command IDs

235 When TEEC\_InvokeCommand is called to send OTrP messages to a Security Domain, the [TEE Client]  
236 Command IDs defined in Table 3-1 are reserved.

237 **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

238 **3.2 Encoding OTrP Messages**

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

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

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

248 **Figure 3-1: Single Envelope Command**



250 **Table 3-2: Envelope Command Encoding**

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

251 **3.2.1 Handling Variable Length Return Values**

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

253 **3.2.2 Atomicity of Operations**

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

258 **3.2.3 Returning OTrP Errors**

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

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

## 265 4 JSON Message Formatting

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

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

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

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

### 278 4.1 COMMAND-TYPE

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

```
281 {  
282     "payload" : COMMAND-PAYLOAD ,  
283     "protected" : PROTECTED-HEADER-TYPE ,  
284     "header" : HEADER-TYPE ,  
285     "signature" : "PRINTABLE-STRING-PRIMITIVE-TYPE"  
286 }
```

287 Where:

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

## 293 4.2 UNPRIVILEGED-COMMAND-TYPE

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

295 GET-TA-INFORMATION-REQUEST

296 GET-TA-INFORMATION-RESPONSE

## 297 4.3 COMMAND-PAYLOAD

298 COMMAND-PAYLOAD SHALL be the base64url encoding of:

299 COMMAND-TBS

300 Where:

301 • COMMAND-TBS: One of the following OTrP message types:

302 GET-DEVICE-TEE-STATE-TBS-REQUEST

303 GET-DEVICE-TEE-STATE-TBS-RESPONSE

304 CREATE-SD-TBS-REQUEST

305 CREATE-SD-TBS-RESPONSE

306 UPDATE-SD-TBS-REQUEST

307 UPDATE-SD-TBS-RESPONSE

308 DELETE-SD-TBS-REQUEST

309 DELETE-SD-TBS-RESPONSE

310 INSTALL-TA-TBS-REQUEST

311 INSTALL-TA-TBS-RESPONSE

312 UPDATE-TA-TBS-REQUEST

313 UPDATE-TA-TBS-RESPONSE

314 DELETE-TA-TBS-REQUEST

315 DELETE-TA-TBS-RESPONSE

316 STORE-TEE-PROPERTY-TBS-REQUEST

317 STORE-TEE-PROPERTY-TBS-RESPONSE

318 FACTORY-RESET-TBS-REQUEST

319 FACTORY-RESET-TBS-RESPONSE

## 320 4.4 PROTECTED-HEADER-TYPE

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

```
323 {
324     "alg" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
325     "rSD" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
326     "tee" : "PRINTABLE-STRING-PRIMITIVE-TYPE"
327 }
```

328 Where:

- 329 • alg: A cryptographic algorithm used to sign a message. Its value SHALL be one of the "alg"  
330 values defined in [RFC 7518]. However, if the selected algorithm in the OTrP request is not supported  
331 by [TEE Core] or is not acceptable by the OTrP Profile, then the rSD SHALL return the response with  
332 an error message. For more details on alg, see section 4.11, SIGNATURE-PRIMITIVE-TYPE.
- 333 • rSD: (OPTIONAL) The UUID of the rSD that is supposed to receive the request message. When this  
334 element is not supplied, the OTrP request SHALL be sent to the default rSD<sub>TA</sub> on the device.
- 335 • tee: (OPTIONAL) A zero-terminated string that describes the TEE to connect to. Its value matches  
336 the parameter name used to connect to a TEE while initializing a context using the  
337 TEEC\_InitializeContext. See [TEE Client] section 4.5.2 for details. When this element is not  
338 supplied, the OTrP request SHALL be sent to the default TEE on the device.

## 339 4.5 HEADER-TYPE

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

```
341 {
342     "x5c" : [ "CERT-PRIMITIVE-TYPE" ],
343     "kid" : "PRINTABLE-STRING-PRIMITIVE-TYPE"
344 }
```

345 Where:

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

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



## 352 4.6 COMMAND-PARAMETER-TYPE

353 OTrP request messages SHALL have the following common elements:

```
354 {  
355     "ver" : "GPD-VERSION-TYPE" ,  
356     "tid" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,  
357     "rid" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,  
358     "tee" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,  
359     "nextdsi" : BOOLEAN ,  
360     "dsihash" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,  
361     "nonce" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,  
362     "content" : CONTENT-ENCRYPTION-TYPE  
363 }
```

364 Where:

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

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

## 381 4.7 RESPONSE-PARAMETER-TYPE

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

```
383 {  
384     "ver": "GPD-VERSION-TYPE" ,  
385     "rid": "PRINTABLE-STRING-PRIMITIVE-TYPE" ,  
386     "tid": "PRINTABLE-STRING-PRIMITIVE-TYPE" ,  
387     "content": CONTENT-ENCRYPTION-TYPE  
388 }
```

389 Where:

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

## 395 4.8 CONTENT-ENCRYPTION-TYPE

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

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

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

```
405 {  
406     "protected": "ENCRYPTION-PRIMITIVE-TYPE",  
407     "recipients": [KEYWRAP-INFO-TYPE],  
408     "iv": "PRINTABLE-STRING-PRIMITIVE-TYPE",  
409     "ciphertext": "PRINTABLE-STRING-PRIMITIVE-TYPE",  
410     "tag": "PRINTABLE-STRING-PRIMITIVE-TYPE"  
411 }
```

412 Where:

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

## 421 4.9 KEYWRAP-INFO-TYPE

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

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

```
425 {
426     "header": {
427         "alg": "KEYWRAP-PRIMITIVE-TYPE",
428         "kid": "PRINTABLE-STRING-PRIMITIVE-TYPE"
429     },
430     "encrypted_key": "PRINTABLE-STRING-PRIMITIVE-TYPE"
431 }
```

432 Where:

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

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

```
440 {
441     "header": {
442         "alg": "KEYWRAP-PRIMITIVE-TYPE",
443         "kid": "PRINTABLE-STRING-PRIMITIVE-TYPE"
444     },
445     "encrypted_key": "PRINTABLE-STRING-PRIMITIVE-TYPE",
446     "epk": PUB-KEY-TYPE,
447     "apu": "PRINTABLE-STRING-PRIMITIVE-TYPE",
448     "apv": "PRINTABLE-STRING-PRIMITIVE-TYPE"
449 }
```

450 Where header, alg, kid, and encrypted\_key are as defined above, and:

- 451 • epk: An ephemeral ECC public key structured as the ECC-based PUB-KEY-TYPE.
- 452 • apu: The base64url encoded agreement PartyUIInfo value for key agreement algorithm.
- 453 • apv: The base64url encoded agreement PartyVIInfo value for key agreement algorithm.

454 **4.10 ENCRYPTION-PRIMITIVE-TYPE**

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

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

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

463 Where:

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

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

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

ENCRYPTION-PRIMITIVE-TYPE	base64url encoded
{"enc": "A128CBC-HS256"}	eyJlbnMiOiJBMTI4Q0JDLUhTMjU2In0g
{"enc": "A256CBC-HS512"}	eyJlbnMiOiJBMTU2Q0JDLUhTNTUyIn0g

470  
 471 Table 4-2 lists the corresponding algorithms in [TEE Core] to support the above example encryption  
 472 algorithms. (These correspondences were true when this specification was published. Confirm the latest  
 473 information in [TEE Core] and [RFC 7518].)

474 **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

475  
 476 A128CBC-HS256 and A256CBC-HS512 use PKCS#7 padding. The padding mechanism should be  
 477 implemented separately as [TEE Core] does not support it.

478 **Note:** See section 2.10 for additional information.

## 479 4.11 SIGNATURE-PRIMITIVE-TYPE

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

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

```
486 { "alg": "RS256" }
```

```
487 { "alg": "ES256" }
```

488 Where:

- 489 • {"alg": "RS256"}: Represents signature generated with RSASSA-PKCS1-v1\_5 ([RFC 3447])  
 490 using SHA-256.
- 491 • {"alg": "ES256"}: Represents signature generated with ECDSA using P-256 curve and SHA-256.

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

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

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

494

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

498 **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

499

500 **Note:** See section 2.10 for additional information.

501 **4.12 KEYWRAP-PRIMITIVE-TYPE**

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

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

```
508 RSA1_5
509 ECDH-ES+A128KW
510 ECDH-ES+A256KW
```

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

514 **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

515  
 516 **Note:** See section 2.10 for additional information.

517 **4.13 CERT-PRIMITIVE-TYPE**

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

## 519 4.14 OPERATION-RESPONSE-PRIMITIVE-TYPE

520 One of the following strings:

```
521 OPERATION_SUCCESS
522 ERR_DEV_STATE_MISMATCH
523 ERR_INVALID_UUID
524 ERR_OCSP_INVALID
525 ERR_OWE_NOT_TRUSTED
526 ERR_REQUEST_INVALID
527 ERR_SD_NOT_EMPTY
528 ERR_SDID_ALREADY_USED
529 ERR_SPCERT_INVALID
530 ERR_TA_ALREADY_INSTALLED
531 ERR_TA_INVALID
532 ERR_TA_NOT_FOUND
533 ERR_TEE_BUSY
534 ERR_TEE_FAIL
535 ERR_TEE_RESOURCE_FULL
536 ERR_TEE_UNKNOWN
537 ERR_TFW_NOT_TRUSTED
538 ERR_UNSUPPORTED_CRYPTO_ALG
539 ERR_UNSUPPORTED_MSG_VERSION
540 ERR_UPDATING_DATA
```

541 Where the values have the following meanings:

- 542 • OPERATION\_SUCCESS: Returned when the corresponding request message has been processed  
543 successfully.
- 544 • ERR\_DEV\_STATE\_MISMATCH: Returned when the DSI hash value from OWE doesn't match that of  
545 the device's current DSI.
- 546 • ERR\_INVALID\_UUID: Returned when the given UUID is not supported or cannot be verified.
- 547 • ERR\_OCSP\_INVALID: Returned when the OCSP stapling is either invalid, not available, or expired.
- 548 • ERR\_OWE\_NOT\_TRUSTED: Returned when the OWE-Cert chain cannot be validated using the root CA  
549 certificate in the OWE-Whitelist while processing a request message.
- 550 • ERR\_REQUEST\_INVALID: Returned when any of the following conditions occurs:
  - 551 ○ Request message is not supported by the rSD.
  - 552 ○ Request message has an invalid message structure; e.g. mandatory element is absent, or  
553 undefined elements or structures are included.



