ETSI TS 103 666-2 V15.5.0 (2024-12)

Smart Secure Platform (SSP);
Part 2: Integrated SSP (iSSP) characteristics
(Release 15)

Reference
RTS/SET-T103666-2vf50

Keywords
M2M, MFF

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI IPR online database.

Pursuant to the ETSI Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™**, **LTE™** and **5G™** logo are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM®** and the GSM logo are trademarks registered and owned by the GSM Association.

Foreword

This Technical Specification (TS) has been produced by ETSI Technical Committee Secure Element Technologies (SET).

The contents of the present document are subject to continuing work within TC SET and may change following formal TC SET approval. If TC SET modifies the contents of the present document, it will then be republished by ETSI with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

- x the first digit:
  - 0 early working draft;
  - 1 presented to TC SET for information;
  - 2 presented to TC SET for approval;
  - 3 or greater indicates TC SET approved document under change control.
- y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.
- z the third digit is incremented when editorial only changes have been incorporated in the document.

The present document is part 2 of a multi-part deliverable covering Smart Secure Platform (SSP), as identified below:

Part 1: "General characteristics";

Part 2: "Integrated SSP (iSSP) characteristics".

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

1 Scope

The present document details the technical specifications for the Smart Secure Platform (SSP) integrated into an SoC, also known as iSSP. The present document defines specific attributes on top of the generic SSP specified in ETSI TS 103 666-1 [3].

2 References

2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

- In the case of a reference to a TC SET document, a non-specific reference implicitly refers to the latest version of that document in the same Release as the present document.

Referenced documents which are not found to be publicly available in the expected location might be found in the ETSI docbox.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are necessary for the application of the present document.

[1] ETSI TS 102 221: "Smart Cards; UICC-Terminal interface; Physical and logical characteristics"....

[2] ETSI TS 102 223: "Smart Cards; Card Application Toolkit (CAT)".

[3] ETSI TS 103 666-1: "Smart Secure Platform (SSP); Part 1: General characteristics".

[4] BSI-CC-PP-0084-2014: "Security IC Platform Protection Profile with Augmentation Packages".

[5] BSI-CC-PP-0089-2015: "Embedded UICC Protection Profile, Version 1.1 25.08.2015".

[6] GlobalPlatform Technology: "Open Firmware Loader for Tamper Resistant Element", Version 1.3.

[7] GlobalPlatform Technology: "Virtual Primary Platform - Concepts and Interfaces", Version 1.0.1.

[8] GlobalPlatform Technology: "Virtual Primary Platform - Firmware Format", Version 2.0.

[9] GlobalPlatform Technology: "Virtual Primary Platform - OFL VNP Extension", Version 1.0.

[10] IETF RFC 4122: "A Universally Unique IDentifier (UUID) URN Namespace".

[11] IETF RFC 5280: "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile".

[12] Void.

[13] NIST 800-108: "Recommendation for Key Derivation Using Pseudorandom Functions".

[14] BSI AIS 20, version 1 (02/12/1999): "Functionality Classes and Evaluation Methodology for Deterministic Random Number Generators".

[15] BSI AIS 31, version 3.1 (25/09/2001): "Functionality classes and evaluation methodology for true (physical) random number generators".

[16] Joint Interpretation Library: "Application of Attack Potential to Smartcards and Similar Devices", v3.1, June 2020.

[17] CCMB-2017-04-003: "Common Criteria for Information Technology Security Evaluation; Part 3: Security assurance components", April 2017 Version 3.1 Revision 5.

[18] NIST 800-56A Revision 2: "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography", May 2013.

[19] IETF RFC 5639: "Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation".

[20] ANSI X9.142-2020: "Financial services - Public Key Cryptography for the Financial Services Industry - The Elliptic Curve Digital Signature Algorithm - ECDSA".

[21] ISO/IEC 14888-3:2018: "IT Security techniques — Digital signatures with appendix — Part 3: Discrete logarithm based mechanisms".

[22] ISO/IEC 10118-3:2018: "IT Security techniques — Hash-functions — Part 3: Dedicated hash-functions".

[23] BSI TR-03111: "Elliptic Curve Cryptography", Version 2.10.

[24] draft-sca-cfrg-sm3-02: "The SM3 Cryptographic Hash Function".

[25] IETF RFC 7540: "Hypertext Transfer Protocol Version 2 (HTTP/2)".

[26] IETF RFC 8446: "The Transport Layer Security (TLS) Protocol Version 1.3".

[27] ETSI TS 101 220: "Smart Cards; ETSI numbering system for telecommunication application providers".

[28] Void.

[29] IETF RFC 5480: "Elliptic Curve Cryptography Subject Public Key Information".

[30] IETF RFC 5758: "Internet X.509 Public Key Infrastructure: Additional Algorithms and Identifiers for DSA and ECDSA".

[31] Recommendation ITU-T X.501 (10/2019) (ISO/IEC 9594-2:2020): "Information technology - Open Systems Interconnection - The Directory: Models".

[32] IETF RFC 4868: "Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec".

[33] NIST SP 800-38B (May 2005): "Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication".

[34] ETSI TS 102 241: "Smart Cards; UICC Application Programming Interface (UICC API) for Java Card™".

[35] ETSI TS 102 226: "Smart Cards; Remote APDU structure for UICC based applications".

[36] ISO 7816 (all parts): "Identification cards — Integrated circuit cards".

[37] IETF RFC 5754: "Using SHA2 Algorithms with Cryptographic Message Syntax".

[38] GlobalPlatform Technology: "Card Specification", Version 2.3.1.

[39] IETF RFC 4648: " The Base16, Base32, and Base64 Data Encodings".

[40] ETSI TS 123 003: "Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); LTE; 5G; Numbering, addressing and identification (3GPP TS 23.003)".

[41] ETSI TS 129 002: "Digital cellular telecommunications system (Phase 2+) (GSM); Universal Mobile Telecommunications System (UMTS); LTE; 5G; Mobile Application Part (MAP) specification (3GPP TS 29.002)".

[42] IETF RFC 3986: "Uniform Resource Identifier (URI): Generic Syntax".

[43] Recommendation ITU-T E.212: "The international identification plan for public networks and subscriptions".

[44] ISO/IEC 18033-3:2010/Amd 1:2021: "Information technology — Security techniques — Encryption algorithms — Part 3: Block ciphers — Amendment 1: SM4".

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

- In the case of a reference to a TC SET document, a non-specific reference implicitly refers to the latest version of that document in the same Release as the present document.

NOTE: While any hyperlinks included in this clause were valid at the time of publication ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] BSI-DSZ-CC-0827-V7-2018: "Security IC Platform Protection Profile", Version 1.0, 15 June 2007.

[i.2] Void.

[i.3] NIST IR 7298: "Glossary of Key Information Security Terms".

[i.4] ETSI TS 134 108: "Universal Mobile Telecommunications System (UMTS); LTE; Common test environments for User Equipment (UE); Conformance testing (3GPP TS 34.108)".

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the terms given in ETSI TS 103 666-1 [3] and the following apply:

3GPP network registration: procedure defined by 3GPP allowing a terminal to get access to services provided by telecommunication networks compliant with 3GPP specifications, using the subscription information stored within the said terminal in a SIM, a USIM or an ISIM application

address space: set of addresses that can be used by a particular program or functional unit

custodian: organization that defines family identifier specific requirements (e.g. trusted CIs, product certification) within its iSSP ecosystem

family identifier: UUID identifying a family of Secondary Platform Bundles

NOTE: It is equivalent to Firmware Family in GP OFL specification [6].

plaintext: intelligible data that has meaning and can be understood without the application of decryption (see NIST IR 7298 [i.3])

process: independent sequences of execution running within independent virtual address space and which may have shared virtual memories with other processes (e.g. virtual shared memory for communication between processes)

program: independent set of instructions executed by CPU

Secondary Platform Bundle (SPB) container: packaged code and data to create a Secondary Platform Bundle instance

Secondary Platform Bundle (SPB) image: data encapsulating an encrypted Secondary Platform Bundle container and cryptographic data to extract a Secondary Platform Bundle container

Secondary Platform Bundle (SPB) instance: runtime instance of the container, running on top of the Primary Platform Interface

Secondary Platform Bundle (SPB) loader: Secondary Platform Bundle instance with special privileges that enable managing Secondary Platform Bundle containers

Secondary Platform Bundle (SPB) management operation: operation related to the state of the Secondary Platform Bundle, including its enablement, its disablement and its deletion

Secondary Platform Bundle (SPB) provisioning: sequence of operations related to the downloading of a Secondary Platform Bundle from a SPB Manager, its loading and its installation within the iSSP

service: hardware dependent low level software running in unprivileged mode

telecom family identifier: family identifier having a reserved value, used to identify a Secondary Platform Bundle as a Telecom Secondary Platform Bundle

telecom Secondary Platform Bundle (SPB): Secondary Platform Bundle (SPB) which contains or is intended to contain at least one 3GPP NAA

test telecom bundle: telecom bundle containing a 3GPP NAA which is intended to access a 3GPP test network (e.g. a network compliant with ETSI TS 134 108 [i.4])

user intent: direct, real time acquisition and validation of the end user input on the LBA to trigger locally a Secondary Platform Bundle provisioning or a Secondary Platform Bundle management operation

virtual address: in a virtual storage system, the address assigned to a storage location in external storage (i.e. outside the SE) to allow that location to be accessed as though it were part of main storage (i.e. inside the SE)

virtual address space: set of virtual addresses that can be used by a particular program or functional unit

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the abbreviations given in ETSI TS 103 666-1 [3] and the following apply:

ABI	Application Binary Interface
API	Application Programming Interface
HLOS	High Level Operating System
iNVM	(internal NVM) Non-Volatile Memory inside the SSP
iRAM	(internal RAM) volatile Random Access Memory inside the SSP
ISN	Individual Serial Number
MMF	Memory Management Function
OID	Object IDentifier
PRF	Pseudorandom Function family
rNVM	(remote NVM) Non-Volatile Memory outside the SE
rRAM	(remote RAM) volatile Random Access Memory outside the SSP
SPB	Secondary Platform Bundle

4 Introduction

4.1 Document Layout

The present document specifies:

- an overview of the iSSP;
- the iSSP architecture;
- the Primary Platform, including the hardware platform requirements and services;
- the Primary Platform Interface;
- the Secondary Platform Bundle;
- the communication interface, including the protocol stack layers;
- the certification requirements for the iSSP.

