

# GlobalPlatform Technology

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

### Version 0.0.0.21

---

**Public Review**

**November 2018**

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

**Copyright © 2017-2018 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>7</b>
1.1	Audience .....	7
1.2	IPR Disclaimer .....	7
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) .....	14
2.5	Outside World Entity (OWE) .....	14
2.6	Service Provider (SP).....	15
2.7	Trusted Firmware (TFW).....	15
2.8	OTrP Agent .....	15
2.9	OWE Certificate (OWE-CERT) .....	15
2.10	Security Model .....	16
2.10.1	Security Mechanism.....	16
2.10.2	Cryptographic Requirements .....	16
2.10.3	Cryptographic Recommendations.....	17
2.10.4	Nonce.....	17
2.10.5	Device State Information (DSI) .....	17
<b>3</b>	<b>Encoding OTrP Messages Using TEE Client API.....</b>	<b>18</b>
3.1	Reserved Command IDs.....	18
3.2	Encoding OTrP Messages .....	19
3.2.1	Handling Variable Length Return Values.....	19
3.2.2	Atomicity of Operations.....	19
3.2.3	Returning OTrP Errors .....	20
<b>4</b>	<b>JSON Message Formatting.....</b>	<b>21</b>
4.1	COMMAND-TYPE.....	21
4.2	UNPRIVILEGED-COMMAND-TYPE .....	22
4.3	COMMAND-PAYLOAD .....	22
4.4	PROTECTED-HEADER-TYPE .....	23
4.5	HEADER-TYPE.....	23
4.6	COMMAND-PARAMETER-TYPE .....	24
4.7	RESPONSE-PARAMETER-TYPE .....	25
4.8	CONTENT-ENCRYPTION-TYPE .....	26
4.9	KEYWRAP-INFO-TYPE .....	27
4.10	ENCRYPTION-PRIMITIVE-TYPE.....	28
4.11	SIGNATURE-PRIMITIVE-TYPE .....	29
4.12	KEYWRAP-PRIMITIVE-TYPE .....	30
4.13	CERT-PRIMITIVE-TYPE.....	30
4.14	OPERATION-RESPONSE-PRIMITIVE-TYPE.....	31
4.15	DSI-TYPE.....	33
4.16	DSI-CONTENT-TYPE .....	33
4.17	TRUSTED-FIRMWARE-TYPE.....	34
4.18	TEE-DESCRIPTION-TYPE.....	35

4.19	SD-DEFINITION-TYPE .....	36
4.20	TA-DEFINITION-TYPE .....	36
4.21	ISA-TYPE .....	37
4.22	TEE-PROPERTY-TYPE .....	37
4.23	TEE-AIK-TYPE .....	38
4.24	PUB-KEY-TYPE .....	39
4.25	OCSP-ARRAY-TYPE .....	40
4.26	UUID-ARRAY-TYPE .....	40
4.27	GPD-VERSION-TYPE .....	40
<b>5</b>	<b>OTrP Messages .....</b>	<b>41</b>
5.1	Unprivileged Messages .....	41
5.2	Privileged Messages .....	42
5.3	TEE Management Messages .....	43
5.4	GetTAInformationRequest .....	44
5.4.1	Processing Requirements .....	44
5.5	GetTAInformationResponse .....	45
5.5.1	Error Conditions .....	46
5.6	GetDeviceTEEStateTBSRequest .....	47
5.6.1	Processing Requirements .....	48
5.7	GetDeviceTEEStateTBSResponse .....	49
5.7.1	Error Conditions .....	50
5.8	CreateSDTBSRequest .....	51
5.8.1	Processing Requirements .....	52
5.9	CreateSDTBSResponse .....	53
5.9.1	Error Conditions .....	54
5.10	UpdateSDTBSRequest .....	55
5.10.1	Processing Requirements .....	56
5.11	UpdateSDTBSResponse .....	57
5.11.1	Error Conditions .....	58
5.12	DeleteSDTBSRequest .....	59
5.12.1	Processing Requirements .....	59
5.13	DeleteSDTBSResponse .....	61
5.13.1	Error Conditions .....	62
5.14	InstallTATBSRequest .....	63
5.14.1	Processing Requirements .....	64
5.15	InstallTATBSResponse .....	65
5.15.1	Error Conditions .....	66
5.16	UpdateTATBSRequest .....	67
5.16.1	Processing Requirements .....	68
5.17	UpdateTATBSResponse .....	69
5.17.1	Error Conditions .....	70
5.18	DeleteTATBSRequest .....	71
5.18.1	Processing Requirements .....	71
5.19	DeleteTATBSResponse .....	73
5.19.1	Error Conditions .....	74
5.20	StoreTEEPPropertyTBSRequest .....	75
5.20.1	Processing Requirements .....	76
5.21	StoreTEEPPropertyTBSResponse .....	77
5.21.1	Error Conditions .....	78
5.22	FactoryResetTBSRequest .....	79
5.22.1	Processing Requirements .....	80
5.23	FactoryResetTBSResponse .....	81

5.23.1	Error Conditions .....	82
<b>Annex A</b>	<b>Changes .....</b>	<b>83</b>
A.1	Terminology .....	83
A.2	JSON Elements.....	83
<b>Annex B</b>	<b>String Identifiers for Curves in ECC.....</b>	<b>85</b>
<b>Annex C</b>	<b>Specification Properties.....</b>	<b>86</b>

## Tables

Table 1-1: Normative References.....	7
Table 1-2: Informative References .....	8
Table 1-3: Terminology and Definitions.....	9
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.....	18
Table 3-2: Envelope Command Encoding.....	19
Table 4-1: Examples of base64url encoded ENCRYPTION-PRIMITIVE-TYPE .....	28
Table 4-2: Example [TEE Core] Algorithms to Support ENCRYPTION-PRIMITIVE-TYPE .....	28
Table 4-3: Examples of base64url encoded SIGNATURE-PRIMITIVE-TYPE.....	29
Table 4-4: Example [TEE Core] Algorithms to Support SIGNATURE-PRIMITIVE-TYPE.....	29
Table 4-5: Example [TEE Core] Algorithms to Support KEYWRAP-PRIMITIVE-TYPE.....	30
Table 4-6: Internal API Names Strings Definition.....	35
Table 5-1: Request/Response String Naming.....	41
Table A-1: Changes to Terminology.....	83
Table A-2: Changes to JSON Elements.....	83
Table B-1: String Identifiers for Curves in ECC.....	85
Table C-1: Specification Reserved Properties .....	86

## Figures

Figure 2-1: OTrP Architecture .....	12
Figure 3-1: Single Envelope Command .....	19

# 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, the mechanisms involved in administration operations. In addition, [TMF ASN.1] defines an ASN.1 profile for TMF.

This document specifies a security mechanism that can be used in the context of TMF for the realization of the Open Trust Protocol (OTrP) Profile and specifies 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.

## 1.1 Audience

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

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

## 1.2 IPR Disclaimer

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

## 1.3 References

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

**Table 1-1: Normative References**

Standard / Specification	Description	Ref
GPD_SPE_010	GlobalPlatform Technology TEE Internal Core API Specification	[TEE Core]
GPD_SPE_120	GlobalPlatform Technology TEE Management Framework (including ASN.1 Profile) [Initially published as TEE Management Framework]	[TMF ASN.1]

Standard / Specification	Description	Ref
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

34

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]

35

## 36 1.4 Terminology and Definitions

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

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



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

44 **Table 1-3: Terminology and Definitions**

Term	Definition
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.)
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]).
Rich Execution Environment (REE)	An environment that is provided and governed by a Rich OS, potentially in conjunction with other supporting operating systems and hypervisors; it is outside of the TEE. This environment and applications running on it are considered untrusted. Contrast <i>Trusted Execution Environment (TEE)</i> .
Service Provider (SP)	An entity that issues TAs. (For more information, see section 2.6.)
Session	Logically connects multiple commands invoked on a Trusted Application or a Security Domain. In the context of this specification, logically connects an OWE to a TEE on a device. Begins with a <code>GetDeviceTEEStateRequest</code> .
spid	A unique value that identifies a Service Provider.
Trusted Application (TA)	An application running inside the Trusted Execution Environment (TEE) that provides security related functionality to Client Applications outside of the TEE or to other Trusted Applications inside the TEE. Contrast <i>Client Application</i> .
Trusted Execution Environment (TEE)	An execution environment that runs alongside but isolated from an REE. A TEE has security capabilities and meets certain security related requirements: It protects TEE assets from general software attacks, defines rigid safeguards as to data and functions that a program can access, and resists a set of defined threats. There are multiple technologies that can be used to implement a TEE, and the level of security achieved varies accordingly. Contrast <i>Rich Execution Environment (REE)</i> .
Trusted OS	An operating system running in the TEE providing the TEE Internal Core API to Trusted Applications.

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.
tsmid	A unique value that identifies an OWE.

45

## 46 1.5 Abbreviations and Notations

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

48 Additional abbreviations and notations are included in Table 1-4 and Table 2-1.

49 **Table 1-4: Abbreviations and Notations**

Abbreviation / Notation	Meaning
CBC	Cipher Block Chaining
DSI	Device State Information
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
rSD	Root Security Domain
SD	Security Domain
SP	Service Provider
TFW	Trusted Firmware
TMF	TEE Management Framework
TSM	Trusted Service Manager

50

51 **1.6 Revision History**52 **Table 1-5: Revision History**

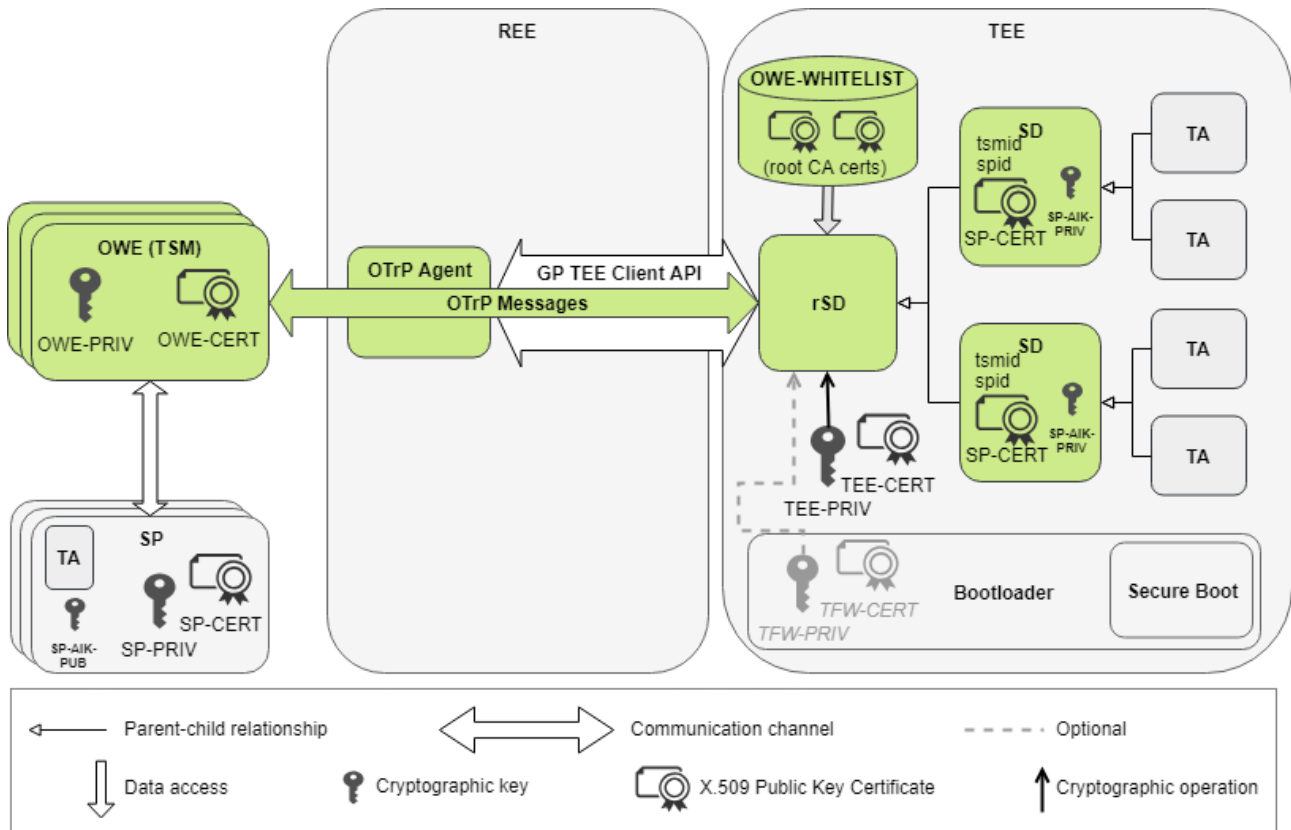
Date	Version	Description
September 2017	0.1	First Working Draft
August 2018	0.0.0.9	Committee Review
October 2018	0.0.0.15	Member Review
November 2018	0.0.0.21	Public Review
TBD	TBD	Public Release

53

## 54 2 OTrP Overview

### 55 2.1 Architecture

56 Figure 2-1: OTrP Architecture



58

59

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

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

75 **2.2 Nomenclature**

76 **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.
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, which 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 only accessible 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 only accessible to the device firmware. The device firmware uses this key to sign data to attest its validity to a remote entity.
TFW-PUB	The public portion of a key pair burned into the device and accessible by the device firmware.