- 554       ○ Failure to verify message signature.
- 555       ○ Failure to decrypt `CONTENT-ENCRYPTION-TYPE` value.
- 556       ○ Insufficient privilege to perform an operation (e.g. deletion of a TA from an SD that the OWE is not  
557       allowed to access).
- 558       • `ERR_SD_NOT_EMPTY`: Returned when an OWE tries to delete an SD that contains one or more TAs.
- 559       • `ERR_SDID_ALREADY_USED`: Returned when an OWE requests creation of an SD with a UUID that  
560       already exists in the namespace of the OWE in the TEE.
- 561       • `ERR_SPCERT_INVALID`: Returned when the new SP-Cert provided while updating an SD is not valid.
- 562       • `ERR_TA_ALREADY_INSTALLED`: Returned when an OWE requests installation of a TA with a given  
563       UUID and a version that already exists.
- 564       • `ERR_TA_INVALID`: Returned when any of the following conditions occurs while checking validity of a  
565       TA:
  - 566       ○ TA binary has a format that the TEE doesn't recognize.
  - 567       ○ TEE fails to decrypt the encoding of TA binary and personalization data.
  - 568       ○ If the SP isn't registered with the SD where a TA is to be installed.
  - 569       ○ During an update, if the version of the TA is lower than the current version installed.
  - 570       ○ If the TA version information provided in the request message is different than the TA version  
571       associated with the TA binary.
- 572       • `ERR_TA_NOT_FOUND`: Returned when the target TA doesn't exist in the SD.
- 573       • `ERR_TEE_BUSY`: Returned when the device TEE is currently busy.
- 574       • `ERR_TEE_FAIL`: Returned when any of the following conditions occurs:
  - 575       ○ TEE fails to respond to an OWE request. The OTrP Agent will construct an error message in  
576       responding to the OWE's request.
  - 577       ○ TEE fails to process a request because of its internal error.
- 578       • `ERR_TEE_RESOURCE_FULL`: Returned when a device resource is no longer available, such as  
579       storage space is full.
- 580       • `ERR_TEE_UNKNOWN`: Returned when the TEE is not supposed to receive the request, as determined  
581       by checking the TEE name or device identifier (`d.id`) in the request message.
- 582       • `ERR_TFW_NOT_TRUSTED`: Returned when the TEE determines that the underlying device firmware is  
583       not trustworthy.
- 584       • `ERR_UNSUPPORTED_CRYPTO_ALG`: Returned when a request message contains `CONTENT-`  
585       `ENCRYPTION-TYPE` value encrypted with a cryptographic algorithm that the TEE doesn't support.
- 586       • `ERR_UNSUPPORTED_MSG_VERSION`: Returned when the OTrP version of the request message is not  
587       supported by the TEE.
- 588       • `ERR_UPDATING_DATA`: Returned when updating a data parameter (`sd_data` or  
589       `encrypted_ta_data`) during an `UpdateSDRequest` or an `UpdateTARequest` is unsuccessful.

## 590 4.15 DSI-TYPE

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

```
593 {  
594     "dsi" : DSI-CONTENT-TYPE  
595 }
```

596 Where:

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

## 598 4.16 DSI-CONTENT-TYPE

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

```
600 {  
601     "tfwdata" : TRUSTED-FIRMWARE-TYPE ,  
602     "tee" : TEE-DESCRIPTION-TYPE  
603 }
```

604 Where:

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

## 607 4.17 TRUSTED-FIRMWARE-TYPE

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

```
610 {  
611     "payload": "PRINTABLE-STRING-PRIMITIVE-TYPE",  
612     "protected": PROTECTED-HEADER-TYPE,  
613     "header": HEADER-TYPE,  
614     "signature": "PRINTABLE-STRING-PRIMITIVE-TYPE"  
615 }
```

616 Where:

- 617 • **payload**: The string representing a challenge that the TFW SHALL sign. The `tid` value from the  
618 corresponding `GetDeviceTEEStateRequest` is used as the challenge.
  - 619 • **protected**: The JWS protected header element structured as `PROTECTED-HEADER-TYPE`. The  
620 `PROTECTED-HEADER-TYPE` SHALL include only the `"alg"` element that indicates the cryptographic  
621 algorithm used to sign the payload.
  - 622 • **header**: The JWS header element structured as `HEADER-TYPE`. The `x5c` element of the  
623 `HEADER-TYPE` SHALL contain the TFW-Cert represented as `CERT-PRIMITIVE-TYPE`. Storage of  
624 TFW-Cert is implementation defined.
- 625 **Note:** Version 1.0 of this specification incorrectly suggested that  
626 `gpd.tee.firmware.implementation.binaryversion` could be used to extract TFW-Cert from  
627 the TFW.
- 628 • **signature**: The base64url encoded signature calculated according to the JWS scheme.

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

## 631 4.18 TEE-DESCRIPTION-TYPE

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

```

633 {
634     "name" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
635     "teever" : "GPD-VERSION-TYPE" ,
636     "cert" : "CERT-PRIMITIVE-TYPE" ,
637     "cacert" : [ "CERT-PRIMITIVE-TYPE" ] ,
638     "sdlist" : [ SD-DEFINITION-TYPE ] ,
639     "teeaiklist" : [ TEE-AIK-TYPE ] ,
640     "isaset" : ISA-TYPE ,
641     "teeImplementationProperty" : [ TEE-PROPERTY-TYPE ]
642 }

```

643 Where:

- 644 • name: A zero-terminated string that describes the TEE to connect to. Its value matches the  
645 parameter `name` used to connect to a TEE while initializing a context using the  
646 `TEEC_InitializeContext`. For details, see [TEE Client] section 4.5.2.
- 647 • teever: The TEE version `gpd.tee.trustedos.implementation.version` structured as  
648 `GPD-VERSION-TYPE`.
- 649 • cert: The TEE-Cert represented as `CERT-PRIMITIVE-TYPE`. The certified key must be unique to  
650 the TEE instance.  
651 **Note:** Storage of the TEE-Cert and associated key is out of scope.  
652 **Note:** Version 1.0 of this specification incorrectly suggested that  
653 `gpd.tee.trustedos.implementation.binaryversion` could be used to extract TEE-Cert from  
654 the TFW.
- 655 • cacert: The X.509 certificate chain starting with the CA certificate that issued the TEE-Cert up to  
656 the root CA certificate structured as the `CERT-PRIMITIVE-TYPE` array.
- 657 • sdlist: An array of `SD-DEFINITION-TYPE`, where each element of the array provides the  
658 metadata of an SD that a given OWE has access to. This element SHALL be excluded if the rSD that  
659 prepares this JSON object is an `rSDTEE`.
- 660 • teeaiklist: An array of `TEE-AIK-TYPE`, where each element of the array provides information  
661 related to an SP-AIK. This element SHALL be excluded if the rSD that prepares this JSON object is an  
662 `rSDTEE`.
- 663 • isaset: (OPTIONAL) Instruction set and architecture definition based on ISA Type defined in  
664 [TMF ASN.1] section 9.1.4.
- 665 • teeImplementationProperty: (OPTIONAL) Lists the TEE properties. For more information on  
666 TEE properties, see [TMF ASN.1] section A.5. This element SHALL be included only if the rSD that  
667 prepares this JSON structure is an `rSDTEE`. However, if the TEE has TMF ASN.1 audit SD capabilities,  
668 then OTrP SHALL provide the following valid API name string to be used with the `optionalApis`  
669 attribute of TEE Type, defined in [TMF ASN.1] section 9.1.6.

670

**Table 4-6: Internal API Names Strings Definition**

<b>Strings</b>	<b>Description</b>
TMF-OTrP-Profile	OTrP Profile of TEE Management Framework

## 671 4.19 SD-DEFINITION-TYPE

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

```
673 {  
674     "sdid" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,  
675     "spid" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,  
676     "protocol" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,  
677     "talist" : [TA-DEFINITION-TYPE ]  
678 }
```

679 Where:

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

## 688 4.20 TA-DEFINITION-TYPE

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

```
690 {  
691     "taid" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,  
692     "taver" : "PRINTABLE-STRING-PRIMITIVE-TYPE "  
693 }
```

694 Where:

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

## 698 4.21 ISA-TYPE

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

```
701 {  
702     "isaName": "PRINTABLE-STRING-PRIMITIVE-TYPE",  
703     "processorType": "PRINTABLE-STRING-PRIMITIVE-TYPE",  
704     "instructionSet": "PRINTABLE-STRING-PRIMITIVE-TYPE",  
705     "addressSize": INTEGER-PRIMITIVE-TYPE,  
706     "abi": "PRINTABLE-STRING-PRIMITIVE-TYPE",  
707     "endianness": INTEGER-PRIMITIVE-TYPE  
708 }
```

709 Where:

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

## 716 4.22 TEE-PROPERTY-TYPE

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

```
719 {  
720     "PROPERTY-NAME": PROPERTY-VALUE  
721 }
```

722 Where:

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

## 725 4.23 TEE-AIK-TYPE

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

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

### 729 Version 1.1

```
730 {  
731     "spaik" : [PUB-KEY-ROLE-TYPE] ,  
732     "spid" : "PRINTABLE-STRING-PRIMITIVE-TYPE"  
733 }  
734
```

735 Where:

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

### 738 Version 1.0

```
739 {  
740     "spaik" : PUB-KEY-TYPE ,  
741     "spid" : "PRINTABLE-STRING-PRIMITIVE-TYPE"  
742 }  
743
```

744 Where:

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



## 747 4.24 PUB-KEY-TYPE

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

### 749 4.24.1 RSA Key

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

```
751 {  
752     "kty": "RSA",  
753     "n": "PRINTABLE-STRING-PRIMITIVE-TYPE",  
754     "e": "PRINTABLE-STRING-PRIMITIVE-TYPE"  
755 }
```

756 Where:

- 757 • kty: The JWK Key Type parameter indicating the cryptographic algorithm used with the key. The  
758 kty value for RSA public keys is fixed to the string "RSA".
- 759 • n: The base64urlUint encoded RSA public key modulus n.
- 760 • e: The base64urlUint encoded RSA public key exponent e.

### 761 4.24.2 ECC Key

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

```
763 {  
764     "kty": "EC",  
765     "crv": "PRINTABLE-STRING-PRIMITIVE-TYPE",  
766     "x": "PRINTABLE-STRING-PRIMITIVE-TYPE",  
767     "y": "PRINTABLE-STRING-PRIMITIVE-TYPE"  
768 }
```

769 Where:

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

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

## 776 4.25 OCSP-ARRAY-TYPE

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

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

## 780 4.26 UUID-ARRAY-TYPE

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

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

## 784 4.27 GPD-VERSION-TYPE

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

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

787 Where:

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

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

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

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

## 797 4.28 POP-TYPE

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

```
799 {  
800     "popkey" : PUB-KEY-TYPE ,  
801     "alg" : SIGNATURE-PRIMITIVE-TYPE ,  
802     "pop" : "PRINTABLE-STRING-PRIMITIVE-TYPE"  
803 }
```

804 Where:

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

## 810 4.29 PUB-KEY-ROLE-TYPE

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

```
812 {  
813     "role" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,  
814     "key" : PUB-KEY-TYPE  
815 }
```

816 Where:

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

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

## 824 4.30 PUB-KEY-ROLE-ARRAY-TYPE

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

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

## 828 4.31 Version Negotiation

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

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

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

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

837 If the client can support that version, it responds with OPERATION-RESPONSE-PRIMITIVE-TYPE set to  
838 OPERATION\_SUCCESS or to a suitable error.

839 If the client cannot support the requested version, it responds with the OPERATION-RESPONSE-PRIMITIVE-  
840 TYPE set to ERR\_UNSUPPORTED\_MSG\_VERSION and the ver field set to either:

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

846

## 5 OTrP Messages

847

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:

848

849

850

**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

851

852

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

853

854

### 5.1 Unprivileged Messages

855

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

856

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

857

858

859

Where:

860

- NAME SHALL be one of the following strings:

861

GetTAInformationRequest

862

GetTAInformationResponse

## 863 5.2 Privileged Messages

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

```
866 {
867     "NAME" : COMMAND-TYPE
868 }
```

869 Where:

870 • NAME SHALL be one of the following strings:

```
871     GetDeviceTEEStateRequest
872     GetDeviceTEEStateResponse
873     CreateSDRequest
874     CreateSDResponse
875     UpdateSDRequest
876     UpdateSDResponse
877     DeleteSDRequest
878     DeleteSDResponse
879     InstallTARRequest
880     InstallTARResponse
881     UpdateTARRequest
882     UpdateTARResponse
883     DeleteTARRequest
884     DeleteTARResponse
```

885 • COMMAND-TYPE: Contains the corresponding signed message.