4.2 ASN.1 syntax

4.2.1 Introduction

The provisions of ETSI TS 103 666-1 [3], clause 4.4.1 shall apply.

The complete ASN.1 code is provided for reference in Annex B.

4.2.2 Start of ASN.1

```
-- ASN1START

ISSPDefinitions { itu-t (0) identified-organization (4) etsi (0) smart-secure-platform (3666) part2
(2) }
DEFINITIONS
AUTOMATIC TAGS
EXTENSIBILITY IMPLIED ::=
BEGIN

/* Imports */

IMPORTS
    Certificate, Time, AlgorithmIdentifier
        FROM PKIX1Explicit88 {iso(1) identified-organization(3) dod(6) internet(1) security(5)
mechanisms(5) pkix(7) id-mod(0) id-pkix1-explicit(18)}
    SubjectKeyIdentifier
        FROM PKIX1Implicit88 {iso(1) identified-organization(3) dod(6) internet(1) security(5)
mechanisms(5) pkix(7) id-mod(0) id-pkix1-implicit(19)};

-- ASN1STOP
```

5 Overview

5.1 Description

An iSSP is an integrated SSP confined in a dedicated sub-system within an SoC. The SoC is usually soldered in the terminal and so the SSP is an integral part of the terminal.

The iSSP is a composition of three parts as described in clause 6.1.

5.2 Security requirements

The provisions of ETSI TS 103 666-1 [3], clause 6.11 shall apply.

The software and sensitive data of the iSSP shall never be exposed from the iSSP to any external component in plaintext. The protection of software and sensitive data shall provide privacy, confidentiality, integrity, protection against rollback attacks, and protection against side-channel attacks.

In the case where software and sensitive data are stored outside the iSSP, they shall also be protected in a way to achieve perfect forward secrecy and they shall be securely bound to that given iSSP instance, in accordance with clause 7.1.3.4.

5.3 References to GlobalPlatform

The present document contains several references to GlobalPlatform VPP - Concepts and Interfaces [7] specification.

The following list maps the terms used by Global Platform VPP - Concepts and Interfaces [7] to the terms used in the present document:

- VPP Application and Firmware, depending on the context, refer to the Secondary Platform Bundle.
- The group of interfaces that include Kernel Functions ABI/API, Communication Service and Firmware Management Service are collectively referred to as the Primary Platform Interface.
- HLOS Applications refers to SSP Applications.
- Tamper Resistant Element (TRE) refers to the iSSP.

6 iSSP Architecture

6.1 Overview

The iSSP consists of:

1) The Primary Platform including the hardware platform and a low-level operating system.
2) The Primary Platform Interface, which provides the abstraction and virtualization of the Primary Platform.
3) Secondary Platform Bundle(s), which contains the Secondary Platform along with its SSP applications:
    a) The Secondary Platform, a software component using the Primary Platform Interface, which contains a high-level operating system and may contain execution framework(s).
    b) The SSP Application(s) that run on top of the Secondary Platform.

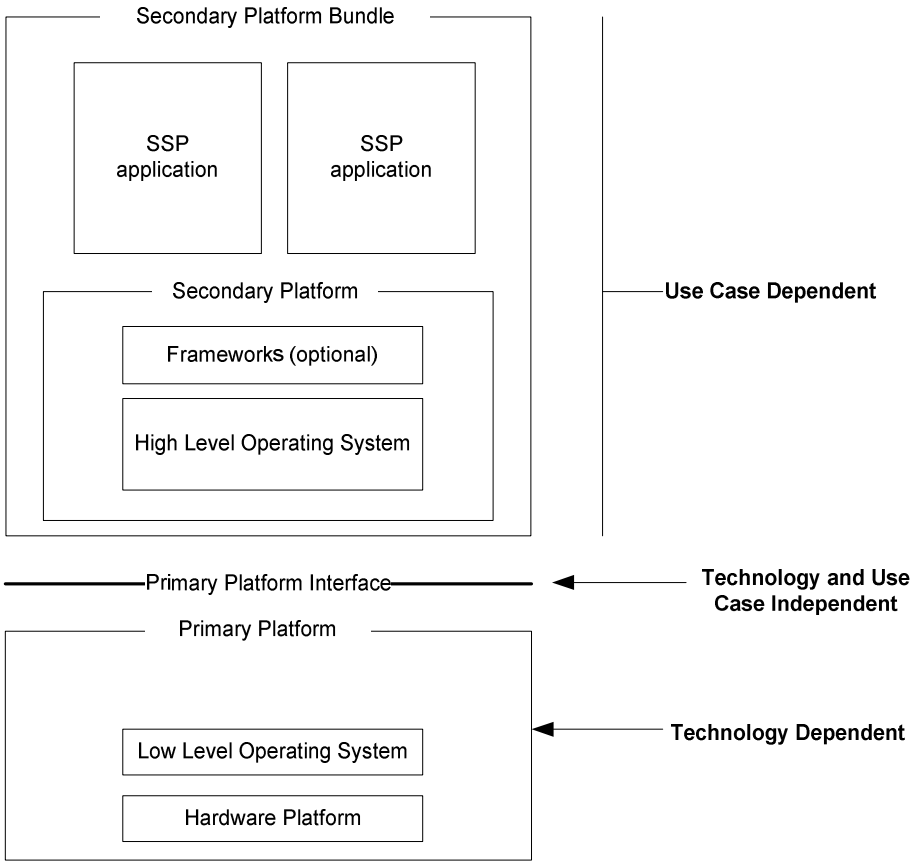

Figure 6.1: iSSP architecture

The provisions of GlobalPlatform VPP - Concepts and Interfaces [7], clause 2.1 shall apply.

6.2 Functional architecture

The provisions of GlobalPlatform VPP - Concepts and Interfaces [7], clauses 5.1 and 5.2 shall apply.

6.3 Security perimeters

The provisions of GlobalPlatform VPP - Concepts and Interfaces [7], clause 5.3 shall apply.

6.4 Unprivileged execution model

The provisions of GlobalPlatform VPP - Concepts and Interfaces [7], clause 5.4 shall apply.

6.5 Unprivileged virtual address space

The provisions of GlobalPlatform VPP - Concepts and Interfaces [7], clause 5.5 shall apply.

6.6 Run time model

The provisions of GlobalPlatform VPP - Concepts and Interfaces [7], clause 5.6 and its subclauses shall apply.

7 Primary Platform

7.1 Hardware Platform

7.1.1 Architecture

The provisions of GlobalPlatform VPP - Concepts and Interfaces [7], clause 3.1 shall apply with the following exceptions:

- the presence of the SoC is mandatory in figure 3-1 in GlobalPlatform VPP - Concepts and Interfaces [7];
- the iSSP shall contain an autonomous and independent clock system;
- the iSSP shall contain communication functions;
- the iSSP may contain the data protection hardware function.

7.1.2 Form factor

No form factor is mandated.

7.1.3 Security functions

7.1.3.1 Hardware Platform isolation

The provisions of GlobalPlatform VPP - Concepts and Interfaces [7], clause 3.4.1 SREQ19 shall apply.

7.1.3.2 Memory Management Function

The Primary Platform shall provide a Memory Management Function (MMF) to avoid dependency of the Secondary Platform Bundle design with respect to the execution memory addressing.

The provisions of GlobalPlatform VPP - Concepts and Interfaces [7], clauses 3.2.2 and 3.5 shall apply.

7.1.3.3 Key protection function

The hardware platform shall provide a hardware function for protecting its long term keys as defined in GlobalPlatform VPP - Concepts and Interfaces [7], clause 3.4.3.

The key protection function shall perform key derivation, as specified in NIST Special Publication 800-108 [13], with robustness of the PRF equivalent to or greater than HMAC-SHA-256 [32] or CMAC [33].

The long term seed value shall be accessible only by the hardware platform. The probability that two distinct hardware platforms have the same long term seed shall be negligible. For example, the long term seed is generated randomly by the hardware platform and is not known to the maker of the Primary Platform.

The hardware platform shall provide a data path for the key protection function output. The key protection function output shall be made available for the data protection hardware function described in clause 7.1.3.4, if that clause is supported, or to the cryptographic functions described in clause 7.1.7.

7.1.3.4 Data protection hardware function

The hardware platform may support a hardware function performing the encryption to export software and data outside the iSSP, and the decryption to import software and data from outside the iSSP.

The hardware platform may support a hardware function performing the integrity computation and check of software and data stored outside the iSSP.