Term	Definition
tsmid	A unique value that identifies an OWE. Its value SHALL be the OWE identifier present in the OWE-CERT.

77

## 78 2.3 Root Security Domain (rSD)

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

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

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

94

## 95 2.4 Security Domain (SD)

96 [TMF ASN.1] section 4.1 defines the concept of Security Domain (SD). In this document, an SD refers to an  
 97 SD created using OTrP messages. An SD SHALL only have TA Management and TA Personalization  
 98 privileges. SDs SHALL be uniquely identified using UUIDs. A UUID for an SD may be derived from the tsmid  
 99 (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 • Calculate the SHA-1 Hash of {tsmid || spid}.
- 103 • Use the left most 128 bits as UUID.

104 UUIDs derived as above SHALL be set either as UUID version 1 or UUID version 4 as specified in [RFC 4122].

105

## 106 2.5 Outside World Entity (OWE)

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

111

## 112 2.6 Service Provider (SP)

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

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

118

## 119 2.7 Trusted Firmware (TFW)

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

124

## 125 2.8 OTrP Agent

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

131

## 132 2.9 OWE Certificate (OWE-CERT)

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

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

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

143

## 144 2.10 Security Model

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

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

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

155

### 156 2.10.1 Security Mechanism

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

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

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

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

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

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

172

### 173 2.10.2 Cryptographic Requirements

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

178



### 179 **2.10.3 Cryptographic Recommendations**

180 OTrP Profile SHOULD use the following cryptographic recommendations.

- 181 • Symmetric cryptography: Minimum equivalent to AES with 128-bit keys.
- 182 • Hash functions: Minimum equivalent to SHA-256.
- 183 • Asymmetric cryptography: Minimum equivalent to RSA with key size of at least 3096 bits. However, it  
184 is recommended to use Elliptic Curve Cryptography (ECC) with P-256. Other curve values may be  
185 used. See Table B-1 for string identifiers of these curves.
- 186 • Key management: It is recommended to use Elliptic Curve Diffie–Hellman (ECDH) with key size of at  
187 least 256 bits for key agreement / management.
- 188 • RSA-based JWE and JWS SHOULD use separate key pairs for signing and encryption.

189

### 190 **2.10.4 Nonce**

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

201

### 202 **2.10.5 Device State Information (DSI)**

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

209

## 210 **3 Encoding OTrP Messages Using TEE Client API**

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

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

216

### 217 **3.1 Reserved Command IDs**

218 When TEEC\_InvokeCommand is called to send OTrP messages to a Security Domain, the following  
219 [TEE Client] Command IDs are reserved.

220

**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
0x40000000 – 0xFFFFFFFF	Implementation defined

221

222 **3.2 Encoding OTrP Messages**

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

224 This command uses a single envelope command with two parameters:

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

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



233  
234

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

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

236

237 **3.2.1 Handling Variable Length Return Values**

238 For handling variable length return values, refer to [TEE Core] section 3.4.4.

239

240 **3.2.2 Atomicity of Operations**

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

245

### 246 3.2.3 Returning OTrP Errors

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

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

253

## 254 4 JSON Message Formatting

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

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

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

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

266

### 267 4.1 COMMAND-TYPE

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

270

```
271 {  
272     "payload" : COMMAND-PAYLOAD ,  
273     "protected" : PROTECTED-HEADER-TYPE ,  
274     "header" : HEADER-TYPE ,  
275     "signature" : "PRINTABLE-STRING-PRIMITIVE-TYPE"  
276 }
```

277 Where:

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

283

## 284 4.2 UNPRIVILEGED-COMMAND-TYPE

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

286

287 GET-TA-INFORMATION-REQUEST

288 GET-TA-INFORMATION-RESPONSE

289

## 290 4.3 COMMAND-PAYLOAD

291 COMMAND-PAYLOAD SHALL be the base64url encoding of:

292 COMMAND-TBS

293 Where:

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

295 GET-DEVICE-TEE-STATE-TBS-REQUEST

296 GET-DEVICE-TEE-STATE-TBS-RESPONSE

297 CREATE-SD-TBS-REQUEST

298 CREATE-SD-TBS-RESPONSE

299 UPDATE-SD-TBS-REQUEST

300 UPDATE-SD-TBS-RESPONSE

301 DELETE-SD-TBS-REQUEST

302 DELETE-SD-TBS-RESPONSE

303 INSTALL-TA-TBS-REQUEST

304 INSTALL-TA-TBS-RESPONSE

305 UPDATE-TA-TBS-REQUEST

306 UPDATE-TA-TBS-RESPONSE

307 DELETE-TA-TBS-REQUEST

308 DELETE-TA-TBS-RESPONSE

309 STORE-TEE-PROPERTY-TBS-REQUEST

310 STORE-TEE-PROPERTY-TBS-RESPONSE

311 FACTORY-RESET-TBS-REQUEST

312 FACTORY-RESET-TBS-RESPONSE

313

## 314 4.4 PROTECTED-HEADER-TYPE

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

```
317 {  
318     "alg" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,  
319     "rSD" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,  
320     "tee" : "PRINTABLE-STRING-PRIMITIVE-TYPE"  
321 }
```

322 Where:

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

333

## 334 4.5 HEADER-TYPE

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

```
336 {  
337     "x5c" : [ "CERT-PRIMITIVE-TYPE" ] ,  
338     "kid" : "PRINTABLE-STRING-PRIMITIVE-TYPE"  
339 }
```

340 Where:

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

344

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

348

## 349 4.6 COMMAND-PARAMETER-TYPE

350 OTrP request messages SHALL have the following common elements:

```

351 {
352     "ver" : "GPD-VERSION-TYPE" ,
353     "tid" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
354     "rid" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
355     "tee" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
356     "nextdsi" : BOOLEAN ,
357     "dsihash" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
358     "nonce" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
359     "content" : CONTENT-ENCRYPTION-TYPE
360 }
```

361 Where:

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

376

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

379



## 380 4.7 RESPONSE-PARAMETER-TYPE

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

```
382 {  
383     "ver": "GPD-VERSION-TYPE" ,  
384     "rid": "PRINTABLE-STRING-PRIMITIVE-TYPE" ,  
385     "tid": "PRINTABLE-STRING-PRIMITIVE-TYPE" ,  
386     "content": CONTENT-ENCRYPTION-TYPE  
387 }
```

388 Where:

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

395

## 396 4.8 CONTENT-ENCRYPTION-TYPE

397 The CONTENT-ENCRYPTION-TYPE is a JSON structure for encrypted data in OTrP messages. CONTENT-  
398 ENCRYPTION-TYPE uses JWE for encrypting data and follows the Flattened JWE JSON Serialization Syntax.  
399 Symmetric keys known as Content Encryption Keys (CEK) are used to encrypt the data. When using RSA, the  
400 CEK and authentication HMAC key are encrypted or wrapped by a recipient's public asymmetric key (OWE-  
401 PUB or TEE-PUB). It is recommended that the recipient's key used for wrapping the CEK is different from the  
402 recipient's signature key. For ECDH the CEK is agreed using the recipient's public key (OWE-PUB or TEE-  
403 PUB) and an ephemeral key is generated by the sender. OTrP Profile does not use JWE AAD (Additional  
404 Authenticated Data) as every message is signed after encryption.

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

```
406 {  
407     "protected": "ENCRYPTION-PRIMITIVE-TYPE" ,  
408     "recipients": [KEYWRAP-INFO-TYPE] ,  
409     "iv": "PRINTABLE-STRING-PRIMITIVE-TYPE" ,  
410     "ciphertext": "PRINTABLE-STRING-PRIMITIVE-TYPE" ,  
411     "tag": "PRINTABLE-STRING-PRIMITIVE-TYPE"  
412 }
```

413 Where:

- 414 • **protected**: A mandatory JWE header parameter that indicates the cryptographic algorithm used for  
415 encryption.
- 416 • **recipients**: An array of KEYWRAP-INFO-TYPE each containing information specific to 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

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

425

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

```
427 {
428     "header": {
429         "alg": "KEYWRAP-PRIMITIVE-TYPE"
430     },
431     "encrypted_key": "PRINTABLE-STRING-PRIMITIVE-TYPE"
432 }
```

433 Where:

- 434 • header: A mandatory header that contains alg element.
- 435 • alg: The KEYWRAP-PRIMITIVE-TYPE value that indicates the algorithm from JSON Web Algorithms ([RFC 7518]) used to encrypt CEK.
- 437 • encrypted\_key: The base64url encoding value of the JWE encrypted CEK.

438

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

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

449

Where header, alg, 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 PartyUInfo value for key agreement algorithm.
- 453 • apv: The base64url encoded agreement PartyVInfo value for key agreement algorithm.

454

## 4.10 ENCRYPTION-PRIMITIVE-TYPE

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

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

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

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

Where:

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

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

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

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

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

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

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

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

**Note:** Refer to section 2.10 for additional information.

482 **4.11 SIGNATURE-PRIMITIVE-TYPE**

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

487 The following JSON structures are examples of SIGNATURE-PRIMITIVE-TYPE defined using RSA and EC  
 488 algorithms with key size of 256 bits.

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

491 Where:

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

495

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

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

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

498

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

502 **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

503

504 **Note:** Refer to section 2.10 for additional information.

505

## 506 4.12 KEYWRAP-PRIMITIVE-TYPE

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

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

513	RSA1_5
514	ECDH-ES+A128KW
515	ECDH-ES+A256KW

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

519 **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

520

521 **Note:** Refer to section 2.10 for additional information.

522

## 523 4.13 CERT-PRIMITIVE-TYPE

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

525

## 526 4.14 OPERATION-RESPONSE-PRIMITIVE-TYPE

527 One of the following strings:

```
528 OPERATION_SUCCESS
529 ERR_DEV_STATE_MISMATCH
530 ERR_SD_NOT_EMPTY
531 ERR_SDID_ALREADY_USED
532 ERR_REQUEST_INVALID
533 ERR_SPCERT_INVALID
534 ERR_TA_ALREADY_INSTALLED
535 ERR_TA_INVALID
536 ERR_TA_NOT_FOUND
537 ERR_TEE_BUSY
538 ERR_TEE_FAIL
539 ERR_TEE_RESOURCE_FULL
540 ERR_TEE_UNKNOWN
541 ERR_TFW_NOT_TRUSTED
542 ERR_OWE_NOT_TRUSTED
543 ERR_OCSP_INVALID
544 ERR_UNSUPPORTED_CRYPTO_ALG
545 ERR_UNSUPPORTED_MSG_VERSION
```

546

547 Where the values have the following meanings:

- 548 • OPERATION\_SUCCESS: Returned when the corresponding request message has been processed  
549 successfully.
- 550 • ERR\_DEV\_STATE\_MISMATCH: Returned when the DSI hash value from OWE doesn't match with that  
551 of device's current DSI.
- 552 • ERR\_SD\_NOT\_EMPTY: Returned when an OWE tries to delete an SD that contains one or more TAs.
- 553 • ERR\_SDID\_ALREADY\_USED: Returned when an OWE requests creation of an SD with a UUID that  
554 already exists in the namespace of the OWE in the TEE.
- 555 • ERR\_REQUEST\_INVALID: Returned when any of the following conditions occurs:
  - 556 ○ Request message is not supported by rSD.
  - 557 ○ Request message has an invalid message structure; e.g. mandatory element is absent, or  
558 undefined elements or structures are included.
  - 559 ○ Failure to verify message signature.
  - 560 ○ Failure to decrypt CONTENT-ENCRYPTION-TYPE value.