### 886 5.2.1 Creating a Privileged Message

887 A privileged message SHALL be created as follows:

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

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

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

### 898 5.3 TEE Management Messages

899 TEE management messages are a set of **optional** OTrP messages intended for managing TEEs. Only the  
900 OWE whose OWE-Cert chains to the root CA certificate in the OWE-Whitelist of the rSD<sub>TEE</sub> SHALL be allowed  
901 to issue TEE management messages. TEE management messages SHALL always be sent to and processed  
902 by an rSD<sub>TEE</sub>. Prior to sending TEE management messages, the OWE SHALL initiate an OTrP session with  
903 rSD<sub>TEE</sub> by sending a GetDeviceTEEStateRequest message.

904 TEE management messages SHALL be formatted as follows:

```
905 {  
906     "NAME" : COMMAND-TYPE  
907 }
```

908 Where:

- 909 • NAME SHALL be one of the following strings:

```
910     StoreTEEPropertyRequest  
911     StoreTEEPropertyResponse  
912     FactoryResetRequest  
913     FactoryResetResponse
```

- 914 • COMMAND-TYPE: Contains the corresponding signed message.

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

## 916 5.4 GetTAInformationRequest

917 The `GetTAInformationRequest` is an unprivileged message that SHALL NOT be signed by the sender. It  
918 is intended for an REE application to query the status of a TA and the TA metadata from the TEE. This  
919 message SHALL always be sent to an `rSDTA`.

```
920 {  
921     "GetTAInformationRequest": {  
922         "ver": "GPD-VERSION-TYPE",  
923         "taid": "PRINTABLE-STRING-PRIMITIVE",  
924         "spid": "PRINTABLE-STRING-PRIMITIVE"  
925     }  
926 }
```

927 Where:

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

931 The `rSDTA` SHALL return `GetTAInformationResponse` with the TA metadata only if the given TA is installed  
932 using OTrP messages. TEE SHALL return `GetTAInformationResponse` with a failure status for a given  
933 TA installed on the device using any method outside the scope of OTrP Profile.

### 934 5.4.1 Processing Requirements

935 Upon receiving the `GetTAInformationRequest` message, the `rSDTA` SHALL:

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

938 Upon successfully completing the above steps, the `rSDTA` prepares a response with the TA metadata.  
939 A response message `GetTAInformationResponse` SHALL always be returned regardless of the status of  
940 the operation.



## 941 5.5 GetTAInformationResponse

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

```
945 {  
946     "GetTAInformationResponse": {  
947         "ver": "GPD-VERSION-TYPE",  
948         "status": "OPERATION-RESPONSE-PRIMITIVE-TYPE",  
949         "taid": "PRINTABLE-STRING-PRIMITIVE-TYPE",  
950         "taver": "PRINTABLE-STRING-PRIMITIVE-TYPE",  
951         "sdid": "PRINTABLE-STRING-PRIMITIVE-TYPE",  
952         "spid": "PRINTABLE-STRING-PRIMITIVE-TYPE",  
953         "tsmid": "PRINTABLE-STRING-PRIMITIVE-TYPE"  
954     }  
955 }
```

956 Where:

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

### 970 5.5.1 Error Conditions

971 If any validation listed in section 5.4.1 fails or if a TEE error occurs, the rSD<sub>TA</sub> SHALL use an appropriate  
972 OPERATION-RESPONSE-PRIMITIVE-TYPE (listed below) as the status value in the corresponding  
973 response message.

- 974 • ERR\_TA\_NOT\_FOUND
- 975 • ERR\_TEE\_BUSY
- 976 • ERR\_TEE\_FAIL
- 977 • ERR\_TEE\_RESOURCE\_FULL
- 978 • ERR\_TEE\_UNKNOWN
- 979 • ERR\_UNSUPPORTED\_MSG\_VERSION

980 See section 4.14 for details on error strings.

## 981 5.6 GetDeviceTEEStateTBSRequest

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

```
986 {  
987     "GetDeviceTEEStateTBSRequest": {  
988         "ver": "GPD-VERSION-TYPE",  
989         "tid": "PRINTABLE-STRING-PRIMITIVE-TYPE",  
990         "rid": "PRINTABLE-STRING-PRIMITIVE-TYPE",  
991         "ocspdat": OSCP-ARRAY-TYPE,  
992         "supportedsigalgs": [SIGNATURE-PRIMITIVE-TYPE]  
993     }  
994 }
```

995 Where:

- 996 • `ver`: The version of the OTrP message structured as `GPD-VERSION-TYPE`.
- 997 • `tid`: A unique value for the ongoing transaction.
- 998 • `rid`: A unique value for this message.
- 999 • `ocspdat`: `OCSP-ARRAY-TYPE` as described in section 4.25. The first element of the array is the  
1000 OSCP stapling for validating the OWE-Cert, followed by OSCP stapling for verifying each subsequent  
1001 intermediate CA in the certificate chain.
- 1002 • `supportedsigalgs`: (OPTIONAL) A list of signature algorithms supported by the OWE. Its value is  
1003 an array of `SIGNATURE-PRIMITIVE-TYPE`. If this element is absent, the TEE SHALL use any  
1004 signature algorithm defined by the `SIGNATURE-PRIMITIVE-TYPE`.

## 1005 **5.6.1 Processing Requirements**

1006 Upon receiving the `GetDeviceTEEStateRequest` message, the rSD SHALL:

- 1007 • Validate the JSON web signature associated with the request, using the OWE-Pub associated with  
1008 the OWE-Cert.
- 1009 • Determine whether the OWE-Cert chains to a root CA certificate in the OWE-Whitelist.
- 1010 • Check the revocation status of the OWE-Cert and its intermediate CA certificates in the chain, using  
1011 the OCSP stapling.
- 1012 • Cache the OCSP stapling for subsequent command checking. The TEE MAY use its own clock for  
1013 OCSP stapling validation.
- 1014 • Challenge the TFW (if available on the device) to sign a UTF-8 encoded `tid` value. The signed value  
1015 is included in the `GetDeviceTEEStateResponse` message as a part of `DSI-TYPE`.

1016 Upon successfully completing the above steps, the rSD gathers DSI to prepare a response. A response  
1017 message `GetDeviceTEEStateResponse` SHALL always be returned regardless of the status of the  
1018 operation.

## 1019 5.7 GetDeviceTEEStateTBSResponse

1020 In response to a `GetDeviceTEEStateRequest`, the rSD SHALL return a `GetDeviceTEEStateResponse`  
 1021 that encapsulates a `GetDeviceTEEStateTBSResponse` message. The JSON structure for the  
 1022 `GetDeviceTEEStateTBSResponse` SHALL be as follows:

```

1023 {
1024     "GetDeviceTEEStateTBSResponse": {
1025         "ver": "GPD-VERSION-TYPE",
1026         "status": "OPERATION-RESPONSE-PRIMITIVE-TYPE",
1027         "rid": "PRINTABLE-STRING-PRIMITIVE-TYPE",
1028         "tid": "PRINTABLE-STRING-PRIMITIVE-TYPE",
1029         "signerreq": BOOLEAN,
1030         "content": CONTENT-ENCRYPTION-TYPE
1031     }
1032 }
  
```

1033 Where:

- 1034 • `ver`: The version of the OTrP message structured as `GPD-VERSION-TYPE`.
- 1035 • `status`: An `OPERATION-RESPONSE-PRIMITIVE-TYPE` indicating the status of the  
 1036 `GetDeviceTEEStateRequest` operation. If successful, the value of `status` SHALL be  
 1037 `OPERATION_SUCCESS`; otherwise its value SHALL be an error string listed in section 5.7.1.
- 1038 • `rid`: A unique value identifying the `GetDeviceTEEStateRequest` message.
- 1039 • `tid`: A unique value identifying the OTrP session. Matches the `tid` value in  
 1040 `GetDeviceTEEStateRequest` message.
- 1041 • `signerreq`: A Boolean value that indicates whether the OWE should send its signer certificate and  
 1042 OCSP stapling again in the subsequent messages. It is recommended that the `signerreq` value is  
 1043 set to `false`. If the value is set to `false`, the TEE SHALL cache the OWE signer certificate and  
 1044 OCSP stapling.
- 1045 • `content`: JWE encrypted data as a `CONTENT-ENCRYPTION-TYPE`.

1046 The following JSON structure SHALL be used as an input to the JWE while generating `CONTENT-`  
 1047 `ENCRYPTION-TYPE`.

```

1048 {
1049     "dsi": DSI-CONTENT-TYPE,
1050     "nextnonce": "PRINTABLE-STRING-PRIMITIVE-TYPE"
1051 }
  
```

1052 Where:

- 1053 • `dsi`: The `DSI-CONTENT-TYPE` that represents the current device state.

- 1054       • nextnonce: A unique value that the OWE SHALL use as nonce in the next request. See  
1055       section 2.10.4 for details.

### 1056   **5.7.1 Error Conditions**

1057   If any validation listed in section 5.6.1 fails or if a TEE error occurs, the rSD SHALL use an appropriate  
1058   OPERATION-RESPONSE-PRIMITIVE-TYPE (listed below) as the status value in the corresponding  
1059   response message.

- 1060       • ERR\_OCSP\_INVALID
- 1061       • ERR\_OWE\_NOT\_TRUSTED
- 1062       • ERR\_REQUEST\_INVALID
- 1063       • ERR\_TEE\_BUSY
- 1064       • ERR\_TEE\_FAIL
- 1065       • ERR\_TEE\_RESOURCE\_FULL
- 1066       • ERR\_TEE\_UNKNOWN
- 1067       • ERR\_TFW\_NOT\_TRUSTED
- 1068       • ERR\_UNSUPPORTED\_CRYPTO\_ALG
- 1069       • ERR\_UNSUPPORTED\_MSG\_VERSION

1070   See section 4.14 for details on error strings.

## 1071 5.8 CreateSDTBSRequest

1072 An OWE SHALL issue a `CreateSDTBSRequest` message to create a new Security Domain with the given  
 1073 parameters. The message SHALL be signed using the JWS scheme and encapsulated in a  
 1074 `CreateSDRequest` message. This message SHALL always be sent to the `rSDTA`. The JSON structure for the  
 1075 `CreateSDTBSRequest` SHALL be as follows:

```
1076 {
1077     "CreateSDTBSRequest" : COMMAND-PARAMETER-TYPE
1078 }
```

1079 Within the `COMMAND-PARAMETER-TYPE`, the following JSON structure is used as an input to the JWE while  
 1080 generating `CONTENT-ENCRYPTION-TYPE`.

```
1081 {
1082     "spid" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1083     "sdid" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1084     "spcert" : "CERT-PRIMITIVE-TYPE" ,
1085     "tsmid" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1086     "did" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1087     "sd_data" : "PRINTABLE-STRING-PRIMITIVE-TYPE"
1088 }
```

1089 Where:

- 1090 • `spid`: The base64 encoded Service Provider identifier that is to be associated with the SD. See  
 1091 section 2.6 for the `spid` generation.
- 1092 • `sdid`: The base64 encoded UUID of the SD to be created. The `sdid` SHALL remain unchangeable  
 1093 throughout its life cycle.
- 1094 • `spcert`: SP-Cert formatted as `CERT-PRIMITIVE-TYPE` that is to be associated with the SD. Only  
 1095 TAs that are signed using a key associated with the SP-Cert SHALL be allowed to be installed in the  
 1096 SD.
- 1097 • `tsmid`: The identifier of the OWE that issued the request.
- 1098 • `did`: The base64 encoded SHA-256 hash of the TEE-Cert binary. `did` is used as the device  
 1099 identifier to which the request is issued.
- 1100 • `sd_data`: (OPTIONAL) The base64url encoded SD personalization data. This element may be used  
 1101 to equip the SD with credentials required to support TMF commands. The format of the SD  
 1102 personalization data SHALL be a DER-encoded `StoredDataObject` as defined in [TMF ASN.1]  
 1103 section 8.3.3.6.