These hardware functions shall only be accessible by the low-level Operating System in the Primary Platform. For the purpose of storing and verifying software and data outside the iSSP, these hardware functions shall only use the keys provided by the key protection function.

If these hardware functions are supported, the provisions of GlobalPlatform VPP - Concepts and Interfaces [7], clause 3.4.2 shall apply.

7.1.3.5 Memory transfer function

The provisions of GlobalPlatform VPP - Concepts and Interfaces [7], clause 3.2.6 shall apply.

7.1.3.6 Test functions

The hardware platform shall protect against the disclosure of keys managed by the Primary Platform, when using test functions of the SoC or test equipment.

7.1.3.7 Remote audit

The provisions of GlobalPlatform VPP - Concepts and Interfaces [7], clause 3.7 shall apply.

7.1.3.8 Security sensor function

The provisions of GlobalPlatform VPP - Concepts and Interfaces [7], clause 3.4.4 shall apply.

7.1.4 Memories

7.1.4.1 Non Volatile Memories

The provisions of GlobalPlatform VPP - Concepts and Interfaces [7], clause 3.2.3 shall apply.

The Primary Platform shall provide the Secondary Platform with direct memory-mapped access to the NVM, whether the NVM is integrated in the iSSP (iNVM) or accessed remotely (rNVM).

In case of rNVM, the requirements for robustness of the algorithms used for the data protection are specified in clause 7.1.3.4.

7.1.4.2 Volatile memory

The Primary Platform shall provide the Secondary Platform with direct memory-mapped access to the volatile memory, whether the memory is integrated in the iSSP (iRAM) or accessed remotely (rRAM).

In case of rRAM, the requirements for robustness of the algorithms used for the data protection are specified in clause 7.1.3.4.

7.1.5 Communication functions

The physical communication interfaces between the Primary Platform and the SoC it is integrated with is outside the scope of the present document. These are abstracted from the Secondary Platform by the interface defined in clause 8.2.

7.1.6 Power

Due to the integrated nature of the iSSP, the power supply, the management of the power states and the overall power consumption of the iSSP are Primary Platform specific and are outside the scope of the present document.

7.1.7 Cryptographic functions

The provisions of GlobalPlatform VPP - Concepts and Interfaces [7], clause 3.2.7 shall apply.

7.1.8 Clock

The iSSP shall embed an autonomous and independent clock system in conformance with the Protection Profile BSI-CC-PP-0084-2014 [4].

The provisions of ETSI TS 103 666-1 [3], clause 6.3 shall apply.

7.1.9 SSP internal interconnect

All elements contained in the iSSP shall only be physically connected to other elements in the iSSP, except as specified in clause 7.1.5.

7.1.10 Secure CPU

The hardware platform shall contain one or more dedicated CPUs, which are inside the iSSP and separated from the rest of the SoC.

The provisions of GlobalPlatform VPP - Concepts and Interfaces [7], clause 3.2.1 shall apply. The CPU(s) shall be based at least on a 32-bit architecture.

The characteristics of the CPUs (e.g. endianness) are implementation dependent and outside the scope of the present document.

7.1.11 Random Number Generator

The provisions of GlobalPlatform VPP - Concepts and Interfaces [7], clause 3.2.8 shall apply.

7.2 Low-level Operating System

7.2.1 Introduction

The Primary Platform includes a low-level Operating System.

The provisions of GlobalPlatform VPP - Concepts and Interfaces [7], clause 5.8 (without its subclauses) shall apply.

7.2.2 Kernel objects

The provisions of GlobalPlatform VPP - Concepts and Interfaces [7], clause 5.8.1 shall apply.

7.2.3 Global requirements and mandatory Access Control rules

The provisions of GlobalPlatform VPP - Concepts and Interfaces [7], clause 5.8.2 shall apply.

7.2.4 Process states diagram

The provisions of GlobalPlatform VPP - Concepts and Interfaces [7], clause 5.8.3 shall apply.

7.2.5 Definition of the process states

The provisions of GlobalPlatform VPP - Concepts and Interfaces [7], clause 5.8.4 shall apply.

7.2.6 Mandatory access control

The provisions of GlobalPlatform VPP - Concepts and Interfaces [7], clause 5.11 and its subclauses shall apply.

The low-level operating system shall only have non-shareable memory regions.

7.3 Services

7.3.1 Secondary Platform Bundle Loader

7.3.1.1 Overview

The Primary Platform shall support a Secondary Platform Bundle Loader as defined in GlobalPlatform Open Firmware Loader for Tamper Resistant Element [6] with the following exceptions:

- The OFL ARP state shall be UNLOCKED.

The Secondary Platform Bundle Loader shall be a system Secondary Platform Bundle and shall support the requirements defined in the augmented package loader 2 in BSI-CC-PP-0084-2014 [4].

The Secondary Platform Bundle Loader shall support the format defined in clause 9.3.

7.3.1.2 Registries

The Secondary Platform Bundle Loader shall implement at least the following registry entries of the OFL service gate defined in GlobalPlatform VPP - OFL VNP Extension [9].

Table 7.1: Registry entry in the OFL Service Gate

Type	Identifier	Parameter	Access right	Comment	Length
Mandatory	'01'	CURR_VERSION	RO	As defined in GlobalPlatform VPP - OFL VNP Extension [9].	2
Mandatory	'03'	TRE_CREDENTIAL_PARAMETER	RO	This registry shall be used by the LBA to retrieve SSP credential data structure defined in clause 12.6.2.4.	Variable
Mandatory	'04'	IDS_CREDENTIAL_PARAMETER	WO	This registry shall be used by the LBA to provide SPBM credential data structure defined in clause 12.6.2.3.	Variable
Mandatory	'09'	OPERATION_TOKEN	RO	This registry shall be used by the LBA to retrieve a notification token as defined in clause 12.6.2.8.	Variable
Mandatory	'0D'	UUID_IMAGE_LIST	RO	As defined in GlobalPlatform VPP - OFL VNP Extension [9].	Variable

In addition, the Secondary Platform Bundle Loader shall implement the following registry entries.

Table 7.2: Additional registry entry in the OFL Service Gate

Type	Identifier	Parameter	Access Right	Comment	Length	Default
Conditional	'80'	TELECOM_CAPABILITY	RO	Maximum number of concurrent 3GPP network registrations.	1	1
Mandatory	'81'	GET_SSP_INFO_PARAMETER	WO	It contains family identifier and/or custodian identifier used to generate SSP_INFO_PUBLIC as defined in clause 12.6.5.5.1.	Variable	-
Mandatory	'82'	SSP_INFO_PUBLIC	RO	Public SSP Information as defined in clause 12.6.2.2.2.	Variable	-
Mandatory	'83'	PP_IDENTIFIER	RO	Primary Platform Identifier as defined in clause 7.5.	20	-
Mandatory	'84'	SPB_ID	WO	Secondary Platform Bundle identifier as defined in clause 9.4.5.	16	-
Mandatory	'85'	SPB_STATE	RO	State of the Secondary Platform Bundle identified by SPB_ID.	1	-

If the iSSP contains or is intended to contain at least one Telecom Secondary Platform Bundle, TELECOM_CAPABILITY shall be set at the time of manufacturing. It shall contain the maximum number of distinct concurrent 3GPP network registrations based on different subscriber identifiers, supported by the terminal.

7.3.1.3 Commands

The Secondary Platform Bundle Loader shall support the commands defined in GlobalPlatform VPP - OFL VNP Extension [9].

In addition, the Secondary Platform Bundle Loader shall support the following additional commands.

Table 7.3: Additional commands supported by the OFL Service Gate

Value of Instruction	Command	Description
'80'	GET_SSP_INFO	The command to get Public SSP information
'81'	GET_SPB_METADATA	The command to get SPB metadata
'82'	SWITCH_TELECOM_SPB	The command to switch Telecom SPBs

The GET_SSP_INFO command shall be used to implement the function defined in clause 12.6.5.5.1.

The GET_SPB_METADATA command shall be used to implement the function defined in clause 12.6.5.5.10.

The SWITCH_TELECOM_SPB command shall be used to implement the function defined in clause A.6.2.

7.3.1.4 Responses

The Secondary Platform Bundle Loader shall support the responses defined in GlobalPlatform VPP - OFL VNP Extension [9].

In addition, the OFL service gate shall support the following responses:

Table 7.4: Additional responses supported by the OFL Service Gate

Value	Response	Description
'10'	eSPBL_E_NO_CI_FOR_SPBM_VERIFICATION	The SSP does not support any CI for signature verification as per the given GET_SSP_INFO_PARAMETER.
'11'	eSPBL_E_NO_CI_FOR_SPBL_VERIFICATION	The SSP does not support any CI for signing as per the given GET_SSP_INFO_PARAMETER.
'12'	eSPBL_E_NO_CI_FOR_KEYAGREEMENT	The SSP does not support any CI for key agreement as per the given GET_SSP_INFO_PARAMETER.
'13'	eSPBL_E_NO_SUPPORTED_CRYPTO	The SSP does not support any cryptographic algorithm as per the given GET_SSP_INFO_PARAMETER.
'14'	eSPBL_E_INVALID_SPBM_CERT	The indication that a received SPBM certificate (chain) is not valid, given as response to any related command supported by the Secondary Platform Bundle Loader.
'15'	eSPBL_E_EXCEED_TELECOM_CAPABILITY	The Telecom Secondary Platform Bundle cannot be enabled due to the TELECOM_CAPABILITY limit.