- 561           ○ Insufficient privilege to perform an operation (e.g. deletion of a TA from an SD that the OWE is not  
562           allowed to access).
- 563           • ERR\_SPCERT\_INVALID: Returned when the new SP-CERT provided while updating an SD is not  
564           valid.
- 565           • ERR\_TA\_ALREADY\_INSTALLED: Returned when an OWE requests installation of a TA with a given  
566           UUID and a version that already exists.
- 567           • ERR\_TA\_INVALID: Returned when any of the following conditions occurs while checking validity of a  
568           TA:
  - 569           ○ TA binary has a format that TEE doesn't recognize.
  - 570           ○ TEE fails to decrypt the encoding of TA binary and personalization data.
  - 571           ○ If the SP isn't registered with the SD where a TA is to be installed.
  - 572           ○ During an update, if the version of the TA is lower than the current version installed.
  - 573           ○ If the TA version information provided in the request message is different than the TA version  
574           associated with the TA binary.
- 575           • ERR\_TA\_NOT\_FOUND: Returned when the target TA doesn't exist in the SD.
- 576           • ERR\_TEE\_BUSY: Returned when the device TEE is currently busy.
- 577           • ERR\_TEE\_FAIL: Returned when any of the following conditions occurs:
  - 578           ○ TEE fails to respond to an OWE request. The OTrP Agent will construct an error message in  
579           responding to the OWE's request.
  - 580           ○ TEE fails to process a request because of its internal error.
- 581           • ERR\_TEE\_RESOURCE\_FULL: Returned when a device resource is no longer available, such as  
582           storage space is full.
- 583           • ERR\_TEE\_UNKNOWN: Returned when the TEE is not supposed to receive the request, as determined  
584           by checking the TEE name or device identifier (did) in the request message.
- 585           • ERR\_TFW\_NOT\_TRUSTED: Returned when the TEE determines that the underlying device firmware is  
586           not trustworthy.
- 587           • ERR\_OWE\_NOT\_TRUSTED: Returned when the OWE-CERT chain cannot be validated using the root  
588           CA certificate in the OWE-WHITELIST while processing a request message.
- 589           • ERR\_OCSP\_INVALID: Returned when the OCSP stapling is either invalid, not available, or expired.
- 590           • ERR\_UNSUPPORTED\_CRYPTO\_ALG: Returned when a request message contains CONTENT-  
591           ENCRYPTION-TYPE value encrypted with a cryptographic algorithm that the TEE doesn't support.
- 592           • ERR\_UNSUPPORTED\_MSG\_VERSION: Returned when the OTrP version of the request message is not  
593           supported by the TEE.



## 594 4.15 DSI-TYPE

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

```
597 {  
598     "dsi" : DSI-CONTENT-TYPE  
599 }
```

600 Where:

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

602

## 603 4.16 DSI-CONTENT-TYPE

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

```
605 {  
606     "tfwdata" : TRUSTED-FIRMWARE-TYPE ,  
607     "tee" : TEE-DESCRIPTION-TYPE  
608 }
```

609 Where:

- 610 • tfwdata: (OPTIONAL) The trusted firmware information structured as TRUSTED-FIRMWARE-TYPE.
- 611 • tee: The underlying trusted execution environment information structured as TEE-DESCRIPTION-  
612 TYPE.

613

## 614 4.17 TRUSTED-FIRMWARE-TYPE

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

```
617 {  
618     "payload": "PRINTABLE-STRING-PRIMITIVE-TYPE",  
619     "protected": PROTECTED-HEADER-TYPE,  
620     "header": HEADER-TYPE,  
621     "signature": "PRINTABLE-STRING-PRIMITIVE-TYPE"  
622 }
```

623 Where:

- 624 • payload: The string representing a challenge that the TFW SHALL sign. The tid value from the  
625 corresponding GetDeviceTEESStateRequest is used as the challenge.
- 626 • protected: The JWS protected header element structured as PROTECTED-HEADER-TYPE. The  
627 PROTECTED-HEADER-TYPE SHALL only include the "alg" element that indicates the cryptographic  
628 algorithm used to sign the payload.
- 629 • header: The JWS header element structured as HEADER-TYPE. The x5c element of the HEADER-  
630 TYPE SHALL contain the TFW-CERT represented as CERT-PRIMITIVE-TYPE. Implementation  
631 property gpd.tee.firmware.implementation.binaryversion as defined in [TEE Core]  
632 section 4.7 may be use to extract TFW-CERT from the TFW.
- 633 • signature: The base64url encoded signature calculated according to the JWS scheme.

634

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

637

638 **4.18 TEE-DESCRIPTION-TYPE**

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

```

640 {
641     "name" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
642     "teever" : "GPD-VERSION-TYPE" ,
643     "cert" : "CERT-PRIMITIVE-TYPE" ,
644     "cacert" : [ "CERT-PRIMITIVE-TYPE" ] ,
645     "sdlist" : [ SD-DEFINITION-TYPE ] ,
646     "teeaiklist" : [ TEE-AIK-TYPE ] ,
647     "isaset" : ISA-TYPE ,
648     "teeImplementationProperty" : [ TEE-PROPERTY-TYPE ]
649 }
    
```

650 Where:

- 651 • name: A zero-terminated string that describes the TEE to connect to. Its value matches the  
652 parameter name used to connect to a TEE while initializing a context using the  
653 TEEC\_InitializeContext. For detail, refer to [TEE Client] section 4.5.2.
- 654 • teever: The TEE version structured as GPD-VERSION-TYPE.
- 655 • cert: The TEE-CERT represented as CERT-PRIMITIVE-TYPE.
- 656 • cacert: The X.509 certificate chain starting with the CA certificate that issued the TEE-CERT up to  
657 the root CA certificate structured as the CERT-PRIMITIVE-TYPE array.
- 658 • sdlist: An array of SD-DEFINITION-TYPE, where each element of the array provides the  
659 metadata of an SD that a given OWE has access to. This element SHALL be excluded if the rSD that  
660 prepares this JSON object is an rSD<sub>TEE</sub>.
- 661 • teeaiklist: An array of TEE-AIK-TYPE, where each element of the array provides information  
662 related to an SP-AIK. This element SHALL be excluded if the rSD that prepares this JSON object is an  
663 rSD<sub>TEE</sub>.
- 664 • isaset: (OPTIONAL) Instruction set and architecture definition based on ISA Type defined in  
665 [TMF ASN.1] section 9.14.
- 666 • teeImplementationProperty: (OPTIONAL) Lists the TEE properties. For more information on  
667 TEE properties, refer to [TMF ASN.1] section A.5. This element SHALL only be included if the rSD that  
668 prepares this JSON structure is an rSD<sub>TEE</sub>. However, if the TEE has TMF ASN.1 audit SD capabilities,  
669 then OTrP SHALL provide the following valid API name string to be used with the optionalApis  
670 attribute of TEE Type, defined in [TMF ASN.1] section 9.1.6.

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

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

672

## 673 4.19 SD-DEFINITION-TYPE

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

```
675 {  
676     "sdid" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,  
677     "spid" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,  
678     "protocol" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,  
679     "talist" : [TA-DEFINITION-TYPE ]  
680 }
```

681 Where:

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

## 690 4.20 TA-DEFINITION-TYPE

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

```
692 {  
693     "taid" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,  
694     "taver" : "PRINTABLE-STRING-PRIMITIVE-TYPE "  
695 }
```

696 Where:

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

## 700 4.21 ISA-TYPE

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

```
703 {  
704     "isaName" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,  
705     "processorType" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,  
706     "instructionSet" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,  
707     "addressSize" : INTEGER-PRIMITIVE-TYPE ,  
708     "abi" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,  
709     "endianness" : INTEGER-PRIMITIVE-TYPE  
710 }
```

711 Where:

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

718

## 719 4.22 TEE-PROPERTY-TYPE

720 A JSON structure that provides the TEE property information. TEE properties are detailed in [TMF ASN.1]  
721 section A.5. OTrP Profile SHALL use only the TEE property `gpd.tee.tmf.resetpreserved.entities`.

```
722 {  
723     "PROPERTY-NAME" : PROPERTY-VALUE  
724 }
```

725 Where:

- 726 • **PROPERTY-NAME**: A string that identifies the TEE property as described in [TMF ASN.1] section A.5.
- 727 • **PROPERTY-VALUE**: The value of the TEE property. OTrP Profile passes only the TEE property  
728 `gpd.tee.tmf.resetpreserved.entities`. Therefore, the **PROPERTY-VALUE** is the  
729 **UUID-ARRAY-TYPE**.

730

## 731 4.23 TEE-AIK-TYPE

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

```
733 {  
734     "spaik" : PUB-KEY-TYPE ,  
735     "spid" : "PRINTABLE-STRING-PRIMITIVE-TYPE"  
736 }  
737
```

738 Where:

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

741

## 742 4.24 PUB-KEY-TYPE

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

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

```
745 {  
746     "kty": "RSA",  
747     "n": "PRINTABLE-STRING-PRIMITIVE-TYPE",  
748     "e": "PRINTABLE-STRING-PRIMITIVE-TYPE"  
749 }
```

750 Where:

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

755

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

```
757 {  
758     "kty": "EC",  
759     "crv": "PRINTABLE-STRING-PRIMITIVE-TYPE",  
760     "x": "PRINTABLE-STRING-PRIMITIVE-TYPE",  
761     "y": "PRINTABLE-STRING-PRIMITIVE-TYPE"  
762 }
```

763 Where:

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

768

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

771

## 772 4.25 OCSP-ARRAY-TYPE

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

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

776

## 777 4.26 UUID-ARRAY-TYPE

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

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

781

## 782 4.27 GPD-VERSION-TYPE

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

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

785 Where:

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

791

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

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



798

## 5 OTrP Messages

799

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:

801

802

**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

803

804

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

805

806

807

### 5.1 Unprivileged Messages

808

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

809

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

810

811

812

Where:

813

- NAME SHALL be a choice of one of the following strings:

814

GetTAInformationRequest

815

GetTAInformationResponse

816

## 817 5.2 Privileged Messages

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

```
820 {
821     "NAME" : COMMAND-TYPE
822 }
```

823 Where:

824 • NAME SHALL be a choice of one of the following strings:

```
825     GetDeviceTEEStateRequest
826     GetDeviceTEEStateResponse
827     CreateSDRequest
828     CreateSDResponse
829     UpdateSDRequest
830     UpdateSDResponse
831     DeleteSDRequest
832     DeleteSDResponse
833     InstallTARRequest
834     InstallTARResponse
835     UpdateTARRequest
836     UpdateTARResponse
837     DeleteTARRequest
838     DeleteTARResponse
```

839 • COMMAND-TYPE: contains the corresponding signed message.

840

841 A privileged message SHALL be created as follows:

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

848 Only OTrP request messages SHALL include the OWE-CERT as a part of the HEADER-TYPE element in the  
849 COMMAND-TYPE. The public key associated with the OWE-CERT SHALL be used to verify the signature. OTrP  
850 response SHALL NOT include its TEE-CERT as a part of the HEADER-TYPE element but SHALL include its  
851 TEE-CERT as a part of the GET-DEVICE-TEE-STATE-RESPONSE message for privacy reasons.

852

853 

### 5.3 TEE Management Messages

854 TEE management messages are a set of **optional** OTrP messages that is intended for managing TEEs. Only  
855 the OWE whose OWE-CERT chains to the root CA certificate in the OWE-WHITELIST of the rSD<sub>TEE</sub> SHALL  
856 be allowed to issue TEE management messages. The TEE management messages SHALL always be sent  
857 to and processed by an rSD<sub>TEE</sub>. Prior to sending TEE management messages, the OWE SHALL initiate an  
858 OTrP session with rSD<sub>TEE</sub> by sending a `GetDeviceTEEStateRequest` message.