1104 See section 2.10.6 for more information on key use.

## 1105 5.8.1 Processing Requirements

1106 Before authorizing SD creation, the `rSDTA` SHALL:

- 1107 • Validate the JSON web signature associated with this request.
- 1108 • Determine whether the OWE-Cert chains to a root CA certificate in OWE-Whitelist.
- 1109 • Check the revocation status of the OWE-Cert chain, using the cached OCSP stapling. If the cache is  
1110 unavailable or expired, the `rSD` SHALL return the corresponding response with an error string along  
1111 with an indication (`signerreq` set to `true`) to provide OCSP stapling in the next request. The OWE  
1112 may reissue the request with OCSP stapling.
- 1113 • Compare the `dsihash` value to the SHA-256 hash of the internal `DSI-TYPE` to ensure that the DSI  
1114 has not changed since the last changes requested by the OWE.
- 1115 • Compare `nonce` to the last `nextnonce` sent to the OWE to ensure that no new operation has been  
1116 authorized on SDs and TAs associated with the OWE since the last operation requested by the OWE.
- 1117 • Decrypt the `ciphertext` element of the `CONTENT-ENCRYPTION-TYPE` to obtain the SD  
1118 information.
- 1119 • Validate the format of the `spcert`.
- 1120 • Verify the `did` to ensure that the request is intended for the correct device.
- 1121 • Verify that the given `sdid` is valid for the SP, using the process defined in section 2.4.
- 1122 • Verify that the SD with the given `sdid` does not already exist.
- 1123 • Verify that the `tsmid` matches the OWE identifier in the OWE-Cert to ensure that the OWE issuing  
1124 the request has access to the SD. See section 2.9 for details.

1125 Upon successfully completing the above processing, the SD with the given parameters SHALL be created. If  
1126 the `spid` associated with the SD is not assigned to any SDs on the device, then the TEE SHALL also generate  
1127 a key pair called SP-AIK. A response message `CreateSDResponse` SHALL always be returned regardless  
1128 of the status of the operation.



## 1129 5.9 CreateSDTBSResponse

1130 In response to a CreateSDRequest, the rSD<sub>TA</sub> SHALL return a CreateSDResponse, encapsulating the  
1131 CreateSDTBSResponse message. The JSON structure for the CreateSDTBSResponse is as follows:

```
1132 {
1133     "CreateSDTBSResponse" : RESPONSE-PARAMETER-TYPE
1134 }
```

1135 Within the RESPONSE-PARAMETER-TYPE, the following JSON structure is used as an input to the JWE while  
1136 generating CONTENT-ENCRYPTION-TYPE.

### 1137 Version 1.1

```
1138 {
1139     "status" : "OPERATION-RESPONSE-PRIMITIVE-TYPE" ,
1140     "did" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1141     "sdid" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1142     "spaik" : PUB-KEY-ROLE-ARRAY-TYPE ,
1143     "dsi" : DSI-CONTENT-TYPE ,
1144     "nextnonce" : "PRINTABLE-STRING-PRIMITIVE-TYPE"
1145 }
```

### 1146 Version 1.0

```
1147 {
1148     "status" : "OPERATION-RESPONSE-PRIMITIVE-TYPE" ,
1149     "did" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1150     "sdid" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1151     "spaik" : PUB-KEY-TYPE ,
1152     "dsi" : DSI-CONTENT-TYPE ,
1153     "nextnonce" : "PRINTABLE-STRING-PRIMITIVE-TYPE"
1154 }
```

1155 Where:

- 1156 • status: An OPERATION-RESPONSE-PRIMITIVE-TYPE indicating the status of the  
1157 CreateSDRequest operation. If the SD is created successfully, the value of status SHALL be  
1158 OPERATION\_SUCCESS; otherwise its value SHALL be one of the error strings listed in section 5.9.1.
- 1159 • did: The value of did from the CreateSDRequest.
- 1160 • sdid: The value of the sdid from the CreateSDRequest.

- 1161       • spaik:
- 1162           **Version 1.1**
- 1163           The SP-AIK-Pub keys structured as a PUB-KEY-ROLE-ARRAY-TYPE.
- 1164           **Version 1.0**
- 1165           A single SP-AIK-Pub key returned as a PUB-KEY-TYPE.
- 1166           This element is returned only if the request caused a new SP-AIK to be generated.
- 1167       • dsi: The DSI-CONTENT-TYPE for the new device state. This element is returned only when the
- 1168           nextdsi is set to true in the CreateSDRequest.
- 1169       • nextnonce: A unique value that the OWE SHALL use as nonce in the next request. See
- 1170           section 2.10.4 for details.

### 1171   **5.9.1 Error Conditions**

1172   If any validation listed in section 5.8.1 fails or if a TEE error occurs, the rSD<sub>TA</sub> SHALL use an appropriate

1173   OPERATION-RESPONSE-PRIMITIVE-TYPE (listed below) as the status value in the corresponding

1174   response message.

- 1175       • ERR\_DEV\_STATE\_MISMATCH
- 1176       • ERR\_INVALID\_UUID
- 1177       • ERR\_OCSP\_INVALID
- 1178       • ERR\_OWE\_NOT\_TRUSTED
- 1179       • ERR\_REQUEST\_INVALID
- 1180       • ERR\_REVERT\_OPERATION
- 1181       • ERR\_SDID\_ALREADY\_USED
- 1182       • ERR\_SPCERT\_INVALID
- 1183       • ERR\_TEE\_BUSY
- 1184       • ERR\_TEE\_FAIL
- 1185       • ERR\_TEE\_RESOURCE\_FULL
- 1186       • ERR\_TEE\_UNKNOWN
- 1187       • ERR\_UNSUPPORTED\_CRYPT\_ALG
- 1188       • ERR\_UNSUPPORTED\_MSG\_VERSION

1189   See section 4.14 for details on error strings.

1190   **Note:** If the OWE receives ERR\_REVERT\_OPERATION, it is recommended that the OWE remove the recently

1191   created SD; otherwise the DSI value will be inconsistent.

## 1192 5.10 UpdateSDTBSRequest

1193 An OWE SHALL issue an UpdateSDTBSRequest message to update SD metadata with the given  
 1194 parameters. The message SHALL be signed using the JWS scheme and encapsulated in an  
 1195 UpdateSDRequest message. This message SHALL always be sent to rSD<sub>TA</sub>. A JSON structure for the  
 1196 UpdateSDTBSRequest SHALL be as follows:

```
1197 {
1198     "UpdateSDTBSRequest":COMMAND-PARAMETER-TYPE
1199 }
```

1200 Within the COMMAND-PARAMETER-TYPE, the following JSON structure is used as an input to the JWE while  
 1201 generating CONTENT-ENCRYPTION-TYPE:

### 1202 Version 1.1

```
1203 {
1204     "tsmid": "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1205     "spid": "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1206     "did": "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1207     "sdid": "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1208     "changes": {
1209         "spcert": [ "CERT-PRIMITIVE-TYPE" ] ,
1210         "deloldspcert": [ "PRINTABLE-STRING-PRIMITIVE-TYPE" ] ,
1211         "sd_data": "PRINTABLE-STRING-PRIMITIVE-TYPE"
1212     }
1213 }
```

### 1214 Version 1.0

```
1215 {
1216     "tsmid": "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1217     "spid": "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1218     "did": "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1219     "sdid": "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1220     "changes": {
1221         "newspid": "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1222         "spcert": [ "CERT-PRIMITIVE-TYPE" ] ,
1223         "deloldspcert": [ "PRINTABLE-STRING-PRIMITIVE-TYPE" ] ,
1224         "sd_data": "PRINTABLE-STRING-PRIMITIVE-TYPE"
```

```

1225     }
1226 }

```

1227 Where:

- 1228 • `tsmid`: The identifier of the OWE that issued the request.
- 1229 • `spid`: The base64 encoded service provide identifier that is associated with the SD.
- 1230 • `did`: The base64 encoded SHA-256 hash of the TEE-Cert binary. `did` is used as the device  
1231 identifier to which the request is issued.
- 1232 • `sdid`: The base64 encoded UUID representing the SD to be updated.
- 1233 • `changes`: A JSON structure containing parameters to be updated. All elements within this JSON  
1234 structure are **optional**.
- 1235 • `newspid`:

#### 1236 **Version 1.1**

1237 Deprecated in v1.1.

1238 The SD name is derived from the `spid`, so changing the `spid` effectively changes the  
1239 SD name. In a pure OTrP system this is permissible, but not in TMF.

#### 1240 **Version 1.0**

1241 The new base64 encoded service provide identifier that is to be associated with the SD. If the  
1242 `newspid` is not associated with any existing SDs in the device, the rSD SHALL generate a new  
1243 SP-AIK key pair for the `newspid`.

- 1244 • `spcert`: An array of SP-Certs formatted as `CERT-PRIMITIVE-TYPE` that is to be associated with  
1245 the SD.
  - 1246 • `deloldspcert`: The base64 encoded SHA-256 hash value of SP-Certs previously assigned to the  
1247 SD that are to be deleted.
- 1248 **Note:** Deleting the certificate without supplying a new certificate would make it impossible to verify  
1249 new OTrP sessions.
- 1250 • `sd_data`: The base64 encoded SD personalization data. This element may be used to equip the SD  
1251 with credentials required to support TMF commands. The format of the SD personalization data  
1252 SHALL be a DER-encoded `StoredDataObject` as defined in [TMF ASN.1] section 8.3.3.6. See  
1253 section 2.10.6 for more information on key use. If the OWE updates keys, the SD SHOULD generate  
1254 a new SP-AIK-Pub.

1255 **Important:** Beginning with version 1.1, implementations SHALL be able to update keys and  
1256 certificates.

## 1257 **5.10.1 Processing Requirements**

1258 Before authorizing the SD update, the rSD<sub>TA</sub> SHALL:

- 1259 • Validate the JSON web signature associated with the request.
- 1260 • Determine whether the OWE-Cert chains to a root CA certificate in OWE-Whitelist.
- 1261 • Check the revocation status of the OWE-Cert chain, using the cached OCSP stapling. If the cache is  
1262 unavailable or expired, the rSD SHALL return the corresponding response with an error string along  
1263 with an indication (`signerreq` set to `true`) to provide OCSP stapling in the next request. The OWE  
1264 may reissue the request with OCSP stapling.

- 1265 • Compare the `dsihash` value to the SHA-256 hash of the internal `DSI-TYPE` to ensure that the DSI  
1266 has not changed since the last changes requested by the OWE.
  - 1267 • Compare `nonce` to the last `nextnonce` sent to the OWE to ensure that no new operation has been  
1268 authorized on SDs and TAs associated with the OWE since the last operation requested by the OWE.
  - 1269 • Decrypt the `ciphertext` element of the `CONTENT-ENCRYPTION-TYPE` to obtain the update  
1270 parameters.
  - 1271 • Verify the `did` to ensure that the request is intended for the correct device.
  - 1272 • Verify that the given `sdid` is valid for the SP, using the process defined in section 2.4.
  - 1273 • Verify that the SD with the given `sdid` exists.
  - 1274 • Verify that the `tsmid` matches the OWE identifier in the OWE-Cert to ensure that the OWE issuing  
1275 the request has access to the SD. See section 2.9 for details.
- 1276 Upon successfully completing the above requirements, the specified SD is updated with the given parameters.  
1277 If the update operation results in the generation of a new SP-AIK, the newly generated SP-AIK SHALL replace  
1278 the existing SP-AIK. A response message `UpdateSDResponse` SHALL always be returned regardless of the  
1279 status of the operation.