7.3.1.5 Firmware session

The Secondary Platform Bundle Loader shall manage firmware sessions as defined in GlobalPlatform Open Firmware Loader for Tamper Resistant Element [6], section 2.2.1 as per the Secondary Platform Bundles installed in the iSSP.

In addition, the Secondary Platform Bundle Loader shall manage the following values as additional parameters of the firmware session as per the Secondary Platform Bundles:

- notification counter: the 4 bytes integer value which is used when the Secondary Platform Bundle Loader generates the notification token as defined in clause 12.6.2.8. The notification counter of the Secondary Platform Bundle shall be pre incremented by one by Secondary Platform Bundle Loader at each generation of a token. The initial value of the counter is '1';
- the Secondary Platform Bundle private identifier as defined in clause 9.4.5;
- SPB metadata as defined in clause 12.6.2.6 of the Secondary Platform Bundle container. When the firmware session is created, the SPB metadata contained in the bound SPB image shall be stored;
- SPB state: the current state of the Secondary Platform Bundle. The value shall be one of 'Disabled (0)' and 'Enabled (1)'.

7.3.2 Communication service

The Primary Platform shall provide communication service for the use of the Secondary Platform Bundle to communicate with entities outside the iSSP. The interface is defined in clause 8.2.

7.3.3 Management service

The Primary Platform shall provide management service for the exclusive use of the Secondary Platform Bundle Loader.

The management service provides the interface to manage:

- the life cycle of a Secondary Platform Bundle; and
- the installation and management of a Secondary Platform Bundle by a Secondary Platform Bundle Loader.

The interface is defined in clause 8.3.

7.4 Minimum level of interoperability

The provisions of GlobalPlatform VPP - Concepts and Interfaces [7], clause 7 and its subclauses shall apply.

7.5 Primary Platform identification

The Primary Platform instance is identified by a Primary Platform identifier. The Primary Platform identifier is a sequence of 32 characters, divided in 8 groups of 4 characters each, with a dash between each group.

The Primary Platform identifier shall not be changed irrespective of the Firmware update of the Secondary Platform Bundle Loader.

The 32 characters are computed using the Base32 algorithm defined in IETF RFC 4648 [39] from a sequence of 20 bytes, composed as follows:

- the first 4 bytes are defined as follows:

<table>
<tr><th colspan="8">1st byte</th><th colspan="8">2nd byte</th><th colspan="8">3rd byte</th><th colspan="8">4th byte</th></tr>
<tr><td>b8</td><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b8</td><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b8</td><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b8</td><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td></tr>
<tr><td colspan="3" rowspan="2">SSP type</td><td colspan="13">SSP manufacturer identifier</td><td colspan="6">Manufacturer proprietary information</td><td colspan="2" rowspan="2">0</td><td rowspan="2">0</td></tr>
<tr><td colspan="5">1st character</td><td colspan="5">2nd character</td><td colspan="5">3rd character</td><td colspan="4">1st character</td><td colspan="4">2nd character</td></tr>
</table>

Where:

- **SSP Type:** identifies the type of SSP and is coded as follows:

Base-32 character	Value	Description
I	'01000'	Integrated SSP (iSSP)
NOTE: All the other values are RFU.		

- **SSP Manufacturer identifier:** identifies the SSP maker which manufactures and provides the iSSP containing that Primary Platform instance and is coded as follows:

Base-32 characters	Value	Description
AAA to 777	'00000 00000 00000' to '11111 11111 11111'	RFU

- **Manufacturer proprietary information:** is a field under the responsibility of the SSP manufacturer.

NOTE: This field may contain information about the Primary Platform.

- The next 16 bytes are a UUID computed using UUID version 5 as defined in IETF RFC 4122 [10] from the string "urn:etsi.org:ppid: <SSP maker OID>:<ISN>" (without quotes), where:
  - **<SSP maker OID>** is the OBJECT IDENTIFIER of the SSP maker which manufactures and provides the iSSP containing that Primary Platform instance.
  - **<ISN>** is the individual serial number of the iSSP and is defined by the SSP maker. <ISN> shall be unique and immutable within the scope of the SSP maker. <ISN> may be based on SN as defined in GlobalPlatform VPP - OFL VNP Extension [9], clause 2.3.2.

7.6 Provisioning of Primary Platform software

The provisions of GlobalPlatform VPP - Concepts and Interfaces [7], clause 5.7 REQ85 shall apply.

7.7 Part Number Identifier

The Part Number identifier is the identifier of the Chip Part Number as defined in GlobalPlatform Open Firmware Loader for Tamper Resistant Element [6]. The Part Number identifier is a UUID computed from a string concatenating SSP maker URI and Part Number (PN) using IETF RFC 4122 UUID version 5 [10].

It shall be possible to retrieve the following information using the Part Number identifier:

- the Primary Platform manufacturer;
- the model of the Primary Platform;
- the assurance level of the Primary Platform and the Secondary Platform Bundle Loader.

8 Primary Platform Interface

8.1 Kernel functions ABI/API

The provisions of GlobalPlatform VPP - Concepts and Interfaces [7], clause 5.8.5 shall apply.

8.2 Communication service interface

The provisions of GlobalPlatform VPP - Concepts and Interfaces [7], clause 5.9 shall apply.

8.3 Secondary Platform Bundle management service interface

The provisions of GlobalPlatform VPP - Concepts and Interfaces [7], clause 5.10 and its subclauses shall apply with the following exceptions:

- The states of the Secondary Platform Bundle are described in clause 9.2 (the additional Active and Deleted states are not relevant for this Secondary Platform Bundle management service interface).

9 Secondary Platform Bundle

9.1 Introduction

The Secondary Platform Bundle refers to either the data container or the runtime instance of the container.

9.2 States

The states of the Secondary Platform Bundle are split among container and instance.

The states and transitions of the Secondary Platform Bundle container shall be as defined in GlobalPlatform VPP - Concepts and Interfaces [7], clause 5.10 and its subclauses. The state of the Secondary Platform Bundle instance shall be as defined in GlobalPlatform VPP - Concepts and Interfaces [7], clause 6.3.

The Secondary Platform Bundle has four states:

- **Active:** the Secondary Platform is running. The Secondary Platform shall restore its context and the context of its SSP Applications when entering the active state. The Secondary Platform shall save its context and the context of its SSP Applications before leaving the active state. This state is only applicable to Secondary Platform Bundle instance.

NOTE 1: To ensure the correct operation of an SSP with multiple enabled Secondary Platform Bundles, the Secondary Platform Bundle should not remain in Active state for longer than it is required to complete its tasks.

- **Enabled:** the Secondary Platform can be activated. This state is only applicable to Secondary Platform Bundle container.

- **Disabled:** the Secondary Platform cannot be activated. This state is only applicable to Secondary Platform Bundle container.
- **Deleted:** the Secondary Platform Bundle and all its data are permanently deleted from the iSSP. This is a final state, which is only applicable to a Secondary Platform Bundle container. The tracking of deleted Secondary Platform Bundles is out of scope of the present document.

The Secondary Platform Bundle Loader manages the states of the Secondary Platform Bundle container, and not the states of the Secondary Platform Bundle instance.

The mechanism to make sure that the Secondary Platform Bundle Loader can disable an active Secondary Platform Bundle is the host arbitration process described in ETSI TS 103 666-1 [3], clause 8.2.

Figure 9.1 shows the transitions among the states of the Secondary Platform Bundle.

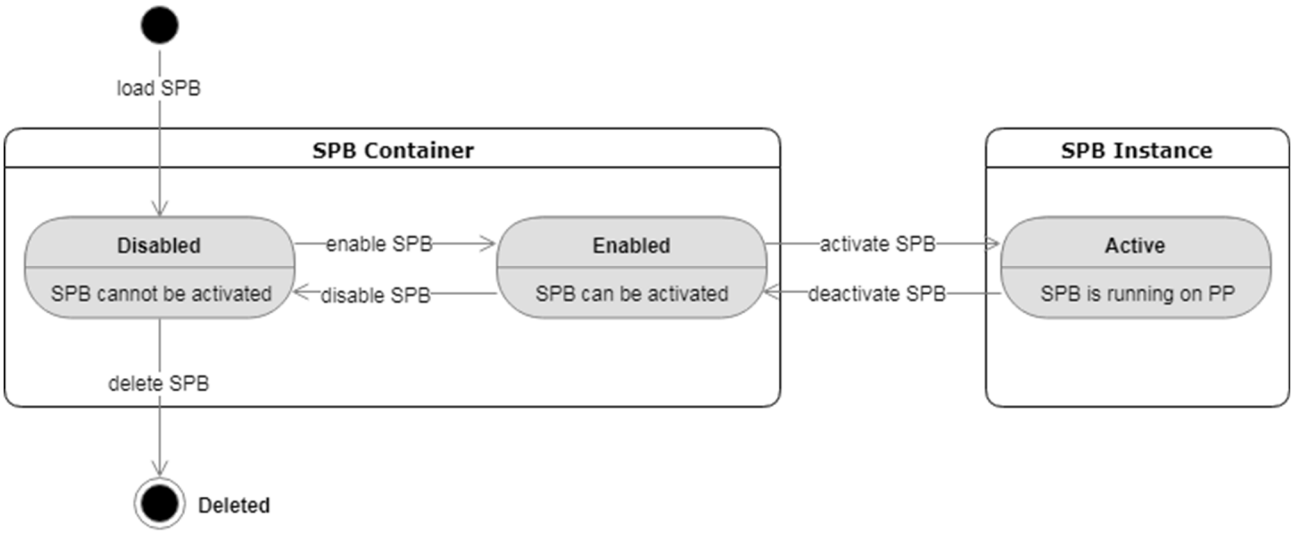

Figure 9.1: Secondary Platform Bundle states

The context of the Secondary Platform Bundle and of its applications is valid only in the Enabled and Active states. The context is created when the Secondary Platform Bundle moves to Active state for the first time. If the state of a Secondary Platform Bundle changes from Disabled to Enabled, upon the next state change to Active, the Secondary Platform Bundle shall erase its context and the context of its SSP Applications.