859 Similarly to privileged messages, TEE management messages SHALL be formatted as follows:

```
860 {  
861     "NAME" : COMMAND-TYPE  
862 }
```

863 Where:

- 864 • NAME SHALL be a choice of one of the following strings:

```
865     StoreTEEPropertyRequest  
866     StoreTEEPropertyResponse  
867     FactoryResetRequest  
868     FactoryResetResponse
```

- 869 • COMMAND-TYPE: contains the corresponding signed message.

870 TEE management messages SHALL be created in a similar fashion as the privileged messages.

871

## 872 5.4 GetTAInformationRequest

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

```
876 {  
877     "GetTAInformationRequest": {  
878         "ver": "GPD-VERSION-TYPE",  
879         "taid": "PRINTABLE-STRING-PRIMITIVE",  
880         "spid": "PRINTABLE-STRING-PRIMITIVE"  
881     }  
882 }
```

883 Where:

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

887

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

### 891 5.4.1 Processing Requirements

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

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

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

## 898 5.5 GetTAInformationResponse

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

902

```
903 {  
904     "GetTAInformationResponse": {  
905         "ver": "GPD-VERSION-TYPE",  
906         "status": "OPERATION-RESPONSE-PRIMITIVE-TYPE",  
907         "taid": "PRINTABLE-STRING-PRIMITIVE-TYPE",  
908         "taver": "PRINTABLE-STRING-PRIMITIVE-TYPE",  
909         "sdid": "PRINTABLE-STRING-PRIMITIVE-TYPE",  
910         "spid": "PRINTABLE-STRING-PRIMITIVE-TYPE",  
911         "tsmid": "PRINTABLE-STRING-PRIMITIVE-TYPE"  
912     }  
913 }
```

914 Where:

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

## 928 5.5.1 Error Conditions

929 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  
930 OPERATION-RESPONSE-PRIMITIVE-TYPE (listed below) as the status value in the corresponding  
931 response message.

- 932 • ERR\_TA\_NOT\_FOUND
- 933 • ERR\_TEE\_BUSY
- 934 • ERR\_TEE\_FAIL
- 935 • ERR\_TEE\_RESOURCE\_FULL
- 936 • ERR\_TEE\_UNKNOWN
- 937 • ERR\_UNSUPPORTED\_MSG\_VERSION

938 See section 4.14 for details on error strings.

939

## 940 5.6 GetDeviceTEEStateTBSRequest

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

```
945 {  
946     "GetDeviceTEEStateTBSRequest": {  
947         "ver": "GPD-VERSION-TYPE",  
948         "tid": "PRINTABLE-STRING-PRIMITIVE-TYPE",  
949         "rid": "PRINTABLE-STRING-PRIMITIVE-TYPE",  
950         "ocspdats": OSCP-ARRAY-TYPE,  
951         "supportedsigalgs": [SIGNATURE-PRIMITIVE-TYPE]  
952     }  
953 }
```

954 Where:

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

964

## 965 5.6.1 Processing Requirements

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

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

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

979



## 980 5.7 GetDeviceTEEStateTBSResponse

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

```
984 {  
985     "GetDeviceTEEStateTBSResponse": {  
986         "ver": "GPD-VERSION-TYPE",  
987         "status": "OPERATION-RESPONSE-PRIMITIVE-TYPE",  
988         "rid": "PRINTABLE-STRING-PRIMITIVE-TYPE",  
989         "tid": "PRINTABLE-STRING-PRIMITIVE-TYPE",  
990         "signerreq": BOOLEAN,  
991         "content": CONTENT-ENCRYPTION-TYPE  
992     }  
993 }
```

994 Where:

- 995 • `ver`: The version of the OTrP message structured as `GPD-VERSION-TYPE`.
- 996 • `status`: An `OPERATION-RESPONSE-PRIMITIVE-TYPE` indicating the status of the  
997 `GetDeviceTEEStateRequest` operation. If successful, the value of `status` SHALL be  
998 `OPERATION_SUCCESS`; otherwise its value SHALL be an error string listed in section 5.7.1.
- 999 • `rid`: A unique value identifying the `GetDeviceTEEStateRequest` message.
- 1000 • `tid`: A unique value identifying the OTrP session. Matches the `tid` value in  
1001 `GetDeviceTEEStateRequest` message.
- 1002 • `signerreq`: A Boolean value that indicates whether the OWE should send its signer certificate and  
1003 OCSP stapling again in the subsequent messages. It is recommended that the `signerreq` value is  
1004 set to `false`. If the value is set to `false`, the TEE SHALL cache the OWE signer certificate and  
1005 OCSP stapling.
- 1006 • `content`: JWE encrypted data as a `CONTENT-ENCRYPTION-TYPE`.

1007 The following JSON structure SHALL be used as an input to the JWE while generating `CONTENT-`  
1008 `ENCRYPTION-TYPE`.

```
1009 {  
1010     "dsi": DSI-CONTENT-TYPE,  
1011     "nextnonce": "PRINTABLE-STRING-PRIMITIVE-TYPE"  
1012 }
```

1013 Where:

- 1014 • `dsi`: The `DSI-CONTENT-TYPE` that represents the current device state.
- 1015 • `nextnonce`: A unique value that the OWE SHALL use as nonce in the next subsequent request. See
- 1016 section 2.10.4 for details.

### 1017 **5.7.1 Error Conditions**

1018 If any validation listed in section 5.6.1 fails or if a TEE error occurs, the rSD SHALL use an appropriate  
1019 `OPERATION-RESPONSE-PRIMITIVE-TYPE` (listed below) as the `status` value in the corresponding  
1020 response message.

- 1021 • `ERR_REQUEST_INVALID`
- 1022 • `ERR_TFW_NOT_TRUSTED`
- 1023 • `ERR_OWE_NOT_TRUSTED`
- 1024 • `ERR_OCSP_INVALID`
- 1025 • `ERR_TEE_BUSY`
- 1026 • `ERR_TEE_FAIL`
- 1027 • `ERR_TEE_RESOURCE_FULL`
- 1028 • `ERR_TEE_UNKNOWN`
- 1029 • `ERR_UNSUPPORTED_CRYPTO_ALG`
- 1030 • `ERR_UNSUPPORTED_MSG_VERSION`

1031 See section 4.14 for details on error strings.

1032

## 1033 5.8 CreateSDTBSRequest

1034 An OWE SHALL issue a `CreateSDTBSRequest` message to create a new Security Domain with the given  
 1035 parameters. The message SHALL be signed using the JWS scheme and encapsulated in a  
 1036 `CreateSDRequest` message. This message SHALL always be sent to the `rSDTA`. The JSON structure for the  
 1037 `CreateSDTBSRequest` SHALL be as follows:

```
1038 {
1039     "CreateSDTBSRequest" : COMMAND-PARAMETER-TYPE
1040 }
```

1041 Within the `COMMAND-PARAMETER-TYPE`, the following JSON structure is used as an input to the JWE while  
 1042 generating `CONTENT-ENCRYPTION-TYPE`.

```
1043 {
1044     "spid" : "PRINTABLE-STRING-PRIMITIVE" ,
1045     "sdid" : "PRINTABLE-STRING-PRIMITIVE" ,
1046     "spcert" : "CERT-PRIMITIVE" ,
1047     "tsmid" : "PRINTABLE-STRING-PRIMITIVE" ,
1048     "did" : "PRINTABLE-STRING-PRIMITIVE" ,
1049     "sd_data" : "PRINTABLE-STRING-PRIMITIVE"
1050 }
```

1051 Where:

- 1052 • `spid`: The base64 encoded Service Provider identifier that is to be associated with the SD. See  
 1053 section 2.6 for the `spid` generation.
- 1054 • `sdid`: The base64 encoded UUID of the SD to be created. The `sdid` SHALL remain unchangeable  
 1055 throughout its life cycle.
- 1056 • `spcert`: SP-CERT formatted as `CERT-PRIMITIVE` that is to be associated with the SD. Only  
 1057 TAs that are signed using a key associated with the SP-CERT SHALL be allowed to be installed in the  
 1058 SD.
- 1059 • `tsmid`: The identifier of the OWE that issued the request.
- 1060 • `did`: The base64 encoded SHA-256 hash of the TEE-CERT binary. `did` is used as the device  
 1061 identifier to which the request is issued.
- 1062 • `sd_data`: (OPTIONAL) The base64url encoded SD personalization data. This element may be used  
 1063 to equip the SD with credentials required to support TMF commands. The format of the SD  
 1064 personalization data SHALL be a DER-encoded `StoredDataObject` as defined in [TMF ASN.1]  
 1065 section 8.3.3.6.

## 1066 5.8.1 Processing Requirements

1067 Before authorizing SD creation, the `rSDTA` SHALL:

- 1068 • Validate the JSON web signature associated with this request.
- 1069 • Determine whether the OWE-CERT chains to a root CA certificate in OWE-WHITELIST.
- 1070 • Check the revocation status of the OWE-CERT chain, using the cached OCSP stapling. If the cache is  
1071 unavailable or expired, the `rSD` SHALL return the corresponding response with an error string along  
1072 with an indication (`signerreq` set to `true`) to provide OCSP stapling in the next request. The OWE  
1073 may reissue the request with OCSP stapling.
- 1074 • Compare the `dsihash` value to the SHA-256 hash of the internal `DSI-TYPE` to ensure that the DSI  
1075 has not changed since the last changes requested by the OWE.
- 1076 • Compare `nonce` to the last `nextnonce` sent to the OWE to ensure that no new operation has been  
1077 authorized on SDs and TAs associated with the OWE since the last operation requested by the OWE.
- 1078 • Decrypt the `ciphertext` element of the `CONTENT-ENCRYPTION-TYPE` to obtain the SD  
1079 information.
- 1080 • Validate the format of the `spcert`.
- 1081 • Verify the `did` to ensure that the request is intended for the correct device.
- 1082 • Verify that the SD with the given `sdid` does not already exist.
- 1083 • Verify that the `tsmid` matches the OWE identifier in the OWE-CERT to ensure that the OWE issuing  
1084 the request has access to the SD. See section 2.9 for details.

1085 Upon successfully completing the above processing, the SD with the given parameters SHALL be created. If  
1086 the `spid` associated with the SD is not assigned to any SDs on the device, then the TEE SHALL also generate  
1087 a key pair called SP-AIK. A response message `CreateSDResponse` SHALL always be returned regardless  
1088 of the status of the operation.

1089

## 1090 5.9 CreateSDTBSResponse

1091 In response to a CreateSDRequest, the rSD<sub>TA</sub> SHALL return a CreateSDResponse, encapsulating the  
1092 CreateSDTBSResponse message. The JSON structure for the CreateSDTBSResponse is as follows:

```
1093 {  
1094     "CreateSDTBSResponse" : RESPONSE-PARAMETER-TYPE  
1095 }
```

1096 Within the RESPONSE-PARAMETER-TYPE, the following JSON structure is used as an input to the JWE while  
1097 generating CONTENT-ENCRYPTION-TYPE.

```
1098 {  
1099     "status" : "OPERATION-RESPONSE-PRIMITIVE-TYPE" ,  
1100     "did" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,  
1101     "sdid" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,  
1102     "spaik" : PUB-KEY-TYPE ,  
1103     "dsi" : DSI-CONTENT-TYPE ,  
1104     "nextnonce" : "PRINTABLE-STRING-PRIMITIVE-TYPE"  
1105 }
```

1106 Where:

- 1107 • status: An OPERATION-RESPONSE-PRIMITIVE-TYPE indicating the status of the  
1108 CreateSDRequest operation. If the SD is created successfully, the value of status SHALL be  
1109 OPERATION\_SUCCESS; otherwise its value SHALL be one of the error strings listed in section 5.9.1.
- 1110 • did: The value of did from the CreateSDRequest.
- 1111 • sdid: The value of the sdid from the CreateSDRequest.
- 1112 • spaik: The SP-AIK-PUB key structured as PUB-KEY-TYPE. This element is only returned if the  
1113 request caused a new SP-AIK to be generated.
- 1114 • dsi: The DSI-CONTENT-TYPE for the new device state. This element is only returned when the  
1115 nextdsi is set to true in the CreateSDRequest.
- 1116 • nextnonce: A unique value that the OWE SHALL use as nonce in the next subsequent request. See  
1117 section 2.10.4 for details.