## 1280 5.11 UpdateSDTBSResponse

1281 In response to an UpdateSDRequest, the rSD<sub>TA</sub> SHALL return an UpdateSDResponse, encapsulating the  
 1282 UpdateSDTBSResponse message. The JSON structure for the UpdateSDTBSResponse SHALL be as  
 1283 follows:

```
1284 {
1285     "UpdateSDTBSResponse" : RESPONSE-PARAMETER-TYPE
1286 }
```

1287 Within the RESPONSE-PARAMETER-TYPE, the following JSON structure is used as an input to the JWE while  
 1288 generating CONTENT-ENCRYPTION-TYPE:

### 1289 Version 1.1

```
1290 {
1291     "status" : "OPERATION-RESPONSE-PRIMITIVE-TYPE" ,
1292     "did" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1293     "spaik" : PUB-KEY-ROLE-ARRAY-TYPE ,
1294     "dsi" : DSI-CONTENT-TYPE ,
1295     "nextnonce" : "PRINTABLE-STRING-PRIMITIVE-TYPE"
1296 }
```

### 1297 Version 1.0

```
1298 {
1299     "status" : "OPERATION-RESPONSE-PRIMITIVE-TYPE" ,
1300     "did" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1301     "spaik" : PUB-KEY-TYPE ,
1302     "dsi" : DSI-CONTENT-TYPE ,
1303     "nextnonce" : "PRINTABLE-STRING-PRIMITIVE-TYPE"
1304 }
```

1305 Where:

- 1306 • status: An OPERATION-RESPONSE-PRIMITIVE-TYPE indicating the status of the  
 1307 UpdateSDRequest operation. If the SD is updated successfully, the value of status SHALL be  
 1308 OPERATION\_SUCCESS; otherwise its value SHALL be an error string listed in section 5.11.1.
- 1309 • did: The value of did from the UpdateSDRequest.

- 1310       • spaik:
- 1311           **Version 1.1**
- 1312           The SP-AIK-Pub keys structured as a PUB-KEY-ROLE-ARRAY-TYPE.
- 1313           **Version 1.0**
- 1314           A single SP-AIK-Pub key returned as a PUB-KEY-TYPE.
- 1315           This element is returned only if the UpdateSDRequest causes the rSD<sub>TA</sub> to generate a new SP-AIK.
- 1316       • dsi: The DSI-CONTENT-TYPE for the new device state. This element is returned only when the
- 1317           nextdsi is set to true in the UpdateSDRequest.
- 1318       • nextnonce: A unique value that the OWE SHALL use as nonce in the next request. See
- 1319           section 2.10.4 for details.

### 1320   **5.11.1 Error Conditions**

1321   If any validation listed in section 5.10.1 fails or if a TEE error occurs, the rSD<sub>TA</sub> SHALL use an appropriate

1322   OPERATION-RESPONSE-PRIMITIVE-TYPE (listed below) as the status value in the corresponding

1323   response message.

- 1324       • ERR\_DEV\_STATE\_MISMATCH
- 1325       • ERR\_INVALID\_UUID
- 1326       • ERR\_OCSP\_INVALID
- 1327       • ERR\_OWE\_NOT\_TRUSTED
- 1328       • ERR\_REQUEST\_INVALID
- 1329       • ERR\_SPCERT\_INVALID
- 1330       • ERR\_TEE\_BUSY
- 1331       • ERR\_TEE\_FAIL
- 1332       • ERR\_TEE\_RESOURCE\_FULL
- 1333       • ERR\_TEE\_UNKNOWN
- 1334       • ERR\_UNSUPPORTED\_CRYPTO\_ALG
- 1335       • ERR\_UNSUPPORTED\_MSG\_VERSION
- 1336       • ERR\_UPDATING\_DATA

1337   See section 4.14 for details on error strings.

1338   **Note:** If the OWE receives ERR\_REVERT\_OPERATION, it is recommended that the OWE remove the recently

1339   created SD; otherwise the DSI value will be inconsistent.

## 1340 5.12 DeleteSDTBSRequest

1341 An OWE SHALL issue a `DeleteSDTBSRequest` message to delete a specified SD and optionally delete all  
1342 TAs contained within the SD. The message SHALL be signed using the JWS scheme and encapsulated in a  
1343 `DeleteSDRequest` message. This message SHALL always be sent to the `rSDTA`. The JSON structure for the  
1344 `DeleteSDTBSRequest` SHALL be as follows:

```
1345 {  
1346     "DeleteSDTBSRequest" : COMMAND-PARAMETER-TYPE  
1347 }
```

1348 Within the `COMMAND-PARAMETER-TYPE`, the following JSON structure is used as an input to the JWE while  
1349 generating `CONTENT-ENCRYPTION-TYPE`:

```
1350 {  
1351     "tsmid" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,  
1352     "did" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,  
1353     "sdid" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,  
1354     "deletetas" : BOOLEAN  
1355 }
```

1356 Where:

- 1357 • `tsmid`: The identifier of the OWE that issued the request.
- 1358 • `did`: The base64 encoded SHA-256 hash of the TEE-Cert binary. `did` is used as the device  
1359 identifier to which the request is issued.
- 1360 • `sdid`: The base64 encoded UUID representing the SD to be deleted.
- 1361 • `deletetas`: A Boolean value indicating whether to delete all TAs within the SD. If set to `false`,  
1362 deleting an SD with one or more TAs installed SHALL cause a failure.



## 1363 5.12.1 Processing Requirements

1364 Before authorizing the deletion of the SD, the `rSDTA` SHALL:

- 1365 • Validate the JSON web signature associated with the request.
- 1366 • Determine whether the OWE-Cert chains to a root CA certificate in the OWE-Whitelist.
- 1367 • Check the revocation status of the OWE-Cert chain, using the cached OCSP stapling. If the cache is  
1368 unavailable or expired, the `rSD` SHALL return the corresponding response with an error string along  
1369 with an indication (`signerreq` set to `true`) to provide OCSP stapling in the next request. The OWE  
1370 may reissue the request with OCSP stapling.
- 1371 • Compare the `dsihash` value to the SHA-256 hash of the internal `DSI-TYPE` to ensure that the DSI  
1372 has not changed since the last changes requested by the OWE.
- 1373 • Compare `nonce` to the last `nextnonce` sent to the OWE to ensure that no new operation has been  
1374 authorized on SDs and TAs associated with the OWE since the last operation requested by the OWE.
- 1375 • Decrypt the `ciphertext` element of the `CONTENT-ENCRYPTION-TYPE` to obtain the deletion  
1376 parameters.
- 1377 • Verify the `did` to ensure that the request is intended for the correct device.
- 1378 • Verify that the `tsmid` matches the OWE identifier in the OWE-Cert to ensure that the OWE issuing  
1379 the request has access to the SD. See section 2.9 for details.
- 1380 • Ensure that, if `deletetas` is set to `false`, the SD contains no TAs; otherwise the deletion SHALL  
1381 be aborted, resulting in a failure.

1382 Upon successfully completing the above requirements, the specified SD SHALL be deleted. A response  
1383 message `DeleteSDResponse` SHALL always be returned regardless of the status of the operation.

## 1384 5.13 DeleteSDTBSResponse

1385 In response to a DeleteSDRequest command, the rSD<sub>TA</sub> SHALL return a DeleteSDResponse,  
1386 encapsulating the DeleteSDTBSResponse message. The JSON structure for the DeleteSDTBSResponse  
1387 is as follows:

```
1388 {  
1389     "DeleteSDTBSResponse" : RESPONSE-PARAMETER-TYPE  
1390 }
```

1391 Within the RESPONSE-PARAMETER-TYPE, the following JSON structure is used as an input to the JWE while  
1392 generating CONTENT-ENCRYPTION-TYPE.

```
1393 {  
1394     "status" : "OPERATION-RESPONSE-PRIMITIVE-TYPE" ,  
1395     "did" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,  
1396     "dsi" : DSI-CONTENT-TYPE ,  
1397     "nextnonce" : "PRINTABLE-STRING-PRIMITIVE-TYPE"  
1398 }
```

1399 Where:

- 1400 • status: An OPERATION-RESPONSE-PRIMITIVE-TYPE indicating the status of the  
1401 DeleteSDRequest operation. If the SD is deleted successfully, the value of status SHALL be  
1402 OPERATION\_SUCCESS; otherwise its value SHALL be an error string listed in section 5.13.1.
- 1403 • did: The value of did from the DeleteSDRequest.
- 1404 • dsi: The DSI-CONTENT-TYPE for the new device state. This element is returned only when the  
1405 nextdsi is set to true in the DeleteSDRequest.
- 1406 • nextnonce: A unique value that the OWE SHALL use as nonce in the next request. See  
1407 section 2.10.4 for details.

### 1408 5.13.1 Error Conditions

1409 If any validation listed in section 5.12.1 fails or if a TEE error occurs, the rSD<sub>TA</sub> SHALL use an appropriate  
1410 OPERATION-RESPONSE-PRIMITIVE-TYPE (listed below) as the status value in the corresponding  
1411 response message.

- 1412 • ERR\_DEV\_STATE\_MISMATCH
- 1413 • ERR\_OCSP\_INVALID
- 1414 • ERR\_OWE\_NOT\_TRUSTED
- 1415 • ERR\_REQUEST\_INVALID
- 1416 • ERR\_SD\_NOT\_EMPTY
- 1417 • ERR\_SPCERT\_INVALID
- 1418 • ERR\_TEE\_BUSY
- 1419 • ERR\_TEE\_FAIL
- 1420 • ERR\_TEE\_RESOURCE\_FULL
- 1421 • ERR\_TEE\_UNKNOWN
- 1422 • ERR\_UNSUPPORTED\_CRYPTO\_ALG
- 1423 • ERR\_UNSUPPORTED\_MSG\_VERSION

1424 See section 4.14 for details on error strings.

## 1425 5.14 InstallTATBSRequest

1426 An OWE SHALL issue an `InstallTATBSRequest` message to install a specified TA into a specified Security  
 1427 Domain. The message SHALL be signed using the JWS scheme and encapsulated in an `InstallTAResponse`  
 1428 message. This message SHALL always be sent to the `rSDTA`. The JSON structure for `InstallTATBSRequest`  
 1429 is as follows:

```
1430 {
1431     "InstallTATBSRequest": COMMAND-PARAMETER-TYPE
1432 }
```

1433 Within the `PROTECTED-HEADER-TYPE` (section 4.4), the following JSON structure is used as an input to the  
 1434 JWE while generating `CONTENT-ENCRYPTION-TYPE`.

```
1435 {
1436     "tsmid": "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1437     "did": "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1438     "spid": "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1439     "sdid": "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1440     "spcert": "CERT-PRIMITIVE-TYPE" ,
1441     "taid": "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1442     "taver": "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1443     "pop_data": POP-TYPE
1444 }
```

1445 Where:

- 1446 • `tsmid`: The identifier of the OWE that issued the request.
- 1447 • `did`: The base64 encoded SHA-256 hash of the TEE-Cert binary. `did` is used as the device  
 1448 identifier to which the request is issued.
- 1449 • `spid`: The base64 encoded Service Provider identifier that is associated with the SD.
- 1450 • `sdid`: The base64 encoded UUID of the SD where the TA is to be installed.
- 1451 • `spcert`: (OPTIONAL) The SP-Cert formatted as `CERT-PRIMITIVE-TYPE` that signed the TA. This  
 1452 element is provided when the TA is signed with an SP-Cert that was not previously associated with  
 1453 the SD.
- 1454 • `taid`: The base64 encoded UUID of the TA to be installed.
- 1455 • `taver`: The string containing the TA version information.
- 1456 • `pop_data`: (OPTIONAL) `POP-TYPE` value SHALL be included when the given `taid` is a UUID  
 1457 version 5. It is used to perform a verification of proof of possession of a UUID version 5 as defined in  
 1458 [TMF ASN.1] section 8.3.3.7. (See details in Annex D.)