NOTE 2: The Secondary Platform Bundle may identify the transition from the Disabled to Enabled state by detecting a change in its host identifier, which is generated for each host registration.

No more than a single Secondary Platform Bundle shall be in an Active state.

9.3 Secondary Platform Bundle container format

The provisions of GlobalPlatform VPP - Firmware Format [8] shall apply.

9.4 Secondary Platform

9.4.1 High-level OS

The Secondary Platform Bundle contains a High Level OS as defined in GlobalPlatform VPP - Concepts and Interfaces [7], clause 6.4 and its subclauses.

9.4.2 Execution framework

The provisions of ETSI TS 103 666-1 [3], clause 5.4 shall apply.

9.4.3 UICC platform as a Secondary Platform

The Secondary Platform may emulate a UICC platform as defined in ETSI TS 102 221 [1] and ETSI TS 102 223 [2]. In this case the Secondary Platform shall support the UICC APDU gate, as described in ETSI TS 103 666-1 [3], clause 10.2.8.2 and the UICC specific mechanisms defined in ETSI TS 103 666-1 [3], clauses 5.5, 6.6.1, 6.8.1, 6.10 and 10.2.

9.4.4 Capability exchange

The data field sent by the terminal to the iSSP during the capability exchange procedure shall contain the data structure defined in ETSI TS 103 666-1 [3], clause 6.4.2.4, with the following modifications:

- aPhysicalInterfaces in the TerminalCapability should not be present. If present, aPhysicalInterfaces shall be ignored.

The data field sent by the iSSP to the terminal during the capability exchange procedure shall contain the data structure defined in ETSI TS 103 666-1 [3], clause 6.4.2.5, with the following modifications:

- SSPClass field shall have the eSSPClass-Integrated (0) value.
- aClassSpecificCapabilities, aPhysicalInterfaces and aSspExternalMaxPowerConsumption in the SSPCapability should not be present. If present, aClassSpecificCapabilities, aPhysicalInterfaces and aSspExternalMaxPowerConsumption shall be ignored.

9.4.5 Identifiers of Secondary Platform Bundle

The Secondary Platform Bundle shall be provided with two identifiers:

- The Secondary Platform Bundle identifier (i.e. SpbId), UUID which is the same as the public UUID of a firmware as defined in GlobalPlatform Open Firmware Loader for Tamper Resistant Element [6].
- The Secondary Platform Bundle private identifier (i.e. PrivateSpbId), UUID which is the same as the private UUID of a firmware as defined in GlobalPlatform Open Firmware Loader for Tamper Resistant Element [6]; where the NSS used to create the UUID shall have an entropy in its of at least 32 bytes.

9.5 SSP Application

9.5.1 Overview

The provisions of ETSI TS 103 666-1 [3], clause 6.10 apply.

9.5.2 Lifecycle management

Different lifecycle management rules apply to different types of Secondary Platform Bundles.

After the Secondary Platform Bundle has been loaded by the Secondary Platform Bundle Loader, the Secondary Platform Bundle and its SSP Applications are in their initial lifecycle state:

- for a Secondary Platform Bundle supporting the legacy execution framework defined in ETSI TS 102 241 [34], the initial lifecycle state is determined by the Secondary Platform Bundle maker;
- for a Secondary Platform Bundle supporting a native application, the initial lifecycle state is determined by the Secondary Platform Bundle maker;
- for a Secondary Platform Bundle supporting a new type of execution framework, the initial lifecycle state is for future study.

After the Secondary Platform Bundle has been enabled by the Secondary Platform Bundle Loader, the Secondary Platform Bundle internal lifecycle states and their management shall be governed by the following rules:

- for a Secondary Platform Bundle supporting the legacy framework defined in ETSI TS 102 241 [34], the rules and mechanisms for the management of the lifecycle of the Security Domains and Applications shall be compliant with the GlobalPlatform Card Specification [38] and ETSI TS 102 226 [35];
- for a Secondary Platform Bundle supporting native SSP Applications, the rules and mechanisms for the management of the lifecycle state of the SSP application(s) are proprietary and out of the scope of the present document;
- for a Secondary Platform Bundle supporting a new type of execution framework, the rules and mechanisms for the management of the lifecycle states of the SSP application(s) are for future study.

NOTE: The rules for the lifecycle states of SSP Applications which are based on the legacy framework defined in ETSI TS 102 241 [34] are based on the assumption that the Secondary Platform Bundle contains a single Java Card™ VM and the associated CAT Runtime Environment (i.e. it is equivalent to a single virtual UICC). Scenarios with multiple instances of CAT Runtime Environments are for future study.

9.6 Lifecycle management of Secondary Platform Bundles

Secondary Platform Bundles are managed by a Secondary Platform Bundle Loader, as defined in clause 7.3.1.

The Secondary Platform Bundle Loader shall enforce that the number of Telecom Secondary Platform Bundles in the Enabled or Active state (see clause 9.2) is not greater than the TELECOM_CAPABILITY parameter value defined in clause 7.3.1.

9.7 Secondary Platform Bundle family identifier

Secondary Platform Bundles are classified into families according to the main use cases addressed, e.g. telecommunication, banking.

A family of Secondary Platform Bundles is identified by a family identifier. A family identifier is a UUID computed from a URN using IETF RFC 4122 [10] UUID version 5. It is equivalent to the Firmware Family in GlobalPlatform Open Firmware Loader for Tamper Resistant Element [6].

10 Communication interface

10.1 Low level protocol layers

10.1.1 Physical layer

The physical layer is specific to the Primary Platform and outside the scope of the present document.

10.1.2 Link layer

The data link layer is specific to the Primary Platform and outside the scope of the present document, as far as it meets the minimum set of requirements defined in ETSI TS 103 666-1 [3], clause 8.3.1.

10.2 SSP Common Layer

The iSSP shall support the SCL protocol layer, as defined in ETSI TS 103 666-1 [3], clause 8.

The SCL router and the network controller host shall share the same security perimeter as the Primary Platform in order to ensure the correct mapping of host aliases with the corresponding SCL host.

Each Secondary Platform Bundle is responsible for the implementation of the SCL protocol as needed for its operation.

10.3 Communication layers above SCL

The definition and usage of protocols layers above the SCL (e.g. ISO 7816 [36] APDUs) are defined in ETSI TS 103 666-1 [3], clause 10.

11 Certification

11.1 Introduction

The iSSP shall be able to support a certification by composition of a Secondary Platform Bundle from the Primary Platform certification.

11.2 Primary Platform certification

11.2.1 Overview

The provisions of GlobalPlatform VPP - Concepts and Interfaces [7], clause 4 shall apply. The certification of the Primary Platform shall include the Loader Package 2, as defined in BSI-CC-PP-0084-2014 [4].

11.2.2 Security Capabilities

A Primary Platform may accommodate different security features and may meet different security levels depending on the targeted applications. The different security levels are evaluated by independent evaluation labs based on industry specific Protection Profiles.

The Evaluation Assurance Level (EAL) is defined in table 1 in Common Criteria for Information Technology Security Evaluation Part 3 [17] according to the level of the different Assurance Families and Assurance Classes. The Evaluation Assurance Level may be augmented if one or more Assurance Families exceed their minimal level.

The following description defines the security capability data elements.

```
-- ASN1START

AssuranceFamily ::= ENUMERATED
{
    eADV-ARC,
    eADV-FSP,
    eADV-IMP,
    eADV-INT,
    eADV-SPM,
    eADV-TDS,  -- Development
    eAGD-OPE,
    eAGD-PRE,  -- Guidance Document
    eALC-CMC,
    eALC-CMS,
    eALC-DEL,
    eALC-DVS,
    eALC-FLR,
    eALC-LCD,
    eALC-TAT,  -- Life-Cycle support
    eASE-CCL,
    eASE-ECD,
    eASE-INT,
    eASE-OBJ,
    eASE-REQ,
    eASE-SPD,
    eASE-TSS,  -- Security target evaluation
    eATE-COV,
    eATE-DPT,
    eATE-FUN,
    eATE-IND,  -- Tests
```

```
    eAVA-VAN  -- Vulnerability assessment
}

AssuranceLevel ::= INTEGER (1..7)

AssuranceComponent ::= [PRIVATE 16] SEQUENCE
{
    aAssuranceFamily AssuranceFamily,  --Assurance family
    aAssuranceLevel AssuranceLevel  --Assurance level
}

EAL ::= [PRIVATE 17] SEQUENCE
{
    aAssuranceLevel AssuranceLevel,
    aAugmented SEQUENCE (SIZE (1..27)) OF AssuranceComponent OPTIONAL
}

Capability ::= [PRIVATE 18] BIT STRING  --Capabilities
{
    aAIS (0),  -- random numbers generated in accordance with AIS20 [14] and AIS31 [15].
    aPP0084 (1),  -- Support of the BSI PP0084-2014 with augmented packages.
    aLoader2 (2)  -- Support of the package loader 2
} (SIZE(1..8))

CertificateIdentifier ::= [PRIVATE 19] IA5String (SIZE (1..27))

CertificateExpDate   ::= [PRIVATE 20] DATE

URNCertificateBody ::= [PRIVATE 21] IA5String (SIZE (1..64))

-- ASN1STOP
```

The following security capability items are defined in the data element Capability:

- aAIS: random numbers generated in accordance with AIS20 [14] and AIS31 [15].
- aPP0084: support of the BSI PP0084-2014 [4] with augmented packages in Security IC Platform BSI Protection Profile 2014 with Augmentation Packages.
- aLoader2: support of the package loader 2 in Security IC Platform BSI Protection Profile 2014 with Augmentation Packages [4].