1118

## 1119 5.9.1 Error Conditions

1120 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  
1121 OPERATION-RESPONSE-PRIMITIVE-TYPE (listed below) as the status value in the corresponding  
1122 response message.

- 1123 • ERR\_REQUEST\_INVALID
- 1124 • ERR\_OWE\_NOT\_TRUSTED
- 1125 • ERR\_OCSP\_INVALID
- 1126 • ERR\_DEV\_STATE\_MISMATCH
- 1127 • ERR\_SDID\_ALREADY\_USED
- 1128 • ERR\_SPCERT\_INVALID
- 1129 • ERR\_TEE\_BUSY
- 1130 • ERR\_TEE\_FAIL
- 1131 • ERR\_TEE\_RESOURCE\_FULL
- 1132 • ERR\_TEE\_UNKNOWN
- 1133 • ERR\_UNSUPPORTED\_CRYPTO\_ALG
- 1134 • ERR\_UNSUPPORTED\_MSG\_VERSION

1135 See section 4.14 for details on error strings.

1136

## 1137 5.10 UpdateSDTBSRequest

1138 An OWE SHALL issue an UpdateSDTBSRequest message to update SD metadata with the given  
 1139 parameters. The message SHALL be signed using the JWS scheme and encapsulated in an  
 1140 UpdateSDRequest message. This message SHALL always be sent to rSD<sub>TA</sub>. A JSON structure for the  
 1141 UpdateSDTBSRequest SHALL be as follows:

```
1142 {
1143     "UpdateSDTBSRequest" : COMMAND-PARAMETER-TYPE
1144 }
```

1145 Within the COMMAND-PARAMETER-TYPE, the following JSON structure is used as an input to the JWE while  
 1146 generating CONTENT-ENCRYPTION-TYPE:

```
1147 {
1148     "tsmid" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1149     "spid" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1150     "did" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1151     "sdid" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1152     "changes" : {
1153         "newspid" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1154         "spcert" : [ "CERT-PRIMITIVE-TYPE" ] ,
1155         "deloldspcert" : [ "PRINTABLE-STRING-PRIMITIVE-TYPE" ] ,
1156         "sd_data" : "PRINTABLE-STRING-PRIMITIVE-TYPE"
1157     }
1158 }
```

1159 Where:

- 1160 • tsmid: The identifier of the OWE that issued the request.
- 1161 • spid: The base64 encoded service provide identifier that is associated with the SD.
- 1162 • did: The base64 encoded SHA-256 hash of the TEE-CERT binary. did is used as the device  
 1163 identifier to which the request is issued.
- 1164 • sdid: The base64 encoded UUID representing the SD to be updated.
- 1165 • changes: A JSON structure containing parameters to be updated. All elements within this JSON  
 1166 structure are **optional**.
- 1167 • newspid: (OPTIONAL) The new base64 encoded service provide identifier that is to be associated  
 1168 with the SD. If the newspid is not associated with any existing SDs in the device, the rSD SHALL  
 1169 generate a new SP-AIK key pair for the newspid.
- 1170 • spcert: (OPTIONAL) An array of SP-CERTs formatted as CERT-PRIMITIVE-TYPE that is to be  
 1171 associated with the SD.

- 1172 • `deloldspcert`: (OPTIONAL) The base64 encoded SHA-256 hash value of SP-CERTs previously  
1173 assigned to the SD that are to be deleted.
- 1174 • `sd_data`: (OPTIONAL) The base64 encoded SD personalization data. This element may be used to  
1175 equip the SD with credentials required to support TMF commands. The format of the SD  
1176 personalization data SHALL be a DER-encoded `StoredDataObject` as defined in [TMF ASN.1]  
1177 section 8.3.3.6.

1178

### 1179 5.10.1 Processing Requirements

1180 Before authorizing the SD update, the `rSDTA` SHALL:

- 1181 • Validate the JSON web signature associated with the request.
- 1182 • Determine whether the OWE-CERT chains to a root CA certificate in OWE-WHITELIST.
- 1183 • Check the revocation status of the OWE-CERT chain, using the cached OCSP stapling. If the cache is  
1184 unavailable or expired, the `rSD` SHALL return the corresponding response with an error string along  
1185 with an indication (`signerreq` set to `true`) to provide OCSP stapling in the next request. The OWE  
1186 may reissue the request with OCSP stapling.
- 1187 • Compare the `dsihash` value to the SHA-256 hash of the internal `DSI-TYPE` to ensure that the DSI  
1188 has not changed since the last changes requested by the OWE.
- 1189 • Compare `nonce` to the last `nextnonce` sent to the OWE to ensure that no new operation has been  
1190 authorized on SDs and TAs associated with the OWE since the last operation requested by the OWE.
- 1191 • Decrypt the `ciphertext` element of the `CONTENT-ENCRYPTION-TYPE` to obtain the update  
1192 parameters.
- 1193 • Verify the `did` to ensure that the request is intended for the correct device.
- 1194 • Verify that the SD with the given `sdid` exists.
- 1195 • Verify that the `tsmid` matches the OWE identifier in the OWE-CERT to ensure that the OWE issuing  
1196 the request has access to the SD. See section 2.9 for details.

1197 Upon successfully completing the above requirements, the specified SD is updated with the given parameters.  
1198 If the update operation results in generation of a new SP-AIK, the newly generated SP-AIK SHALL replace the  
1199 existing SP-AIK. A response message `UpdateSDResponse` SHALL always be returned regardless of the  
1200 status of the operation.

1201



## 1202 5.11 UpdateSDTBSResponse

1203 In response to an UpdateSDRequest, the rSD<sub>TA</sub> SHALL return an UpdateSDResponse, encapsulating the  
1204 UpdateSDTBSResponse message. The JSON structure for the UpdateSDTBSResponse SHALL be as  
1205 follows:

```
1206 {  
1207     "UpdateSDTBSResponse" : RESPONSE-PARAMETER-TYPE  
1208 }
```

1209 Within the RESPONSE-PARAMETER-TYPE, the following JSON structure is used as an input to the JWE while  
1210 generating CONTENT-ENCRYPTION-TYPE:

```
1211 {  
1212     "status" : "OPERATION-RESPONSE-PRIMITIVE-TYPE" ,  
1213     "did" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,  
1214     "spaik" : PUB-KEY-TYPE ,  
1215     "dsi" : DSI-CONTENT-TYPE ,  
1216     "nextnonce" : "PRINTABLE-STRING-PRIMITIVE-TYPE"  
1217 }
```

1218 Where:

- 1219 • status: An OPERATION-RESPONSE-PRIMITIVE-TYPE indicating the status of the  
1220 UpdateSDRequest operation. If the SD is updated successfully, the value of status SHALL be  
1221 OPERATION\_SUCCESS; otherwise its value SHALL be an error string listed in section 5.11.1.
- 1222 • did: The value of did from the UpdateSDRequest.
- 1223 • spaik: The SP-AIK-PUB key structured as PUB-KEY-TYPE. This element is only returned if the  
1224 UpdateSDRequest causes the rSD<sub>TA</sub> to generate a new SP-AIK.
- 1225 • dsi: The DSI-CONTENT-TYPE for the new device state. This element is only returned when the  
1226 nextdsi is set to true in the UpdateSDRequest.
- 1227 • nextnonce: A unique value that the OWE SHALL use as nonce in the next subsequent request. See  
1228 section 2.10.4 for details.

1229

### 1230 5.11.1 Error Conditions

1231 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  
1232 OPERATION-RESPONSE-PRIMITIVE-TYPE (listed below) as the status value in the corresponding  
1233 response message.

- 1234 • ERR\_REQUEST\_INVALID
- 1235 • ERR\_OWE\_NOT\_TRUSTED
- 1236 • ERR\_OCSP\_INVALID
- 1237 • ERR\_DEV\_STATE\_MISMATCH
- 1238 • ERR\_SPCERT\_INVALID
- 1239 • ERR\_TEE\_BUSY
- 1240 • ERR\_TEE\_FAIL
- 1241 • ERR\_TEE\_RESOURCE\_FULL
- 1242 • ERR\_TEE\_UNKNOWN
- 1243 • ERR\_UNSUPPORTED\_CRYPT\_ALG
- 1244 • ERR\_UNSUPPORTED\_MSG\_VERSION

1245 See section 4.14 for details on error strings.

1246

## 1247 5.12 DeleteSDTBSRequest

1248 An OWE SHALL issue a `DeleteSDTBSRequest` message to delete a specified SD and optionally delete all  
 1249 TAs contained within the SD. The message SHALL be signed using the JWS scheme and encapsulated in a  
 1250 `DeleteSDRequest` message. This message SHALL always be sent to the `rSDTA`. The JSON structure for the  
 1251 `DeleteSDTBSRequest` SHALL be as follows:

```
1252 {
1253     "DeleteSDTBSRequest" : COMMAND-PARAMETER-TYPE
1254 }
```

1255 Within the `COMMAND-PARAMETER-TYPE`, the following JSON structure is used as an input to the JWE while  
 1256 generating `CONTENT-ENCRYPTION-TYPE`:

```
1257 {
1258     "tsmid" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1259     "did" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1260     "sdid" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1261     "deletetas" : BOOLEAN
1262 }
```

1263 Where:

- 1264 • `tsmid`: The identifier of the OWE that issued the request.
- 1265 • `did`: The base64 encoded SHA-256 hash of the TEE-CERT binary. `did` is used as the device  
 1266 identifier to which the request is issued.
- 1267 • `sdid`: The base64 encoded UUID representing the SD to be deleted.
- 1268 • `deletetas`: A Boolean value indicating whether to delete all TAs within the SD. If set as `false`,  
 1269 deleting an SD with one or more TAs installed SHALL cause a failure.

1270

### 1271 5.12.1 Processing Requirements

1272 Before authorizing the deletion of the SD, the `rSDTA` SHALL:

- 1273 • Validate the JSON web signature associated with the request.
- 1274 • Determine whether the OWE-CERT chains to a root CA certificate in the OWE-WHITELIST.
- 1275 • Check the revocation status of the OWE-CERT chain, using the cached OCSP stapling. If the cache is  
 1276 unavailable or expired, the `rSD` SHALL return the corresponding response with an error string along  
 1277 with an indication (`signerreq` set to `true`) to provide OCSP stapling in the next request. The OWE  
 1278 may reissue the request with OCSP stapling.
- 1279 • Compare the `dsihash` value to the SHA-256 hash of the internal `DSI-TYPE` to ensure that the DSI  
 1280 has not changed since the last changes requested by the OWE.
- 1281 • Compare `nonce` to the last `nextnonce` sent to the OWE to ensure that no new operation has been  
 1282 authorized on SDs and TAs associated with the OWE since the last operation requested by the OWE.