1459 Additionally, the `InstallTATBSRequest` SHALL include the following JSON elements in the `COMMAND-`  
 1460 `PARAMETER-TYPE`.

1461 `"encrypted_ta_bin":CONTENT-ENCRYPTION-TYPE`

1462 `"encrypted_ta_data":CONTENT-ENCRYPTION-TYPE`

- 1463 • `encrypted_ta_bin`: An encrypted TA binary structured as a `CONTENT-ENCRYPTION-TYPE` where  
 1464 the CEK is wrapped using the SP-AIK-Pub.
- 1465 • `encrypted_ta_data`: (OPTIONAL) An encrypted TA personalization data structured as a  
 1466 `CONTENT-ENCRYPTION-TYPE` where the CEK is wrapped using the SP-AIK-Pub. The TA should be  
 1467 able to access the personalization data via interfaces defined in [TEE Core]. The format of the TA  
 1468 personalization data SHALL be a DER-encoded `StoredDataObject` as defined in [TMF ASN.1]  
 1469 section 8.3.3.6.

### 1470 5.14.1 Processing Requirements

1471 Before authorizing the deletion of the SD, the `rSDTA` SHALL:

- 1472 • Validate the JSON web signature associated with the request.
- 1473 • Determine whether the OWE-Cert chains to a root CA certificate in the OWE-Whitelist.
- 1474 • Check the revocation status of the OWE-Cert chain, using the cached OCSP stapling. If the cache is  
 1475 unavailable or expired, the `rSD` SHALL return the corresponding response with an error string along  
 1476 with an indication (`signerreq` set to `true`) to provide OCSP stapling in the next request. The OWE  
 1477 may reissue the request with OCSP stapling.
- 1478 • Compare the `dsihash` value to the SHA-256 hash of the internal `DSI-TYPE` to ensure that the DSI  
 1479 has not changed since the last changes requested by the OWE.
- 1480 • Compare `nonce` to the last `nextnonce` sent to the OWE to ensure that no new operation has been  
 1481 authorized on SDs and TAs associated with the OWE since the last operation requested by the OWE.
- 1482 • Decrypt the `ciphertext` element of the `CONTENT-ENCRYPTION-TYPE` to obtain the TA  
 1483 information.
- 1484 • Validate the format of the `spcert`.
- 1485 • Verify the `did` to ensure that the request is intended for the correct device.
- 1486 • Verify that the `tsmid` matches the OWE identifier in the OWE-Cert to ensure that the OWE issuing  
 1487 the request has access to the SD. See section 2.9 for details.
- 1488 • If `taid` is a UUID version 5, validate the proof of possession of the TA UUID as defined in  
 1489 [TMF ASN.1] section 8.3.3.7. (See details in Annex D.)
- 1490 • Verify that, if the TA already exists in the device, the version of the TA to be installed is higher than the  
 1491 existing TA.
- 1492 • Use SP-AIK-Priv key to decrypt TA binary and personalization data. If the version of the TA  
 1493 associated with the TA binary is different than the `taver` element, the `rSDTA` SHALL abort the update  
 1494 process.
- 1495 • Validate the TA signature using an SP-Cert associated with the SD. The TA signing mechanism may  
 1496 be specific to the TEE OS.

1497 Upon successfully completing the above requirements, the given TA SHALL be installed into the specified SD.  
1498 A response message InstallTAResponse SHALL always be returned regardless of the status of the  
1499 operation.

## 1500 5.15 InstallTATBSResponse

1501 In response to an InstallTAResponse, the rSD<sub>TA</sub> SHALL return an InstallTAResponse, encapsulating  
1502 the InstallTATBSResponse message. The JSON structure for the InstallTATBSResponse SHALL be  
1503 as follows:

```
1504 {  
1505     "InstallTATBSResponse" : RESPONSE-PARAMETER-TYPE  
1506 }
```

1507 Within the RESPONSE-PARAMETER-TYPE, the following JSON structure SHALL be used as an input to the  
1508 JWE while generating CONTENT-ENCRYPTION-TYPE.

```
1509 {  
1510     "status" : "OPERATION-RESPONSE-PRIMITIVE-TYPE" ,  
1511     "did" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,  
1512     "dsi" : DSI-CONTENT-TYPE ,  
1513     "nextnonce" : "PRINTABLE-STRING-PRIMITIVE-TYPE"  
1514 }
```

1515 Where:

- 1516 • status: An OPERATION-RESPONSE-PRIMITIVE-TYPE indicating the status of the  
1517 InstallTAResponse operation. If the TA is installed successfully, the value of status SHALL be  
1518 OPERATION\_SUCCESS; otherwise its value SHALL be an error string listed in section 5.15.1.
- 1519 • did: The value of did from previous messages.
- 1520 • dsi: The DSI-CONTENT-TYPE for the new device state. This element is returned only when the  
1521 nextdsi is set to true in the InstallTAResponse.
- 1522 • nextnonce: A unique value that the OWE SHALL use as nonce in the next request. See  
1523 section 2.10.4 for details.

### 1524 5.15.1 Error Conditions

1525 If any validation listed in section 5.14.1 fails or if a TEE error occurs, the rSD<sub>TA</sub> SHALL use an appropriate  
1526 OPERATION-RESPONSE-PRIMITIVE-TYPE (listed below) as the status value in the corresponding  
1527 response message.

- 1528 • ERR\_DEV\_STATE\_MISMATCH
- 1529 • ERR\_INVALID\_UUID
- 1530 • ERR\_OCSP\_INVALID
- 1531 • ERR\_OWE\_NOT\_TRUSTED
- 1532 • ERR\_REQUEST\_INVALID
- 1533 • ERR\_REVERT\_OPERATION
- 1534 • ERR\_SPCERT\_INVALID
- 1535 • ERR\_TA\_ALREADY\_INSTALLED
- 1536 • ERR\_TA\_INVALID
- 1537 • ERR\_TEE\_BUSY
- 1538 • ERR\_TEE\_FAIL
- 1539 • ERR\_TEE\_RESOURCE\_FULL
- 1540 • ERR\_TEE\_UNKNOWN
- 1541 • ERR\_UNSUPPORTED\_CRYPTO\_ALG
- 1542 • ERR\_UNSUPPORTED\_MSG\_VERSION

1543 See section 4.14 for details on error strings.

1544 **Note:** If the OWE receives ERR\_REVERT\_OPERATION, it is recommended that the OWE remove the recently  
1545 created SD; otherwise the DSI value will be inconsistent.



## 1546 5.16 UpdateTATBSRequest

1547 An OWE SHALL issue an `UpdateTATBSRequest` message to update TA binary and/or TA personalization  
 1548 data. The message SHALL be signed using the JWS scheme and encapsulated in an `UpdateTARequest`  
 1549 message. This message SHALL always be sent to `rSDTA`. The JSON structure for the `UpdateTATBSRequest`  
 1550 SHALL be as follows:

```
1551 {
1552     "UpdateTATBSRequest" : COMMAND-PARAMETER-TYPE
1553 }
```

1554 Within the `COMMAND-PARAMETER-TYPE`, the following JSON structure SHALL be used as an input to the JWE  
 1555 while generating `CONTENT-ENCRYPTION-TYPE`:

```
1556 {
1557     "tsmid" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1558     "did" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1559     "spid" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1560     "sdid" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1561     "spcert" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1562     "taid" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1563     "newtaver" : "PRINTABLE-STRING-PRIMITIVE-TYPE" " " ,
1564     "pop_data" : POP-TYPE
1565 }
```

1566 Where:

- 1567 • `tsmid`: The identifier of the OWE that issued the request.
- 1568 • `did`: The base64 encoded SHA-256 hash of the TEE-Cert binary. `did` is used as the device  
 1569 identifier to which the request is issued.
- 1570 • `spid`: The base64 encoded Service Provider identifier that is associated with the SD.
- 1571 • `sdid`: The base64 encoded UUID of the SD where the TA is to be installed.
- 1572 • `spcert`: (OPTIONAL) The SP-Cert formatted as `CERT-PRIMITIVE-TYPE` that signed the TA. This  
 1573 element is provided when the TA is signed with a SP-Cert that was not previously associated with the  
 1574 SD.
- 1575 • `taid`: The base64 encoded UUID of the TA to be installed.
- 1576 • `newtaver`: (OPTIONAL) The string containing the TA version information that is to be updated.
- 1577 • `pop_data`: (OPTIONAL) The value of `POP-TYPE` SHALL be included when the given `taid` is a  
 1578 UUID version 5. It is used to perform a verification of proof of possession of a UUID version 5 as  
 1579 defined in [TMF ASN.1] section 8.3.3.7. (See details in Annex D.)

1580 Additionally, the `UpdateTATBSRequest` SHALL include at least one of the following JSON elements in the  
1581 `COMMAND-PARAMETER-TYPE`.

1582 `"encrypted_ta_bin" :CONTENT-ENCRYPTION-TYPE`

1583 `"encrypted_ta_data" :CONTENT-ENCRYPTION-TYPE`

- 1584 • `encrypted_ta_bin`: An encrypted TA binary that replaces the existing TA binary, structured as a  
1585 `CONTENT-ENCRYPTION-TYPE` where the CEK is wrapped using the SP-AIK-Pub.
- 1586 • `encrypted_ta_data`: (OPTIONAL) An encrypted TA personalization data to replace the existing TA  
1587 personalization data, structured as a `CONTENT-ENCRYPTION-TYPE` where the CEK is wrapped using  
1588 the SP-AIK-Pub. The TA should be able to access the personalization data via interfaces defined in  
1589 [TEE Core]. The format of the TA personalization data SHALL be a DER-encoded  
1590 `StoredDataObject` as defined in [TMF ASN.1] section 8.3.3.6.

### 1591 5.16.1 Processing Requirements

1592 Before authorizing the update on the TA, the `rSDTA` SHALL:

- 1593 • Validate the JSON web signature associated with the request.
- 1594 • Determine whether the OWE-Cert chains to a root CA certificate in OWE-Whitelist.
- 1595 • Check the revocation status of the OWE-Cert chain, using the cached OCSP stapling. If the cache is  
1596 unavailable or expired, the `rSD` SHALL return the corresponding response with an error string along  
1597 with an indication (`signerreq` set to `true`) to provide OCSP stapling in the next request. The OWE  
1598 may reissue the request with OCSP stapling.
- 1599 • Compare the `dsihash` value to the SHA-256 hash of the internal `DSI-TYPE` to ensure that the DSI  
1600 has not changed since the last changes requested by the OWE.
- 1601 • Compare `nonce` to the last `nextnonce` sent to the OWE to ensure that no new operation has been  
1602 authorized on SDs and TAs associated with the OWE since the last operation requested by the OWE.
- 1603 • Decrypt the `ciphertext` element of the `CONTENT-ENCRYPTION-TYPE` to obtain the TA  
1604 information.
- 1605 • Verify the `did` to ensure that the request is intended for the correct device.
- 1606 • Verify that the `tsmid` matches the OWE identifier in the OWE-Cert to ensure that the OWE issuing  
1607 the request has access to the SD. See section 2.9 for details.
- 1608 • If `taid` is a UUID version 5, validate the proof of possession of the TA UUID as defined in  
1609 [TMF ASN.1] section 8.3.3.7. (See details in Annex D.)
- 1610 • Verify that the version of the TA to be updated is higher than the one that is currently installed. If the  
1611 update command does not contain the `encrypted_ta_bin` element, the `rSDTA` SHALL ignore the  
1612 `newtaver` element.
- 1613 • Use the SP-AIK-Priv key to decrypt TA binary and personalization data. If the version of the TA  
1614 associated with the TA binary is different than the `newtaver` element, the `rSDTA` SHALL abort the  
1615 update process.
- 1616 • Validate the TA signature using an SP-Cert associated with the SD. The TA signing mechanism may  
1617 be specific to the TEE OS.