AVA_VAN.5 tests shall be performed in accordance with the JIL Application of Attack potential to Smartcards documentation [16].

```
-- ASN1START

SecurityCapability ::= [APPLICATION 16] SEQUENCE
{
    aEAL EAL,  -- Evaluation security level
    aCapability Capability,  -- Capability
    aCertificateIdentifier CertificateIdentifier,  -- Certificate identifier
    aURNCertificateBody URNCertificateBody,  -- URN of the certification body
    aCertificateExpDate CertificateExpDate  -- Expiration date of the certificate
}

-- ASN1STOP
```

Where:

- aEAL: contains the Evaluation Assurance level with optional augmented components.
- aCertificateIdentifier: the unique identifier of the certificate which has been granted to the primary platform (e.g. BSI-DSZ-CC-0827-V7-2018 [i.1]). The value is represented as a string and valid in the scope of the certification body.
- aURNCertificateBody: URN of the certification body.
- aCertificateExpDate: expiration date of the certificate of the primary platform defined by the certification body.

11.3 Secondary Platform Bundle certification

The composite certification of the Secondary Platform Bundle, including the high-level OS and its applications, may claim in its Security Target the conformance with Protection Profile BSI-CC-PP-0089-2015 [5].

The certification of the Secondary Platform Bundle should be performed on a Primary Platform certified as defined in clause 11.2.

12 iSSP ecosystem and interfaces

12.1 General architecture

12.1.1 Introduction

This clause specifies the entities and interfaces associated with the management (e.g. loading, enabling, disabling and deleting) of a Secondary Platform Bundle container.

12.1.2 Architecture overview

Figure 12.1 shows the iSSP ecosystem.

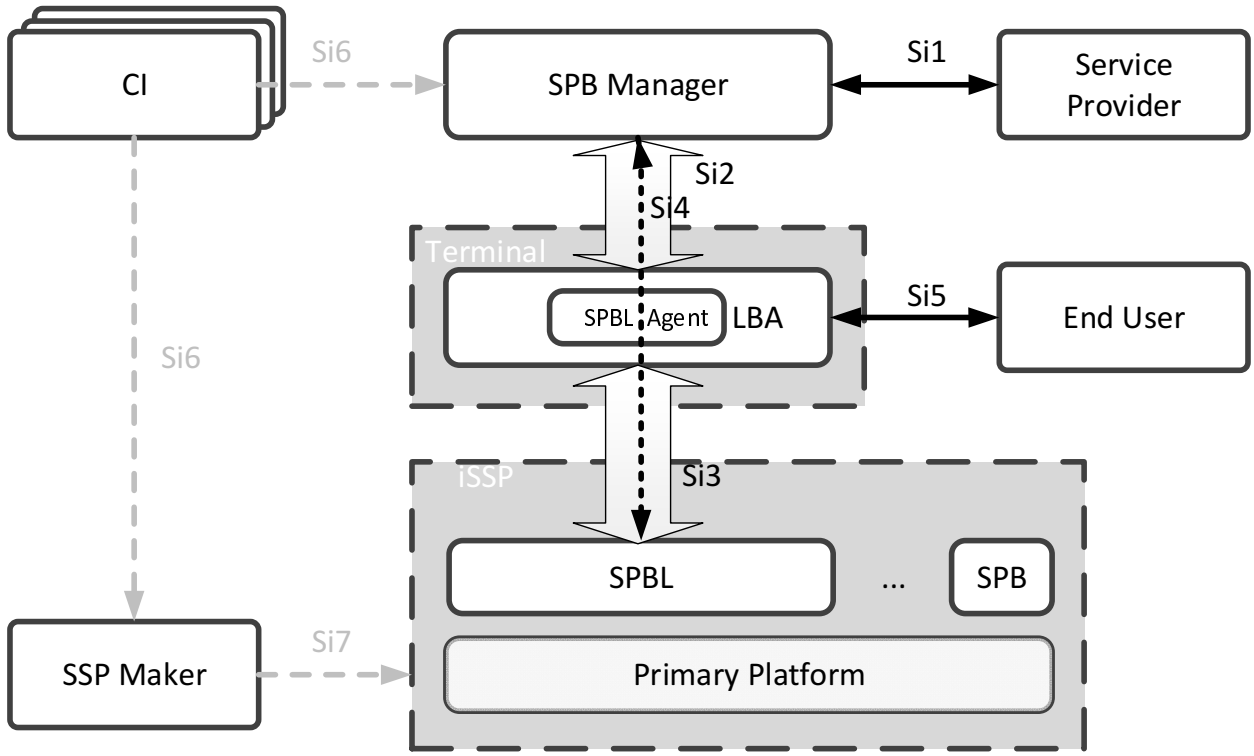

Figure 12.1: Entities and interfaces for Secondary Platform Bundle management

12.1.3 Entities

The Secondary Platform Bundle Manager (SPB Manager) is an entity which creates the Secondary Platform Bundle image according to the directives of a Service Provider. The Secondary Platform Bundle Manager also makes the bound Secondary Platform Bundle image for the target iSSP and delivers it securely over the target iSSP Primary Platform through the LBA.

The Local Bundle Assistant (LBA) is an entity in the terminal that communicates with the Secondary Platform Bundle Manager and transfers the bound Secondary Platform Bundle image to the Secondary Platform Bundle Loader. The Local Bundle Assistant also performs local management of the Secondary Platform Bundle containers via the Secondary Platform Bundle Loader.

The Secondary Platform Bundle Loader is as specified in clause 7.3.1.

A Certificate Issuer (CI) is a Root CA in the iSSP ecosystem which issues certificates to the Secondary Platform Bundle Manager and/or to the SSP maker. There may exist more than one Certificate Issuer in the iSSP ecosystem.

The SSP maker is an entity that provides the Primary Platform and the Secondary Platform Bundle Loader.

The Service Provider is an entity which defines the requirements of a Secondary Platform Bundle and requests the SPB Manager to prepare Secondary Platform Bundle images for target iSSPs.

12.1.4 Interfaces

Table 12.1 provides information about the interfaces within the architecture.

Table 12.1: Interfaces

Interface	Between		Description
Si1	Service Provider	SPB Manager	This interface is used by the Service Provider to request the SPB Manager to prepare Secondary Platform Bundle image for target iSSPs. This interface is also used by the SPB Manager to send notifications to the Service Provider about Secondary Platform Bundle provisioning or management operations.
Si2	SPB Manager	LBA	This interface is used between the SPB Manager and the LBA to provide a transport of the bound Secondary Platform Bundle image and the management commands on the Secondary Platform Bundles installed in the iSSP. This interface is also used by the LBA to send notifications to the SPB Manager about Secondary Platform Bundle provisioning or management operations.
Si3	LBA	Secondary Platform Bundle Loader	This interface is used between the LBA and the SPBL to transfer a bound Secondary Platform Bundle image and management commands to the SPBL.
Si4	SPB Manager	Secondary Platform Bundle Loader	This interface provides authorization, mutual authentication, integrity and confidentiality for loading of the bound Secondary Platform Bundle image. This interface is tunneled through the LBA by using the interfaces Si2 and Si3.
Si5	End user	LBA	This interface is used to initiate local management functions of the LBA by the End User.
Si6	CI	SPB Manager, SSP Maker	This interface is used by the SPB Manager and the SSP maker to request certificate signing (out of scope of the present document).
Si7	SSP Maker	iSSP	This interface is the interface between the SSP maker and the iSSP (out of scope of the present document).

12.2 Security overview

12.2.1 Public key infrastructures

12.2.1.1 Public key infrastructure for Si4 interface

12.2.1.1.1 Certificate chains

12.2.1.1.1.1 Certification path for Secondary Platform Bundle Loader

The Secondary Platform Bundle Loader certification path for digital signature shall include the following certificates:

- CI certificate.
- SSP maker certificate.
- SPBL certificate: The SPBL certificate shall contain the public key used to verify the signature generated by the Secondary Platform Bundle Loader.

The Secondary Platform Bundle Loader certification path may include the following certificates:

- CI subordinate CA certificate: The CI subordinate CA certificate shall be issued by a CI. The CI subordinate CA certificate can be used to verify an SSP maker certificate.
- SSP maker subordinate CA certificate: The SSP maker subordinate CA certificate shall be issued by an SSP maker. The SSP maker subordinate CA certificate can be used to verify an SPBL certificate.

12.2.1.1.1.2 Certification path for SPB Manager

The SPBM certification path for digital signature shall include the following certificates:

- CI certificate.
- SPBM DS certificate: The SPBM DS certificate shall be used to verify the signature generated by the SPB Manager.