- 1283       • Decrypt the `ciphertext` element of the `CONTENT-ENCRYPTION-TYPE` to obtain the deletion  
1284       parameters.
- 1285       • Verify the `did` to ensure that the request is intended for the correct device.
- 1286       • Verify that the `tsmid` matches the OWE identifier in the OWE-CERT to ensure that the OWE issuing  
1287       the request has access to the SD. See section 2.9 for details.
- 1288       • Ensure that, if `deletetas` is set to `false`, the SD contains no TAs; otherwise the deletion SHALL  
1289       be aborted, resulting in a failure.
- 1290   Upon successfully completing the above requirements, the specified SD SHALL be deleted. A response  
1291   message `DeleteSDResponse` SHALL always be returned regardless of the status of the operation.
- 1292

## 1293 5.13 DeleteSDTBSResponse

1294 In response to a DeleteSDRequest command, the rSD<sub>TA</sub> SHALL return a DeleteSDResponse,  
1295 encapsulating the DeleteSDTBSResponse message. The JSON structure for the DeleteSDTBSResponse  
1296 is as follows:

```
1297 {  
1298     "DeleteSDTBSResponse" : RESPONSE-PARAMETER-TYPE  
1299 }
```

1300 Within the RESPONSE-PARAMETER-TYPE, the following JSON structure is used as an input to the JWE while  
1301 generating CONTENT-ENCRYPTION-TYPE.

```
1302 {  
1303     "status" : "OPERATION-RESPONSE-PRIMITIVE-TYPE" ,  
1304     "did" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,  
1305     "dsi" : DSI-CONTENT-TYPE ,  
1306     "nextnonce" : "PRINTABLE-STRING-PRIMITIVE-TYPE"  
1307 }
```

1308 Where:

- 1309 • status: An OPERATION-RESPONSE-PRIMITIVE-TYPE indicating the status of the  
1310 DeleteSDRequest operation. If the SD is deleted successfully, the value of status SHALL be  
1311 OPERATION\_SUCCESS; otherwise its value SHALL be an error string listed in section 5.13.1.
- 1312 • did: The value of did from the DeleteSDRequest.
- 1313 • dsi: The DSI-CONTENT-TYPE for the new device state. This element is only returned when the  
1314 nextdsi is set to true in the DeleteSDRequest.
- 1315 • nextnonce: A unique value that the OWE SHALL use as nonce in the next subsequent request. See  
1316 section 2.10.4 for details.

### 1317 5.13.1 Error Conditions

1318 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  
1319 OPERATION-RESPONSE-PRIMITIVE-TYPE (listed below) as the status value in the corresponding  
1320 response message.

- 1321 • ERR\_REQUEST\_INVALID
- 1322 • ERR\_OWE\_NOT\_TRUSTED
- 1323 • ERR\_OCSP\_INVALID
- 1324 • ERR\_DEV\_STATE\_MISMATCH
- 1325 • ERR\_SPCERT\_INVALID
- 1326 • ERR\_SD\_NOT\_EMPTY
- 1327 • ERR\_TEE\_BUSY
- 1328 • ERR\_TEE\_FAIL
- 1329 • ERR\_TEE\_RESOURCE\_FULL
- 1330 • ERR\_TEE\_UNKNOWN
- 1331 • ERR\_UNSUPPORTED\_CRYPTO\_ALG
- 1332 • ERR\_UNSUPPORTED\_MSG\_VERSION

1333 See section 4.14 for details on error strings.

1334

## 1335 5.14 InstallTATBSRequest

1336 An OWE SHALL issue an `InstallTATBSRequest` message to install a specified TA into a specified Security  
 1337 Domain. The message SHALL be signed using the JWS scheme and encapsulated in an `InstallTAResponse`  
 1338 message. This message SHALL always be sent to `rSDTA`. The JSON structure for `InstallTATBSRequest`  
 1339 is as follows:

```
1340 {
1341     "InstallTATBSRequest": COMMAND-PARAMETER-TYPE
1342 }
```

1343 Within the `COMMAND-HEADER-TYPE` (section 4.4), the following JSON structure is used as an input to the  
 1344 JWE while generating `CONTENT-ENCRYPTION-TYPE`.

```
1345 {
1346     "tsmid": "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1347     "did": "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1348     "spid": "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1349     "sdid": "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1350     "spcert": "CERT-PRIMITIVE-TYPE" ,
1351     "taid": "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1352     "taver": "PRINTABLE-STRING-PRIMITIVE-TYPE"
1353 }
```

1354 Where:

- 1355 • `tsmid`: The identifier of the OWE that issued the request.
- 1356 • `did`: The base64 encoded SHA-256 hash of the TEE-CERT binary. `did` is used as the device  
 1357 identifier to which the request is issued.
- 1358 • `spid`: The base64 encoded Service Provider identifier that is associated with the SD.
- 1359 • `sdid`: The base64 encoded UUID of the SD where the TA is to be installed.
- 1360 • `spcert`: (OPTIONAL) The SP-CERT formatted as `CERT-PRIMITIVE-TYPE` that signed the TA.  
 1361 This element is provided when the TA is signed with an SP-CERT that was not previously associated  
 1362 with the SD.
- 1363 • `taid`: The base64 encoded UUID of the TA to be installed.
- 1364 • `taver`: The string containing the TA version information.

1365

1366 Additionally, the `InstallTATBSRequest` SHALL include the following JSON elements in the `COMMAND-`  
 1367 `PARAMETER-TYPE`.

1368 `"encrypted_ta_bin":CONTENT-ENCRYPTION-TYPE`

1369 `"encrypted_ta_data":CONTENT-ENCRYPTION-TYPE`

1370 • `encrypted_ta_bin`: An encrypted TA binary structured as a `CONTENT-ENCRYPTION-TYPE` where  
 1371 the CEK is wrapped using the SP-AIK-PUB.

1372 • `encrypted_ta_data`: (OPTIONAL) An encrypted TA personalization data structured as a  
 1373 `CONTENT-ENCRYPTION-TYPE` where the CEK is wrapped using the SP-AIK-PUB. The TA should be  
 1374 able to access the personalization data via interfaces defined in [TEE Core]. The format of the SD  
 1375 personalization data SHALL be a DER-encoded `StoredDataObject` as defined in [TMF ASN.1]  
 1376 section 8.3.3.6.

1377

### 1378 5.14.1 Processing Requirements

1379 Before authorizing the deletion of the SD, the `rSDTA` SHALL:

- 1380 • Validate the JSON web signature associated with the request.
- 1381 • Determine whether the OWE-CERT chains to a root CA certificate in the OWE-WHITELIST.
- 1382 • Check the revocation status of the OWE-CERT chain, using the cached OCSP stapling. If the cache is  
 1383 unavailable or expired, the `rSD` SHALL return the corresponding response with an error string along  
 1384 with an indication (`signerreq` set to `true`) to provide OCSP stapling in the next request. The OWE  
 1385 may reissue the request with OCSP stapling.
- 1386 • Compare the `dsihash` value to the SHA-256 hash of the internal `DSI-TYPE` to ensure that the DSI  
 1387 has not changed since the last changes requested by the OWE.
- 1388 • Compare `nonce` to the last `nextnonce` sent to the OWE to ensure that no new operation has been  
 1389 authorized on SDs and TAs associated with the OWE since the last operation requested by the OWE.
- 1390 • Decrypt the `ciphertext` element of the `CONTENT-ENCRYPTION-TYPE` to obtain the TA  
 1391 information.
- 1392 • Validate the format of the `spcert`.
- 1393 • Verify the `did` to ensure that the request is intended for the correct device.
- 1394 • Verify that the `tsmid` matches the OWE identifier in the OWE-CERT to ensure that the OWE issuing  
 1395 the request has access to the SD. See section 2.9 for details.
- 1396 • Verify that, if the TA already exists in the device, the version of the TA to be installed is higher than the  
 1397 existing TA.
- 1398 • Use SP-AIK-PRIV key to decrypt TA binary and personalization data. If the version of the TA  
 1399 associated with the TA binary is different than the `taver` element, the `rSDTA` SHALL abort the update  
 1400 process.
- 1401 • Validate the TA signature using an SP-CERT associated with the SD. The TA signing mechanism may  
 1402 be specific to the TEE OS.

1403 Upon successfully completing the above requirements, the given TA SHALL be installed into the specified SD.  
 1404 A response message `InstallTAResponse` SHALL always be returned regardless of the status of the  
 1405 operation.



## 1406 5.15 InstallTATBSResponse

1407 In response to an InstallTAResponse, the rSD<sub>TA</sub> SHALL return an InstallTAResponse, encapsulating  
1408 the InstallTATBSResponse message. The JSON structure for the InstallTATBSResponse SHALL be  
1409 as follows:

```
1410 {  
1411     "InstallTATBSResponse" : RESPONSE-PARAMETER-TYPE  
1412 }
```

1413 Within the RESPONSE-PARAMETER-TYPE, the following JSON structure SHALL be used as an input to the  
1414 JWE while generating CONTENT-ENCRYPTION-TYPE.

```
1415 {  
1416     "status" : "OPERATION-RESPONSE-PRIMITIVE-TYPE" ,  
1417     "did" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,  
1418     "dsi" : DSI-CONTENT-TYPE ,  
1419     "nextnonce" : "PRINTABLE-STRING-PRIMITIVE-TYPE"  
1420 }
```

1421 Where:

- 1422 • status: An OPERATION-RESPONSE-PRIMITIVE-TYPE indicating the status of the  
1423 InstallTAResponse operation. If the TA is installed successfully, the value of status SHALL be  
1424 OPERATION\_SUCCESS; otherwise its value SHALL be an error string listed in section 5.15.1.
- 1425 • did: The value of did from previous messages.
- 1426 • dsi: The DSI-CONTENT-TYPE for the new device state. This element is only returned when the  
1427 nextdsi is set to true in the InstallTAResponse.
- 1428 • nextnonce: A unique value that the OWE SHALL use as nonce in the next subsequent request. See  
1429 section 2.10.4 for details.

1430

### 1431 5.15.1 Error Conditions

1432 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  
1433 OPERATION-RESPONSE-PRIMITIVE-TYPE (listed below) as the status value in the corresponding  
1434 response message.

- 1435 • ERR\_REQUEST\_INVALID
- 1436 • ERR\_OWE\_NOT\_TRUSTED
- 1437 • ERR\_OCSP\_INVALID
- 1438 • ERR\_DEV\_STATE\_MISMATCH
- 1439 • ERR\_SPCERT\_INVALID
- 1440 • ERR\_TA\_ALREADY\_INSTALLED
- 1441 • ERR\_TA\_INVALID
- 1442 • ERR\_TEE\_BUSY
- 1443 • ERR\_TEE\_FAIL
- 1444 • ERR\_TEE\_RESOURCE\_FULL
- 1445 • ERR\_TEE\_UNKNOWN
- 1446 • ERR\_UNSUPPORTED\_CRYPTO\_ALG
- 1447 • ERR\_UNSUPPORTED\_MSG\_VERSION

1448 See section 4.14 for details on error strings.

1449

## 1450 5.16 UpdateTATBSRequest

1451 An OWE SHALL issue an `UpdateTATBSRequest` message to update TA binary and/or TA personalization  
 1452 data. The message SHALL be signed using the JWS scheme and encapsulated in an `UpdateTARequest`  
 1453 message. This message SHALL always be sent to `rSDTA`. The JSON structure for the `UpdateTATBSRequest`  
 1454 SHALL be as follows:

```
1455 {
1456     "UpdateTATBSRequest" : COMMAND-PARAMETER-TYPE
1457 }
```

1458 Within the `COMMAND-PARAMETER-TYPE`, the following JSON structure SHALL be used as an input to the JWE  
 1459 while generating `CONTENT-ENCRYPTION-TYPE`:

```
1460 {
1461     "tsmid" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1462     "did" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1463     "spid" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1464     "sdid" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1465     "spcert" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1466     "taid" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1467     "newtaver" : "PRINTABLE-STRING-PRIMITIVE-TYPE"
1468 }
1469
```

1470 Where:

- 1471 • `tsmid`: The identifier of the OWE that issued the request.
- 1472 • `did`: The base64 encoded SHA-256 hash of the TEE-CERT binary. `did` is used as the device  
 1473 identifier to which the request is issued.
- 1474 • `spid`: The base64 encoded Service Provider identifier that is associated with the SD.
- 1475 • `sdid`: The base64 encoded UUID of the SD where the TA is to be installed.
- 1476 • `spcert`: (OPTIONAL) The SP-CERT formatted as `CERT-PRIMITIVE-TYPE` that signed the TA.  
 1477 This element is provided when the TA is signed with a SP-CERT that was not previously associated  
 1478 with the SD.
- 1479 • `taid`: The base64 encoded UUID of the TA to be installed.
- 1480 • `newtaver`: (OPTIONAL) The string containing the TA version information that is to be updated.

1481 Additionally, the `UpdateTATBSRequest` SHALL include at least one of the following JSON elements in the  
 1482 `COMMAND-PARAMETER-TYPE`.