1618 Upon successfully completing the above steps, the given TA SHALL be updated. Prior to an update, the TA  
1619 SHALL be forcefully shut down as defined in [TMF ASN.1] section 11. A response message  
1620 `UpdateTAResponse` SHALL always be returned regardless of the status of the operation.

## 1621 5.17 UpdateTATBSResponse

1622 In response to an UpdateTAResponse, the TEE SHALL return an UpdateTAResponse, encapsulating the  
1623 UpdateTATBSResponse message. The JSON structure for the UpdateTATBSResponse SHALL be as  
1624 follows:

```
1625 {  
1626     "UpdateTATBSResponse" : RESPONSE-PARAMETER-TYPE  
1627 }
```

1628 Within the RESPONSE-PARAMETER-TYPE, the following JSON structure is used as an input to the JWE while  
1629 generating CONTENT-ENCRYPTION-TYPE.

```
1630 {  
1631     "status" : "OPERATION-RESPONSE-PRIMITIVE-TYPE" ,  
1632     "did" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,  
1633     "dsi" : DSI-CONTENT-TYPE ,  
1634     "nextnonce" : "PRINTABLE-STRING-PRIMITIVE-TYPE"  
1635 }
```

1636 Where:

- 1637 • status: An OPERATION-RESPONSE-PRIMITIVE-TYPE indicating the status of the  
1638 UpdateTAResponse operation. If the TA is successfully updated, the value of status SHALL be  
1639 OPERATION\_SUCCESS; otherwise its value SHALL be an error string listed in section 5.17.1.
- 1640 • did: The value of did from previous messages.
- 1641 • dsi: The DSI-CONTENT-TYPE for the new device state. This element is returned only when the  
1642 nextdsi is set to true in the UpdateTAResponse.
- 1643 • nextnonce: A unique value that the OWE SHALL use as nonce in the next request. See  
1644 section 2.10.4 for details.

### 1645 5.17.1 Error Conditions

1646 If any validation listed in section 5.16.1 fails or if a TEE error occurs, the rSD<sub>TA</sub> SHALL use an appropriate  
1647 OPERATION-RESPONSE-PRIMITIVE-TYPE (listed below) as the status value in the corresponding  
1648 response message.

- 1649 • ERR\_DEV\_STATE\_MISMATCH
- 1650 • ERR\_INVALID\_UUID
- 1651 • ERR\_OCSP\_INVALID
- 1652 • ERR\_OWE\_NOT\_TRUSTED
- 1653 • ERR\_REQUEST\_INVALID
- 1654 • ERR\_SPCERT\_INVALID
- 1655 • ERR\_TA\_ALREADY\_INSTALLED
- 1656 • ERR\_TA\_INVALID
- 1657 • ERR\_TEE\_BUSY
- 1658 • ERR\_TEE\_FAIL
- 1659 • ERR\_TEE\_RESOURCE\_FULL
- 1660 • ERR\_TEE\_UNKNOWN
- 1661 • ERR\_UNSUPPORTED\_CRYPT\_ALG
- 1662 • ERR\_UNSUPPORTED\_MSG\_VERSION
- 1663 • ERR\_UPDATING\_DATA

1664 See section 4.14 for details on error strings.

1665 **Note:** If the OWE receives ERR\_REVERT\_OPERATION, it is recommended that the OWE remove the recently  
1666 created SD; otherwise the DSI value will be inconsistent.

## 1667 5.18 DeleteTATBSRequest

1668 An OWE SHALL issue a `DeleteTATBSRequest` message to delete a specific TA from a specified SD. The  
 1669 message SHALL be signed using the JWS scheme and encapsulated in a `DeleteTAResponse` message.  
 1670 This message SHALL always be sent to `rSDTA`. The JSON structure for the `DeleteTATBSRequest` is as  
 1671 follows:

```
1672 {
1673     "DeleteTATBSRequest" : COMMAND-PARAMETER-TYPE
1674 }
```

1675 Within the `COMMAND-PARAMETER-TYPE`, the following JSON structure SHALL be used as an input to the JWE  
 1676 while generating `CONTENT-ENCRYPTION-TYPE`.

```
1677 {
1678     "tsmid" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1679     "did" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1680     "sdid" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1681     "taid" : "PRINTABLE-STRING-PRIMITIVE-TYPE"
1682 }
```

1683 Where:

- 1684 • `tsmid`: The identifier of the OWE that issued the request.
- 1685 • `did`: The base64 encoded SHA-256 hash of the TEE-Cert binary. `did` is used as the device  
 1686 identifier to which the request is issued.
- 1687 • `sdid`: The base64 encoded UUID of the SD where the TA is installed.
- 1688 • `taid`: The base64 encoded UUID of a TA that is to be deleted.

### 1689 5.18.1 Processing Requirements

1690 Before authorizing the deletion of the SD, the `rSDTA` SHALL:

- 1691 • Validate the JSON web signature associated with the request.
- 1692 • Determine whether the OWE-Cert chains to a root CA certificate in OWE-Whitelist.
- 1693 • Check the revocation status of the OWE-Cert chain, using the cached OCSP stapling. If the cache is  
 1694 unavailable or expired, the `rSD` SHALL return the corresponding response with an error string along  
 1695 with an indication (`signerreq` set to `true`) to provide OCSP stapling in the next request. The OWE  
 1696 may reissue the request with OCSP stapling.
- 1697 • Compare the `dsihash` value to the SHA-256 hash of the internal `DSI-TYPE` to ensure that the DSI  
 1698 has not changed since the last changes requested by the OWE.
- 1699 • Compare `nonce` to the last `nextnonce` sent to the OWE to ensure that no new operation has been  
 1700 authorized on SDs and TAs associated with the OWE since the last operation requested by the OWE.
- 1701 • Decrypt the `ciphertext` element of the `CONTENT-ENCRYPTION-TYPE` to obtain the TA information  
 1702 for deletion.

- 1703       • Verify the `did` to ensure that the request is intended for the correct device.
- 1704       • Verify that the `tsmid` matches the OWE identifier in the OWE-Cert to ensure that the OWE issuing
- 1705       the request has access to the SD. See section 2.9 for details.
- 1706       Upon successfully completing the above requirements, the given TA SHALL be deleted. Prior to deletion, the
- 1707       TA SHALL be forcefully shut down as defined in [TMF ASN.1] section 11. A response message
- 1708       DeleteTAResponse SHALL always be returned regardless of the status of the operation.

## 1709 5.19 DeleteTATBSResponse

1710 In response to a DeleteTAResponse command, the rSD<sub>TA</sub> SHALL return a DeleteTAResponse,  
1711 encapsulating the DeleteTATBSResponse message. The JSON structure for the DeleteTATBSResponse  
1712 SHALL be as follows:

```
1713 {
1714     "DeleteTATBSResponse" : RESPONSE-PARAMETER-TYPE
1715 }
```

1716 Within the RESPONSE-PARAMETER-TYPE, the following JSON structure is used as an input to the JWE while  
1717 generating CONTENT-ENCRYPTION-TYPE.

```
1718 {
1719     "status" : "OPERATION-RESPONSE-PRIMITIVE-TYPE" ,
1720     "did" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1721     "dsi" : DSI-CONTENT-TYPE ,
1722     "nextnonce" : "PRINTABLE-STRING-PRIMITIVE-TYPE"
1723 }
```

1724 Where:

- 1725 • status: An OPERATION-RESPONSE-PRIMITIVE-TYPE indicating the status of the  
1726 DeleteSDRequest operation. If the TA is deleted successfully, the value of status SHALL be  
1727 OPERATION\_SUCCESS; otherwise its value SHALL be an error string listed in section 5.19.1.
- 1728 • did: The value of did from the DeleteSDRequest.
- 1729 • dsi: The DSI-CONTENT-TYPE for the new device state. This element is returned only when the  
1730 nextdsi is set to true in the DeleteSDRequest.
- 1731 • nextnonce: A unique value that the OWE SHALL use as nonce in the next request. See  
1732 section 2.10.4 for details.

### 1733 5.19.1 Error Conditions

1734 If any validation listed in section 5.18.1 fails or if a TEE error occurs, the rSD<sub>TA</sub> SHALL use an appropriate  
1735 OPERATION-RESPONSE-PRIMITIVE-TYPE (listed below) as the status value in the corresponding  
1736 response message.

- 1737 • ERR\_DEV\_STATE\_MISMATCH
- 1738 • ERR\_OCSP\_INVALID
- 1739 • ERR\_OWE\_NOT\_TRUSTED
- 1740 • ERR\_REQUEST\_INVALID
- 1741 • ERR\_TA\_NOT\_FOUND
- 1742 • ERR\_TEE\_BUSY
- 1743 • ERR\_TEE\_FAIL

- 1744      • ERR\_TEE\_RESOURCE\_FULL
- 1745      • ERR\_TEE\_UNKNOWN
- 1746      • ERR\_UNSUPPORTED\_CRYPTO\_ALG
- 1747      • ERR\_UNSUPPORTED\_MSG\_VERSION
- 1748      See section 4.14 for details on error strings.



## 1749 5.20 StoreTEEPPropertyTBSRequest

1750 An OWE SHALL issue a `StoreTEEPPropertyTBSRequest` message to store, update, or delete TEE  
 1751 properties. The message SHALL be signed using the JWS scheme and encapsulated in a  
 1752 `StoreTEEPPropertyRequest` message. This message SHALL always be issued to an `rSDTEE`.

1753 TEE properties are described in [TMF ASN.1] section A.5.

1754 The only property that can be updated is `gpd.tee.tmf.resetpreserved.entities`, which is used to  
 1755 indicate entities as UUIDs to be preserved across a Factory Reset operation on the TEE.

1756 The JSON structure for the `StoreTEEPPropertyTBSRequest` is as follows:

```
1757 {
1758     "StoreTEEPPropertyTBSRequest" : COMMAND-PARAMETER-TYPE
1759 }
```

1760 The following JSON elements will be used as input to the `CONTENT-ENCRYPTION-TYPE` within `COMMAND-`  
 1761 `TYPE`.

```
1762 {
1763     "tsmid" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1764     "did" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1765     "property" : "gpd.tee.tmf.resetpreserved.entities" ,
1766     "value" : {
1767         "taids" : UUID-ARRAY-TYPE ,
1768         "sdids" : UUID-ARRAY-TYPE
1769     }
1770 }
```

1771 Where:

- 1772 • `tsmid`: The identifier of the OWE that issued the request.
- 1773 • `did`: The base64 encoded SHA-256 hash of the TEE-Cert binary. `did` is used as the device  
 1774 identifier to which the request is issued.
- 1775 • `property`: OTrP Profile SHALL support only the TEE property  
 1776 `gpd.tee.tmf.resetpreserved.entities`.
- 1777 • `value`: The value of the TEE property.
- 1778 • `taids`: UUIDS of TAs structured as `UUID-ARRAY-TYPE` that SHALL be preserved across a Factory  
 1779 Reset operation on TEE.
- 1780 • `sdids`: UUIDS of SDs structured as `UUID-ARRAY-TYPE` that SHALL be preserved across a Factory  
 1781 Reset operation on TEE.