The SPBM certification path for key agreement shall include the following certificates:

- CI certificate.
- SPBM KA certificate: The SPBM KA certificate shall be used to generate a session key for secure communication between the SPB Manager and the Secondary Platform Bundle Loader.

The SPBM certification path for digital signature and key agreement may include the following certificates:

- CI subordinate CA certificate: The CI subordinate CA certificate shall be issued by a CI. The CI subordinate CA certificate can be used to verify an SPBM Subordinate CA certificate. The CI subordinate CA certificate can be used to verify an SPBM DS certificate and SPBM KA certificate.
- SPBM subordinate CA certificate: The SPBM subordinate CA certificate shall be issued by an SPBM or CI. The SPBM subordinate CA certificate can be used to verify an SPBM DS certificate and SPBM KA certificate.

12.2.1.1.2 Certificate description

12.2.1.1.2.1 Certificates common fields

Table 12.2 describes the basic certificate fields for all certificates used by the Secondary Platform Bundle Loader and the SPB Manager that follow X.509 v3 certificate format as defined in IETF RFC 5280 [11].

Table 12.2: Certificates common fields

Certificate common fields		
Field	**Value Description**	
tbsCertificate	Data to be signed	
	Field	**Value Description**
	Version	Integer value of 2 denoting version 3.
	serialNumber	Positive integer value assigned by the issuer to identify this certificate.
	Signature	Identifier of signature algorithm used by the issuer to sign this certificate. This value shall be the same as the one of 'signatureAlgorithm' field.
	Issuer	Distinguished Name (DN) of the entity that has signed and issued this certificate. The value is defined as X.501 type name [31].
	Validity	Certificate validity period.
	Subject	Distinguished Name (DN) of the entity associated with the public key in this certificate.
	subjectPublicKeyInfo	Value of the public key and algorithm with which the key is used. subjectPublicKeyInfo.algorithm shall be 'AlgorithmIdentifier' defined in clause 12.2.1.1.3 subjectPublicKeyInfo.subjectPublicKey shall be the value of public key coded as defined in IETF RFC 5480 [29].
signatureAlgorithm	Identifier of signature algorithm used by the issuer to sign this certificate. This value shall be the same as the one of 'signature' field.	
signatureValue	Digital signature computed over ASN1. DER encoded tbsCertificate using digital signature algorithm.	

12.2.1.1.2.2 Extension fields for Certificates

The following extension fields defined in IETF RFC 5280 [11] are considered:

- **Authority key identifier** (IETF RFC 5280 [11], section 4.2.1.1): All the certificate except for CI certificate shall contain the extension for authority key identifier.
- **Subject key identifier** (IETF RFC 5280 [11], section 4.2.1.2): All the certificate shall contain the extension for subject key identifier. The value of this field shall be the identifier of the public key contained in the certificate.
- **Key usage** (IETF RFC 5280 [11], section 4.2.1.3): For a certificate used for verifying its subject certificate, keyCertSign (bit 5) shall be asserted to the key usage extension field of the certificate. For the last certificate in the Secondary Platform Bundle Loader certification path for signature generation, digitalSignature (bit 0) shall be asserted to the key usage extension. For the last certificate in the SPB Manager certification path for key agreement, keyAgreement (bit 4) bit shall be asserted to the key usage extension.
- **Certificate polices** (IETF RFC 5280 [11], section 4.2.1.4): Each certificate shall have the appropriate value of the extension for certificate policies. The OIDs used for value of the extension for certificate polices are defined as follows:

```
-- ASN1START

id-issp OBJECT IDENTIFIER ::= {itu-t (0) identified-organization (4) etsi (0) smart-secure-platform
(3666) part2 (2) }

id-issp-role OBJECT IDENTIFIER ::= {id-issp role (xx)}

id-issp-role-ci OBJECT IDENTIFIER ::= {id-issp-role ci (xx)}
id-issp-role-ci-subordinateca OBJECT IDENTIFIER ::= {id-issp-role-ci subordinate-ca (xx)}

id-issp-role-spbm OBJECT IDENTIFIER ::= {id-issp-role spbm (xx)}

id-issp-role-spbm-subordinateca OBJECT IDENTIFIER ::= {id-issp-role-spbm subordinate-ca (xx)}
id-issp-role-spbm-ds1 OBJECT IDENTIFIER ::= {id-issp-role-spbm ds1 (xx)}
id-issp-role-spbm-ds2 OBJECT IDENTIFIER ::= {id-issp-role-spbm ds2 (xx)}
```

```
id-issp-role-spbm-ka OBJECT IDENTIFIER ::= {id-issp-role-spbm ka (xx)}

id-issp-role-sspm OBJECT IDENTIFIER ::= {id-issp-role sspm (xx)}
id-issp-role-sspm-subordinateca OBJECT IDENTIFIER ::= {id-issp-role-sspm subordinate-ca (xx)}
id-issp-role-spbl OBJECT IDENTIFIER ::= {id-issp-role spbl (xx)}

-- ASN1STOP
```

- **SubjectAltName** (IETF RFC 5280 [11], section 4.2.1.6): A certificate may have the extension for subjectAltName.

- **Basic constraints** (IETF RFC 5280 [11], section 4.2.1.9): For any CA or subordinate CA certificate, the value of the extension for basic constraints shall be asserted.

The following additional extension fields shall be present according to the following rules:

- **Primary Platform identifier:** The Primary Platform identifier extension field shall only be contained in the SPBL certificate. The Primary Platform identifier extension field shall contain the Primary Platform identifier of the SSP.

- **Family identifier:** A certificate shall contain the family identifier extension field if it is present in its parent-certificate. If it is not present in the parent-certificate, a certificate may contain the family identifier extension field. The family identifier extension field shall indicate the list of family identifiers associated with the certification path to load a Secondary Platform Bundle image. If the family identifier extension field is present in the parent certificate, this list may contain the list or a subset of the list of family identifiers indicated in the family identifier extension field of the parent certificate.

- **Custodian identifier:** A certificate shall contain the custodian identifier extension field if it is present in its parent-certificate. If it is not present in the parent-certificate, a certificate may contain the custodian identifier extension field. The custodian identifier extension field shall indicate the list of OIDs of custodians associated with the certification path to load a Secondary Platform Bundle image. If the custodian identifier extension field is present in the parent certificate, this list may contain the list or a subset of the list of custodian identifiers indicated in the custodian identifier extension field of the parent certificate.

Table 12.3: Additional extension field

Extension field	extnID	extnValue	Value Description
Primary Platform identifier	id-issp-ppidentifier	ExtensionPpIdentifier	The Primary Platform identifier of the SSP.
Family identifier	id-issp-familyidentifier	ExtensionFamilyIdentifier	The list of family identifier(s) of which the Secondary Platform Bundle image allowed to be loaded with this certificate.
Custodian identifier	id-issp-custodianidentifier	ExtensionCustodianIdentifier	The list of OIDs of custodians of which the Secondary Platform Bundle image allowed to be loaded with this certificate.

The ASN.1 data structure for family identifier extension and custodian identifier extension are defined as follows:

```
-- ASN1START

id-issp OBJECT IDENTIFIER ::= {itu-t (0) identified-organization (4) etsi (0) smart-secure-platform
(3666) part2 (2) }

id-issp-ppidentifier OBJECT IDENTIFIER ::= {id-issp primary-platform-identifier (xx)}
id-issp-familyidentifier OBJECT IDENTIFIER ::= {id-issp family-identifier (xx)}
id-issp-custodianidentifier OBJECT IDENTIFIER ::= {id-issp custodian-identifier (xx)}

ExtensionPpIdentifier ::= OCTET STRING -- the Primary Platform identifier

ExtensionFamilyIdentifier ::= SEQUENCE
{
    aListOfAllowedFamilyIdentifier SET OF UUID
-- list of family identifier(s) associated with the certification path allowed to load
-- the Secondary Platform Bundle image
}

ExtensionCustodianIdentifier ::= SEQUENCE
```

```
{
    aListOfAllowedCustodianIdentifier SET OF AllowedCustodianIdentifier
-- list of custodian identifier(s) associated with the certification path allowed to
-- load the Secondary Platform Bundle image
}

AllowedCustodianIdentifier ::= SEQUENCE
{
    aCustodianId OBJECT IDENTIFIER, -- the OID of a custodian
    aSpbFamilyId UUID -- the family identifier associated with aCustodianId
}

-- ASN1STOP
```

12.2.1.1.3 Algorithm identifiers and parameters

This clause provides the value to be set in 'AlgorithmIdentifier' structure contained in 'subjectPublicKeyInfo', 'signature', and 'signatureAlgorithm' fields of the certificates for each of cryptographic algorithms used in the present document.