1483 `"encrypted_ta_bin" :CONTENT-ENCRYPTION-TYPE`

1484 `"encrypted_ta_data" :CONTENT-ENCRYPTION-TYPE`

- 1485 • `encrypted_ta_bin`: An encrypted TA binary that replaces the existing TA binary, structured as a  
 1486 `CONTENT-ENCRYPTION-TYPE` where the CEK is wrapped using the SP-AIK-PUB.
- 1487 • `encrypted_ta_data`: (OPTIONAL) An encrypted TA personalization data, that replaces the existing  
 1488 TA personalization data, structured as a `CONTENT-ENCRYPTION-TYPE` where the CEK is wrapped  
 1489 using the SP-AIK-PUB. The TA should be able to access the personalization data via interfaces  
 1490 defined in [TEE Core]. The format of the SD personalization data SHALL be a DER-encoded  
 1491 `StoredDataObject` as defined in [TMF ASN.1] section 8.3.3.6.

### 1492 5.16.1 Processing Requirements

1493 Before authorizing the update on the TA, the `rSDTA` SHALL:

- 1494 • Validate the JSON web signature associated with the request.
- 1495 • Determine whether the OWE-CERT chains to a root CA certificate in OWE-WHITELIST.
- 1496 • Check the revocation status of the OWE-CERT chain, using the cached OCSP stapling. If the cache is  
 1497 unavailable or expired, the `rSD` SHALL return the corresponding response with an error string along  
 1498 with an indication (`signerreq` set to `true`) to provide OCSP stapling in the next request. The OWE  
 1499 may reissue the request with OCSP stapling.
- 1500 • Compare the `dsihash` value to the SHA-256 hash of the internal `DSI-TYPE` to ensure that the DSI  
 1501 has not changed since the last changes requested by the OWE.
- 1502 • Compare `nonce` to the last `nextnonce` sent to the OWE to ensure that no new operation has been  
 1503 authorized on SDs and TAs associated with the OWE since the last operation requested by the OWE.
- 1504 • Decrypt the `ciphertext` element of the `CONTENT-ENCRYPTION-TYPE` to obtain the TA  
 1505 information.
- 1506 • Verify the `did` to ensure that the request is intended for the correct device.
- 1507 • Verify that the `tsmid` matches the OWE identifier in the OWE-CERT to ensure that the OWE issuing  
 1508 the request has access to the SD. See section 2.9 for details.
- 1509 • Verify that the version of the TA to be updated is higher than the one that is currently installed. If the  
 1510 update command does not contain the `encrypted_ta_bin` element, the `rSDTA` SHALL ignore the  
 1511 `newtaver` element.
- 1512 • Use the SP-AIK-PRIV key to decrypt TA binary and personalization data. If the version of the TA  
 1513 associated with the TA binary is different than the `newtaver` element, the `rSDTA` SHALL abort the  
 1514 update process.
- 1515 • Validate the TA signature using an SP-CERT associated with the SD. The TA signing mechanism may  
 1516 be specific to the TEE OS.

1517 Upon successfully completing the above steps, the given TA SHALL be updated. Prior to an update, the TA  
 1518 SHALL be forcefully shut down as defined in [TMF ASN.1] section 11. A response message  
 1519 `UpdateTAResponse` SHALL always be returned regardless of the status of the operation.

1520

## 1521 5.17 UpdateTATBSResponse

1522 In response to an UpdateTAResponse, the TEE SHALL return an UpdateTAResponse, encapsulating the  
1523 UpdateTATBSResponse message. The JSON structure for the UpdateTATBSResponse SHALL be as  
1524 follows:

```
1525 {  
1526     "UpdateTATBSResponse" : RESPONSE-PARAMETER-TYPE  
1527 }
```

1528 Within the RESPONSE-PARAMETER-TYPE, the following JSON structure is used as an input to the JWE while  
1529 generating CONTENT-ENCRYPTION-TYPE.

```
1530 {  
1531     "status" : "OPERATION-REASON-PRIMITIVE-TYPE" ,  
1532     "did" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,  
1533     "dsi" : DSI-CONTENT-TYPE ,  
1534     "nextnonce" : "PRINTABLE-STRING-PRIMITIVE-TYPE"  
1535 }
```

1536 Where:

- 1537 • status: An OPERATION-RESPONSE-PRIMITIVE-TYPE indicating the status of the  
1538 UpdateTAResponse operation. If the TA is successfully updated, the value of status SHALL be  
1539 OPERATION\_SUCCESS; otherwise its value SHALL be an error string listed in section 5.17.1.
- 1540 • did: The value of did from previous messages.
- 1541 • dsi: The DSI-CONTENT-TYPE for the new device state. This element is only returned when the  
1542 nextdsi is set to true in the UpdateTAResponse.
- 1543 • nextnonce: A unique value that the OWE SHALL use as nonce in the next subsequent request. See  
1544 section 2.10.4 for details.

1545

### 1546 5.17.1 Error Conditions

1547 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  
1548 OPERATION-RESPONSE-PRIMITIVE-TYPE (listed below) as the status value in the corresponding  
1549 response message.

- 1550 • ERR\_REQUEST\_INVALID
- 1551 • ERR\_OWE\_NOT\_TRUSTED
- 1552 • ERR\_OCSP\_INVALID
- 1553 • ERR\_DEV\_STATE\_MISMATCH
- 1554 • ERR\_SPCERT\_INVALID
- 1555 • ERR\_TA\_ALREADY\_INSTALLED
- 1556 • ERR\_TA\_INVALID
- 1557 • ERR\_TEE\_BUSY
- 1558 • ERR\_TEE\_FAIL
- 1559 • ERR\_TEE\_RESOURCE\_FULL
- 1560 • ERR\_TEE\_UNKNOWN
- 1561 • ERR\_UNSUPPORTED\_CRYPTO\_ALG
- 1562 • ERR\_UNSUPPORTED\_MSG\_VERSION

1563 See section 4.14 for details on error strings.

1564

## 1565 5.18 DeleteTATBSRequest

1566 An OWE SHALL issue a `DeleteTATBSRequest` message to delete a specific TA from a specified SD. The  
 1567 message SHALL be signed using the JWS scheme and encapsulated in a `DeleteTAREquest` message.  
 1568 This message SHALL always be sent to `rSDTA`. The JSON structure for the `DeleteTATBSRequest` is as  
 1569 follows:

```
1570 {
1571     "DeleteTATBSRequest" : COMMAND-PARAMETER-TYPE
1572 }
```

1573 Within the `COMMAND-PARAMETER-TYPE`, the following JSON structure SHALL be used as an input to the JWE  
 1574 while generating `CONTENT-ENCRYPTION-TYPE`.

```
1575 {
1576     "tsmid" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1577     "did" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1578     "sdid" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1579     "taid" : "PRINTABLE-STRING-PRIMITIVE-TYPE"
1580 }
```

1581 Where:

- 1582 • `tsmid`: The identifier of the OWE that issued the request.
  - 1583 • `did`: The base64 encoded SHA-256 hash of the TEE-CERT binary. `did` is used as the device  
 1584 identifier to which the request is issued.
  - 1585 • `sdid`: The base64 encoded UUID of the SD where the TA is installed.
  - 1586 • `taid`: The base64 encoded UUID of a TA that is to be deleted.
- 1587

### 1588 5.18.1 Processing Requirements

1589 Before authorizing the deletion of the SD, the `rSDTA` SHALL:

- 1590 • Validate the JSON web signature associated with the request.
- 1591 • Determine whether the OWE-CERT chains to a root CA certificate in OWE-WHITELIST.
- 1592 • Check the revocation status of the OWE-CERT chain, using the cached OCSP stapling. If the cache is  
 1593 unavailable or expired, the `rSD` SHALL return the corresponding response with an error string along  
 1594 with an indication (`signerreq` set to `true`) to provide OCSP stapling in the next request. The OWE  
 1595 may reissue the request with OCSP stapling.
- 1596 • Compare the `dsihash` value to the SHA-256 hash of the internal `DSI-TYPE` to ensure that the DSI  
 1597 has not changed since the last changes requested by the OWE.
- 1598 • Compare `nonce` to the last `nextnonce` sent to the OWE to ensure that no new operation has been  
 1599 authorized on SDs and TAs associated with the OWE since the last operation requested by the OWE.

- 1600       • Decrypt the `ciphertext` element of the `CONTENT-ENCRYPTION-TYPE` to obtain the TA information  
1601       for deletion.
- 1602       • Verify the `did` to ensure that the request is intended for the correct device.
- 1603       • Verify that the `tsmid` matches the OWE identifier in the OWE-CERT to ensure that the OWE issuing  
1604       the request has access to the SD. See section 2.9 for details.
- 1605       Upon successfully completing the above requirements, the given TA SHALL be deleted. Prior to the deletion,  
1606       the TA SHALL be forcefully shut down as defined in [TMF ASN.1] section 11. A response message  
1607       `DeleteTAResponse` SHALL always be returned regardless of the status of the operation.
- 1608



## 1609 5.19 DeleteTATBSResponse

1610 In response to a DeleteTAResponse command, the rSD<sub>TA</sub> SHALL return a DeleteTAResponse,  
1611 encapsulating the DeleteTATBSResponse message. The JSON structure for the DeleteTATBSResponse  
1612 SHALL be as follows:

```
1613 {  
1614     "DeleteTATBSResponse": RESPONSE-PARAMETER-TYPE  
1615 }
```

1616 Within the RESPONSE-PARAMETER-TYPE, the following JSON structure is used as an input to the JWE while  
1617 generating CONTENT-ENCRYPTION-TYPE.

```
1618 {  
1619     "status": "OPERATION-RESPONSE-PRIMITIVE-TYPE" ,  
1620     "did": "PRINTABLE-STRING-PRIMITIVE-TYPE" ,  
1621     "dsi": DSI-CONTENT-TYPE ,  
1622     "nextnonce": "PRINTABLE-STRING-PRIMITIVE-TYPE"  
1623 }
```

1624 Where:

- 1625 • status: An OPERATION-RESPONSE-PRIMITIVE-TYPE indicating the status of the  
1626 DeleteSDRequest operation. If the TA is deleted successfully, the value of status SHALL be  
1627 OPERATION\_SUCCESS; otherwise its value SHALL be an error string listed in section 5.19.1.
- 1628 • did: The value of did from the DeleteSDRequest.
- 1629 • dsi: The DSI-CONTENT-TYPE for the new device state. This element is only returned when the  
1630 nextdsi is set to true in the DeleteSDRequest.
- 1631 • nextnonce: A unique value that the OWE SHALL use as nonce in the next subsequent request. See  
1632 section 2.10.4 for details.

1633

### 1634 5.19.1 Error Conditions

1635 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  
1636 OPERATION-RESPONSE-PRIMITIVE-TYPE (listed below) as the status value in the corresponding  
1637 response message.

- 1638 • ERR\_REQUEST\_INVALID
- 1639 • ERR\_OWE\_NOT\_TRUSTED
- 1640 • ERR\_OCSP\_INVALID
- 1641 • ERR\_DEV\_STATE\_MISMATCH
- 1642 • ERR\_TA\_NOT\_FOUND
- 1643 • ERR\_TEE\_BUSY
- 1644 • ERR\_TEE\_FAIL
- 1645 • ERR\_TEE\_RESOURCE\_FULL
- 1646 • ERR\_TEE\_UNKNOWN
- 1647 • ERR\_UNSUPPORTED\_CRYPT\_ALG
- 1648 • ERR\_UNSUPPORTED\_MSG\_VERSION

1649 See section 4.14 for details on error strings.

1650

## 1651 5.20 StoreTEEPPropertyTBSRequest

1652 An OWE SHALL issue a `StoreTEEPPropertyTBSRequest` message to store, update, or delete TEE  
 1653 properties. The message SHALL be signed using the JWS scheme and encapsulated in a  
 1654 `StoreTEEPPropertyRequest` message. This message SHALL always be issued to an `rSDTEE`.

1655 TEE properties are described in [TMF ASN.1] section A.5. The OTrP Profile supports only the TEE property  
 1656 `gpd.tee.tmf.resetpreserved.entities`, which is used to indicate entities as UUIDs to be preserved  
 1657 across a Factory Reset operation on TEE.