1782 The `StoreTEEPPropertyTBSRequest` SHALL always replace the previous value of the TEE property.

## 1783 5.20.1 Processing Requirements

1784 Upon receiving the `StoreTEEPPropertyRequest` message, the `rSDTEE` SHALL:

- 1785 • Validate the JSON web signature associated with the request, using the OWE-Pub associated with  
1786 the OWE-Cert.
- 1787 • Determine whether the OWE-Cert chains to a root CA certificate in its OWE-Whitelist.
- 1788 • Check the revocation status of the OWE-Cert chain, using the cached OCSP stapling. If the cache is  
1789 unavailable or expired, the `rSD` SHALL return the corresponding response with an error string along  
1790 with an indication (`signerreq` set to `true`) to provide OCSP stapling in the next request. The OWE  
1791 may reissue the request with OCSP stapling.

1792 Upon successfully completing the above requirements, the `rSDTEE` SHALL replace the TEE property with the  
1793 given value. A response message `StoreTEEPPropertyResponse` SHALL always be returned regardless of  
1794 the status of the operation.

## 1795 5.21 StoreTEEPPropertyTBSResponse

1796 In response to a StoreTEEPPropertyRequest command, the rSD<sub>TEE</sub> SHALL return a  
1797 StoreTEEPPropertyResponse, encapsulating the StoreTEEPPropertyTBSResponse message. The JSON  
1798 structure for the StoreTEEPPropertyTBSResponse is as follows:

```
1799 {  
1800     "StoreTEEPPropertyTBSResponse": RESPONSE-PARAMETER-TYPE  
1801 }
```

1802 Within the RESPONSE-PARAMETER-TYPE, the following JSON structure is used as an input to the JWE while  
1803 generating CONTENT-ENCRYPTION-TYPE.

```
1804 {  
1805     "status": "OPERATION-RESPONSE-PRIMITIVE-TYPE" ,  
1806     "did": "PRINTABLE-STRING-PRIMITIVE-TYPE" ,  
1807     "dsi": DSI-CONTENT-TYPE ,  
1808     "nextnonce": "PRINTABLE-STRING-PRIMITIVE-TYPE"  
1809 }
```

1810 Where:

- 1811 • status: An OPERATION-RESPONSE-PRIMITIVE-TYPE indicating the status of the  
1812 StoreTEEPPropertyRequest operation. If the TEE property is stored successfully, the value of  
1813 status SHALL be OPERATION\_SUCCESS; otherwise its value SHALL be an error string listed in  
1814 section 5.21.1.
- 1815 • did: The value of did from the StoreTEEPPropertyRequest.
- 1816 • dsi: The DSI-CONTENT-TYPE for the new device state. This element is returned only when the  
1817 nextdsi is set to true in the StoreTEEPPropertyRequest.
- 1818 • nextnonce: A unique value that the OWE SHALL use as nonce in the next request. See  
1819 section 2.10.4 for details.

### 1820 5.21.1 Error Conditions

1821 If any validation listed in section 5.20.1 fails or if a TEE error occurs, the rSD<sub>TEE</sub> SHALL use an appropriate  
1822 OPERATION-RESPONSE-PRIMITIVE-TYPE (listed below) as the status value in the corresponding  
1823 response message.

- 1824 • ERR\_OCSP\_INVALID
- 1825 • ERR\_OWE\_NOT\_TRUSTED
- 1826 • ERR\_REQUEST\_INVALID
- 1827 • ERR\_TEE\_BUSY
- 1828 • ERR\_TEE\_FAIL
- 1829 • ERR\_TEE\_RESOURCE\_FULL

- 1830      • ERR\_TEE\_UNKNOWN
- 1831      • ERR\_UNSUPPORTED\_CRYPTO\_ALG
- 1832      • ERR\_UNSUPPORTED\_MSG\_VERSION
- 1833      See section 4.14 for details on error strings.

## 1834 5.22 FactoryResetTBSRequest

1835 An OWE issues a `FactoryResetTBSRequest` message to move the TEE to a notional “factory” state. This  
1836 message SHALL be signed using the JWS scheme and encapsulated in a `FactoryResetRequest` message.  
1837 This message SHALL always be issued to an `rSDTEE`.

1838 The JSON structure for the `FactoryResetTBSRequest` is as follows:

```
1839 {  
1840     "FactoryResetTBSRequest" : COMMAND-PARAMETER-TYPE  
1841 }
```

1842 The following JSON element will be used as input to the `CONTENT-ENCRYPTION-TYPE` within `COMMAND-`  
1843 `TYPE`.

```
1844 {  
1845     "tsmid" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,  
1846     "did" : "PRINTABLE-STRING-PRIMITIVE-TYPE"  
1847 }
```

1848 Where:

- 1849 • `tsmid`: The identifier of the OWE that issued the request.
- 1850 • `did`: The base64 encoded SHA-256 hash of the TEE-Cert binary. `did` is used as the device  
1851 identifier to which the request is issued.

## 1852 5.22.1 Processing Requirements

1853 Upon receiving the `FactoryResetRequest` message, the `rSDTEE` SHALL:

- 1854 • Validate the JSON web signature associated with the request, using the OWE-Pub associated with  
1855 the OWE-Cert.
- 1856 • Determine whether the OWE-Cert chains to a root CA certificate in its OWE-Whitelist.
- 1857 • Check the revocation status of the OWE-Cert chain, using the cached OCSP stapling. If the cache is  
1858 unavailable or expired, the `rSD` SHALL return the corresponding response with an error string along  
1859 with an indication (`signerreq` set to `true`) to provide OCSP stapling in the next request. The OWE  
1860 may reissue the request with OCSP stapling.

1861 Upon successfully completing the above steps, the `rSDTEE` SHALL perform a factory reset on the device. All  
1862 SDs and TAs created or installed using OTrP Profile or [TMF ASN.1] that are not listed in  
1863 `gpd.tee.tmf.resetpreserved.entities` SHALL be removed. All TAs that are listed in  
1864 `gpd.tee.tmf.resetpreserved.entities` SHALL be reset as follows:

- 1865 • All active TEE Client or TEE Internal sessions are terminated. If the administration session used to  
1866 perform the Factory Reset operation is terminated, then the factory reset SHALL continue.
- 1867 • All data (if any) in the `TEE_STORAGE_PERSO` storage space is retained unmodified.
- 1868 • All data (if any) in the `TEE_STORAGE_PRIVATE` storage space is removed atomically.

1869 WARNING: Future TEE specifications may add new storage IDs that are not mentioned in this  
1870 document. Consult those specifications to determine how the new storage IDs react to factory reset.

1871 A response message `StoreTEEPPropertyResponse` SHALL always be returned regardless of the status of  
1872 the operation.

## 1873 5.23 FactoryResetTBSResponse

1874 In response to a `FactoryResetRequest` command, the `rSDTEE` SHALL return a `FactoryResetResponse`,  
1875 encapsulating the `FactoryResetTBSResponse` message. The JSON structure for the  
1876 `FactoryResetTBSResponse` is as follows:

```
1877 {  
1878     "FactoryResetTBSResponse" : RESPONSE-PARAMETER-TYPE  
1879 }
```

1880 Within the `RESPONSE-PARAMETER-TYPE`, the following JSON structure is used as an input to the JWE while  
1881 generating `CONTENT-ENCRYPTION-TYPE`.

```
1882 {  
1883     "status" : "OPERATION-RESPONSE-PRIMITIVE-TYPE" ,  
1884     "did" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,  
1885     "dsi" : DSI-CONTENT-TYPE ,  
1886     "nextnonce" : "PRINTABLE-STRING-PRIMITIVE-TYPE"  
1887 }
```

1888 Where:

- 1889 • `status`: An `OPERATION-RESPONSE-PRIMITIVE-TYPE` indicating the status of the  
1890 `FactoryResetRequest` operation. If the TEE property is stored successfully, the value of `status`  
1891 SHALL be `OPERATION_SUCCESS`; otherwise its value SHALL be an error string in section 5.23.1.
- 1892 • `did`: The value of `did` from the `FactoryResetRequest`.
- 1893 • `dsi`: The `DSI-CONTENT-TYPE` for the new device state. This element is returned only when the  
1894 `nextdsi` is set to `true` in the `FactoryResetRequest`.
- 1895 • `nextnonce`: A unique value that the OWE SHALL use as nonce in the next request. See  
1896 section 2.10.4 for details.

### 1897 **5.23.1 Error Conditions**

1898 If any validation listed in section 5.22.1 fails or if a TEE error occurs, the rSD<sub>TEE</sub> SHALL use an appropriate  
1899 OPERATION-RESPONSE-PRIMITIVE-TYPE (listed below) as the status value in the corresponding  
1900 response message.

- 1901 • ERR\_OCSP\_INVALID
- 1902 • ERR\_OWE\_NOT\_TRUSTED
- 1903 • ERR\_REQUEST\_INVALID
- 1904 • ERR\_TEE\_BUSY
- 1905 • ERR\_TEE\_FAIL
- 1906 • ERR\_TEE\_RESOURCE\_FULL
- 1907 • ERR\_TEE\_UNKNOWN
- 1908 • ERR\_UNSUPPORTED\_CRYPTO\_ALG
- 1909 • ERR\_UNSUPPORTED\_MSG\_VERSION

1910 See section 4.14 for details on error strings.

1911



## 1912 **Annex A Changes**

1913 This annex describes changes between the original Open Trust Protocol (OTrP) v1.0 ([OTPA OTrP]) and the  
1914 GlobalPlatform OTrP Profile described in this document.

### 1915 **A.1 Terminology**

1916 **Table A-1: Changes to Terminology**

<b>Original Terminology</b>	<b>Terminology Used</b>	<b>Notes</b>
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.

1917 **A.2 JSON Elements**1918 **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.

1919

## Annex B String Identifiers for Curves in ECC

1920

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

1922

**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	

## 1923 Annex C Specification Properties

1924 Most properties used in this specification can be retrieved by the generic Property Access Functions described  
1925 in [TEE Core], using the pseudo-handle specified in Table C-1.

1926 The `gpd.sd.isRootSD` property of an SD is flagged internally by the TEE at SD installation time and  
1927 SHOULD NOT be retrieved using the generic Property Access Functions.

1928 **Table C-1: Specification Reserved Properties**

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

1929 **Annex D Verification of UUID Version 5**

1930 Verification of the UUID version 5 proof of possession is defined in [TMF ASN.1] section 8.3.3.7 and  
 1931 section 5.6.1. The proof of possession is a signature calculated over the sequence of bytes resulting from the  
 1932 concatenation of the tag-length-value octets of the TA UUID and the TA Binary File.

1933 **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"

1934

1935 Where:

1936 L: The length of the TA binary encoded as defined in [TMF ASN.1] section 7.6.

1937 The proof of possession then has the following form:

1938  
1939**Table D-2: Example of Proof of Possession Encoding Values for InstallTARquest, UpdateTARquest**

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"

1940

1941

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

1942