1658 The JSON structure for the `StoreTEEPPropertyTBSRequest` is as follows:

```
1659 {
1660     "StoreTEEPPropertyTBSRequest" : COMMAND-PARAMETER-TYPE
1661 }
```

1662 The following JSON elements will be used as input to the `CONTENT-ENCRYPTION-TYPE` within `COMMAND-`  
 1663 `TYPE`.

```
1664 {
1665     "tsmid" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1666     "did" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,
1667     "property" : "gpd.tee.tmf.resetpreserved.entities" ,
1668     "value" : {
1669         "taids" : UUID-ARRAY-TYPE ,
1670         "sdids" : UUID-ARRAY-TYPE
1671     }
1672 }
```

1673 Where:

- 1674 • `tsmid`: The identifier of the OWE that issued the request.
- 1675 • `did`: The base64 encoded SHA-256 hash of the TEE-CERT binary. `did` is used as the device  
 1676 identifier to which the request is issued.
- 1677 • `property`: OTrP Profile SHALL support only the TEE property  
 1678 `gpd.tee.tmf.resetpreserved.entities`.
- 1679 • `value`: The value of the TEE property.
- 1680 • `taids`: UUIDS of TAs structured as `UUID-ARRAY-TYPE` that SHALL be preserved across a Factory  
 1681 Reset operation on TEE.
- 1682 • `sdids`: UUIDS of SDs structured as `UUID-ARRAY-TYPE` that SHALL be preserved across a Factory  
 1683 Reset operation on TEE.

1684 The `StoreTEEPPropertyTBSRequest` SHALL always replace the previous value of the TEE property.

1685

## 1686 5.20.1 Processing Requirements

1687 Upon receiving the `StoreTEEPPropertyRequest` message, the `rSDTEE` SHALL:

- 1688 • Validate the JSON web signature associated with the request, using the OWE-PUB associated with  
1689 the OWE-CERT.
- 1690 • Determine whether the OWE-CERT chains to a root CA certificate in its OWE-WHITELIST.
- 1691 • Check the revocation status of the OWE-CERT chain, using the cached OCSP stapling. If the cache is  
1692 unavailable or expired, the `rSD` SHALL return the corresponding response with an error string along  
1693 with an indication (`signerreq` set to `true`) to provide OCSP stapling in the next request. The OWE  
1694 may reissue the request with OCSP stapling.

1695 Upon successfully completing the above requirements, the `rSDTEE` SHALL replace the TEE property with the  
1696 given value. A response message `StoreTEEPPropertyResponse` SHALL always be returned regardless of  
1697 the status of the operation.

1698

## 1699 5.21 StoreTEEPPropertyTBSResponse

1700 In response to a StoreTEEPPropertyRequest command, the rSD<sub>TEE</sub> SHALL return a  
1701 StoreTEEPPropertyResponse, encapsulating the StoreTEEPPropertyTBSResponse message. The JSON  
1702 structure for the StoreTEEPPropertyTBSResponse is as follows:

```
1703 {  
1704     "StoreTEEPPropertyTBSResponse": RESPONSE-PARAMETER-TYPE  
1705 }
```

1706 Within the RESPONSE-PARAMETER-TYPE, the following JSON structure is used as an input to the JWE while  
1707 generating CONTENT-ENCRYPTION-TYPE.

```
1708 {  
1709     "status": "OPERATION-RESPONSE-PRIMITIVE-TYPE" ,  
1710     "did": "PRINTABLE-STRING-PRIMITIVE-TYPE" ,  
1711     "dsi": DSI-CONTENT-TYPE ,  
1712     "nextnonce": "PRINTABLE-STRING-PRIMITIVE-TYPE"  
1713 }
```

1714 Where:

- 1715 • status: An OPERATION-RESPONSE-PRIMITIVE-TYPE indicating the status of the  
1716 StoreTEEPPropertyRequest operation. If the TEE property is stored successfully, the value of  
1717 status SHALL be OPERATION\_SUCCESS; otherwise its value SHALL be an error string listed in  
1718 section 5.21.1.
- 1719 • did: The value of did from the StoreTEEPPropertyRequest.
- 1720 • dsi: The DSI-CONTENT-TYPE for the new device state. This element is only returned when the  
1721 nextdsi is set to true in the StoreTEEPPropertyRequest.
- 1722 • nextnonce: A unique value that the OWE SHALL use as nonce in the next subsequent request. See  
1723 section 2.10.4 for details.

1724

### 1725 5.21.1 Error Conditions

1726 If any validation listed in section 5.20.1 fails or if a TEE error occurs, the  $rSD_{TEE}$  SHALL use an appropriate  
1727 OPERATION-RESPONSE-PRIMITIVE-TYPE (listed below) as the status value in the corresponding  
1728 response message.

1729 • ERR\_REQUEST\_INVALID

1730 • ERR\_OWE\_NOT\_TRUSTED

1731 • ERR\_OCSP\_INVALID

1732 • ERR\_TEE\_BUSY

1733 • ERR\_TEE\_FAIL

1734 • ERR\_TEE\_RESOURCE\_FULL

1735 • ERR\_TEE\_UNKNOWN

1736 • ERR\_UNSUPPORTED\_CRYPTO\_ALG

1737 • ERR\_UNSUPPORTED\_MSG\_VERSION

1738 See section 4.14 for details on error strings.

1739

## 1740 5.22 FactoryResetTBSRequest

1741 An OWE issues a `FactoryResetTBSRequest` message to move the TEE to a notional “factory” state. This  
1742 message SHALL be signed using the JWS scheme and encapsulated in a `FactoryResetRequest` message.  
1743 This message SHALL always be issued to an `rSDTEE`.

1744 The JSON structure for the `FactoryResetTBSRequest` is as follows:

```
1745 {  
1746     "FactoryResetTBSRequest" : COMMAND-PARAMETER-TYPE  
1747 }
```

1748 The following JSON element will be used as input to the `CONTENT-ENCRYPTION-TYPE` within `COMMAND-`  
1749 `TYPE`.

```
1750 {  
1751     "tsmid" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,  
1752     "did" : "PRINTABLE-STRING-PRIMITIVE-TYPE"  
1753 }
```

1754 Where:

- 1755 • `tsmid`: The identifier of the OWE that issued the request.
- 1756 • `did`: The base64 encoded SHA-256 hash of the TEE-CERT binary. `did` is used as the device  
1757 identifier to which the request is issued.

1758

## 1759 5.22.1 Processing Requirements

1760 Upon receiving the `FactoryResetRequest` message, the `rSDTEE` SHALL:

- 1761 • Validate the JSON web signature associated with the request, using the OWE-PUB associated with  
1762 the OWE-CERT.
- 1763 • Determine whether the OWE-CERT chains to a root CA certificate in its OWE-WHITELIST.
- 1764 • Check the revocation status of the OWE-CERT chain, using the cached OCSP stapling. If the cache is  
1765 unavailable or expired, the `rSD` SHALL return the corresponding response with an error string along  
1766 with an indication (`signerreq` set to `true`) to provide OCSP stapling in the next request. The OWE  
1767 may reissue the request with OCSP stapling.

1768 Upon successfully completing the above steps, the `rSDTEE` SHALL perform a factory reset on the device. All  
1769 SDs and TAs created or installed using OTrP Profile or [TMF ASN.1] that are not listed in  
1770 `gpd.tee.tmf.resetpreserved.entities` SHALL be removed. All TAs which are listed in  
1771 `gpd.tee.tmf.resetpreserved.entities` SHALL be reset according to the following requirements:

- 1772 • All data (if any) in the `TEE_STORAGE_PERSO` storage space is retained unmodified.
- 1773 • All data (if any) in the `TEE_STORAGE_PRIVATE` storage space is removed atomically.
- 1774 • All active TEE Client or TEE Internal sessions are terminated. If the administration session used to  
1775 perform the Factory Reset operation is terminated, then the factory reset SHALL continue.

1776 A response message `StoreTEEPPropertyResponse` SHALL always be returned regardless of the status of  
1777 the operation.

1778



## 1779 5.23 FactoryResetTBSResponse

1780 In response to a `FactoryResetRequest` command, the `rSDTEE` SHALL return a `FactoryResetResponse`,  
1781 encapsulating the `FactoryResetTBSResponse` message. The JSON structure for the  
1782 `FactoryResetTBSResponse` is as follows:

```
1783 {  
1784     "FactoryResetTBSResponse" : RESPONSE-PARAMETER-TYPE  
1785 }
```

1786 Within the `RESPONSE-PARAMETER-TYPE`, the following JSON structure is used as an input to the JWE while  
1787 generating `CONTENT-ENCRYPTION-TYPE`.

```
1788 {  
1789     "status" : "OPERATION-RESPONSE-PRIMITIVE-TYPE" ,  
1790     "did" : "PRINTABLE-STRING-PRIMITIVE-TYPE" ,  
1791     "dsi" : DSI-CONTENT-TYPE ,  
1792     "nextnonce" : "PRINTABLE-STRING-PRIMITIVE-TYPE"  
1793 }
```

1794 Where:

- 1795 • `status`: An `OPERATION-RESPONSE-PRIMITIVE-TYPE` indicating the status of the  
1796 `FactoryResetRequest` operation. If the TEE property is stored successfully, the value of `status`  
1797 SHALL be `OPERATION_SUCCESS`; otherwise its value SHALL be an error string in section 5.23.1.
- 1798 • `did`: The value of `did` from the `FactoryResetRequest`.
- 1799 • `dsi`: The `DSI-CONTENT-TYPE` for the new device state. This element is returned only when the  
1800 `nextdsi` is set to `true` in the `FactoryResetRequest`.
- 1801 • `nextnonce`: A unique value that the OWE SHALL use as nonce in the next subsequent request. See  
1802 section 2.10.4 for details.

### 1803 5.23.1 Error Conditions

1804 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  
1805 OPERATION-RESPONSE-PRIMITIVE-TYPE (listed below) as the status value in the corresponding  
1806 response message.

- 1807 • ERR\_REQUEST\_INVALID
- 1808 • ERR\_OWE\_NOT\_TRUSTED
- 1809 • ERR\_OCSP\_INVALID
- 1810 • ERR\_TEE\_BUSY
- 1811 • ERR\_TEE\_FAIL
- 1812 • ERR\_TEE\_RESOURCE\_FULL
- 1813 • ERR\_TEE\_UNKNOWN
- 1814 • ERR\_UNSUPPORTED\_CRYPTO\_ALG
- 1815 • ERR\_UNSUPPORTED\_MSG\_VERSION

1816 See section 4.14 for details on error strings.

1817

1818 **Annex A Changes**

1819 This annex describes changes between the original Open Trust Protocol (OTrP) v1.0 ([OTPA OTrP]) and the  
 1820 GlobalPlatform OTrP Profile described in this document.

1821 **A.1 Terminology**

1822 **Table A-1: Changes to Terminology**

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

1823

1824 **A.2 JSON Elements**

1825 **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.

Original JSON Element Name	JSON Element Name Used	Status	Notes
encrypted_ta	encrypted_ta_bin	Updated	<p>encrypted_ta used an ad hoc JSON structure to represent encrypted TA binary, and included TA personalization data.</p> <p>The encrypted_ta_bin SHALL follow CONTENT-ENCRYPTION-TYPE, which is based on the JWE format for representing encrypted data.</p> <p>TA personalization data SHALL be represented using a separate JSON element, encrypted_ta_data.</p>
n/a	encrypted_ta_data	Added	TA personalization data previously included in encrypted_ta.

1826

1827

## Annex B String Identifiers for Curves in ECC

1828

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

1829

1830

**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	

1831

## 1832 **Annex C Specification Properties**

1833 The `gpd.tee.tmf.*` properties listed in Table C-1 can be retrieved by the generic Property Access  
1834 Functions with the `TEE_PROPSET_TEE_IMPLEMENTATION` pseudo-handle (see [TEE Core]).

1835 The property `gpd.ta.parentSD` can be retrieved by a TA using these generic functions with the  
1836 `TEE_PROPSET_CURRENT_TA` pseudo-handle.

1837 The property `gpd.client.parentSD` can be retrieved by a TA (called by a client TA) using these generic  
1838 functions with the `TEE_PROPSET_CURRENT_CLIENT` pseudo-handle.

1839 The `gpd.sd.isRootSD` property of an SD is flagged internally by the TEE at SD installation time and  
1840 SHOULD NOT be retrieved using these generic functions.

1841 **Table C-1: Specification Reserved Properties**

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

1842