For 'subjectPublicKeyInfo' field, the following settings shall apply:

- 'AlgorithmIdentifier.algorithm' field shall be set to:
  - if the value of 'Extension for KeyUsage' field is set to digitalSignature(0) and/or keyCertSign(5):
    - for Elliptic Curve Digital Signature Algorithm (ECDSA), "iso(1) member-body(2) us(840) ansi-X9-62(10045) keyType(2) ecPublicKey(1)" as defined in IETF RFC 5480 [29];
    - for SM2 digital signature algorithm, "iso(1) standard(0) digital-signature-with-appendix(14888) part3(3) algorithm(0) sm2(14)" as defined in ISO/IEC 14888-3 [21];
  - if the value of 'Extension for KeyUsage' field is set to keyAgreement(4):
    - for Elliptic Curve Diffie-Hellman (ECDH), "iso(1) identified-organization(3) certicom(132) schemes (1) ecdh(12)" as defined in IETF RFC 5480 [29].
- 'AlgorithmIdentifier.parameters' field shall be set to:
  - for BrainpoolP256r1: "iso(1) identified-organization(3) teletrust(36) algorithm(3) signatureAlgorithm(3) ecSign(2) ecStdCurvesAndGeneration(8) ellipticCurve(1) versionOne(1) brainpoolP256r1(7)" as defined in IETF RFC 5639 [19];
  - for BrainpoolP384r1: "iso(1) identified-organization(3) teletrust(36) algorithm(3) signatureAlgorithm(3) ecSign(2) ecStdCurvesAndGeneration(8) ellipticCurve(1) versionOne(1) brainpoolP384r1(11)" as defined in IETF RFC 5639 [19];
  - for NIST P-256: "iso(1) member-body(2) us(840) ansi-X-9-62(10045) curves(3) prime(1) secp256v1(7)" as defined in IETF RFC 5480 [29];
  - for NIST P-384: "iso(1) identified-organization(3) certicom(132) curve(0) secp384r1(34)" as defined in IETF RFC 5480 [29].

For 'signature' and 'signatureAlgorithm' fields, the following settings shall apply:

- 'AlgorithmIdentifier.algorithm' field shall be set to:
  - for ECDSA-with-SHA256: "iso(1) member-body(2) us(840) ansi-X9-62(10045) signatures(4) ecdsa-with-SHA2(3) ecdsa-with-SHA256(2)" as defined in IETF RFC 5758 [30];
  - for ECDSA-with-SHA384: "iso(1) member-body(2) us(840) ansi-X9-62(10045) signatures(4) ecdsa-with-SHA2(3) ecdsa-with-SHA384(3)" as defined in IETF RFC 5758 [30];
  - for SM2 digital signature algorithm, "iso(1) standard(0) digital-signature-with-appendix(14888) part3(3) algorithm(0) sm2(14)" as defined in ISO/IEC 14888-3 [21].

- 'AlgorithmIdentifier.parameters' field shall be set to:
  - for ECDSA-with-SHA256 and ECDSA-with-SHA384: the parameters field shall be omitted as defined in IETF RFC 5754 [37], section 3.2.

12.2.1.1.4 Certification path verification

Both the Secondary Platform Bundle Loader and the SPB Manager shall verify the certificate chain received by each other as described in this clause.

A certificate chain to be verified shall satisfy the following conditions:

- The value of Authority Key Identifier extension field in a subject's certificate shall be the same as the value of Subject Key Identifier extension field in the issuer's certificate which is used to verify the subject's certificate.
- The value of issuer field in a subject's certificate shall be the same as the value of subject field in the issuer's certificate which is used to verify the subject's certificate.
- All certificates in the certificate chain shall use the same digital signature algorithm and parameter set indicated by one of the algorithmIdentifier defined in clause 12.2.1.1.3.
- All certificates in the certificate chain shall not have been expired.
- All certificates in the certificate chain shall not have been revoked. The certificate revocation status shall be checked as defined in clause 12.2.1.1.5.

The Secondary Platform Bundle Loader and the SPB Manager shall manage the trusted Public Key information to verify received certificate chain. The Secondary Platform Bundle Loader and the SPB Manager may have multiple sets of trusted public key information. Each set of trusted public key information shall contain the following:

- Public Key information (SubjectPublicKeyInfo as defined in IETF RFC 5280 [11]).
- Subject Key Identifier of the Public Key.
- Family identifier(s) associated with that Public Key, if any.
- Custodian identifier(s) associated with that Public Key, if any.

The Secondary Platform Bundle Loader and the SPB Manager shall verify the received certificate chain by using one set of trusted Public Key information as the trust anchor information for the certification path validation procedure defined in IETF RFC 5280 [11]. The trust anchor shall be determined during the download procedure as defined in clause 12.3.3.

The Secondary Platform Bundle Loader and the SPB Manager shall:

- Perform the certification path validation defined in IETF RFC 5280 [11] to verify the received certificate chain.
- Verify that the received certificate chain follows one of the certificate chains as defined in clause 12.2.1.1.1.
- Verify that all certificate(s) in the received certificate chain follow the corresponding certificate description(s) as defined in clause 12.2.1.1.2.

In addition, the Secondary Platform Bundle Loader and the SPB Manager shall verify that the certificates in the received certificate chain satisfy the condition as described below:

- If a certificate contains the list of family identifier(s) in the family identifier extension field, the list shall contain the family identifier associated with the trust anchor used for the certification path validation.
- If a certificate contains the list of custodian identifier(s) in the custodian identifier extension field, the list shall contain the custodian identifier associated with the trust anchor used for the certification path validation.

If any of the verifications described above fails, the certificate chain shall be considered as invalid.

12.2.1.1.5 Certificate revocation status verification

The Secondary Platform Bundle Loader and the SPB Manager may verify the revocation status of certificates in the received certification path by using a Certificate Revocation List (CRL) as defined in IETF RFC 5280 [11].

The SPB Manager may obtain a CRL by accessing a public repository with the uniform resource identifier indicated in cRLDistributionPoint extension of a certificate as defined in IETF RFC 5280 [11].

The Secondary Platform Bundle Loader may obtain a CRL with the following mechanism:

- The SPB Manager may provide the list of the latest CRL for each certificate in its certification path along with the certification path. The Secondary Platform Bundle Loader shall verify the revocation status of the certificates with the corresponding CRLs provided by the SPB Manager.

12.2.2 Cryptographic algorithms

12.2.2.1 Elliptic curve domain parameter sets

The Secondary Platform Bundle Loader and the SPB Manager shall support at least one out of the following elliptic curve domain parameter sets:

- NIST P-256 as defined in Digital Signature Standard [18].
- NIST P-384 as defined in Digital Signature Standard [18].
- brainpoolP256r1 as defined in IETF RFC 5639 [19].
- brainpoolP384r1 as defined in IETF RFC 5639 [19].

12.2.2.2 Digital signature algorithm

The Secondary Platform Bundle Loader and the SPB Manager shall support at least one of the following algorithms to generate and verify signatures:

- Elliptic Curve Digital Signature Algorithm (ECDSA) as defined in ANSI X9.142-2020 [20].
- SM2 digital signature algorithm as defined in ISO/IEC 14888-3 [21]. The hash function shall be the SM3 hash function as defined in ISO/IEC 10118-3 [22].

12.2.2.3 Key agreement algorithm

The Secondary Platform Bundle Loader and the SPB Manager shall support at least one of the following key agreement algorithms to establish session keys:

- Elliptic Curve Key Agreement Algorithm (ECKA) as defined in BSI TR-03111 [23].
- SM2 Key exchange Algorithm (SM2KA) as defined in SM2 Digital Signature Algorithm [24], section 6.2.

If ECKA is used as the key agreement algorithm, a shared secret shall be generated by using one between either the ElGamal key agreement protocol or the Diffie-Hellman key agreement protocol. The session key shall be computed from the generated shared secret value using the X9.63 key derivation function as described in GP Open Firmware Loader for Tamper Resistant Element [6], clause 3.2.1.

12.2.2.4 Block cipher algorithm

For data encryption algorithm, the Secondary Platform Bundle Loader and the SPB Manager shall support the following block cipher algorithms to encrypt data:

- eGCM based on AES-128 as defined in GP Open Firmware Loader for Tamper Resistant Element [6], Annex A.

- eGCM based on AES-256 as defined in GP Open Firmware Loader for Tamper Resistant Element [6], Annex A.

The Secondary Platform Bundle Loader and/or the SPB Manager may additionally support the following block cipher algorithm:

- eGCM algorithm as defined in GP Open Firmware Loader for Tamper Resistant Element [6], Annex A using the SM4 block cipher as defined in [44].

12.3 Secondary Platform Bundle provisioning procedure

12.3.1 Overview

This clause specifies the procedures for provisioning a Secondary Platform Bundle from an SPB Manager to an iSSP. The overall procedure consists of four sub procedures as follows:

- preparation procedure;
- download procedure;
- installation procedure;
- notification procedure.

The preparation procedure shall be performed between a Service Provider and an SPB Manager over the Si1 interface and a Service Provider and a Subscriber. The Subscriber is able to obtain from the service provider the relevant information for loading a Secondary Platform Bundle, including the URL of the SPB Manager to which the iSSP shall establish a secure communication channel to start the download procedure.

The download procedure shall be performed between a Secondary Platform Bundle Loader and an SPB Manager over the Si2 and Si3 interfaces. The bound Secondary Platform Bundle image shall be delivered securely from the SPB Manager to the LBA through the download procedure.

The installation procedure shall be performed between an LBA and a Secondary Platform Bundle Loader over the Si3 interface. The bound Secondary Platform Bundle image shall be delivered from the LBA to the Secondary Platform Bundle Loader through the installation procedure. The Secondary Platform Bundle Loader installs the Secondary Platform Bundle container by decrypting the bound Secondary Platform Bundle image.

The notification procedure shall be performed between the Secondary Platform Bundle Loader and the SPB Manager over the Si1, Si2 and Si3 interfaces.

Figures 12.2 and 12.3 present two cases:

- Case 1: the preparation procedure is completed before the download procedure is triggered.
- Case 2: the preparation procedure is not completed before the download procedure is triggered, i.e. the service provider needs additional information from the terminal and/or the SSP to select the Secondary Platform Manager.

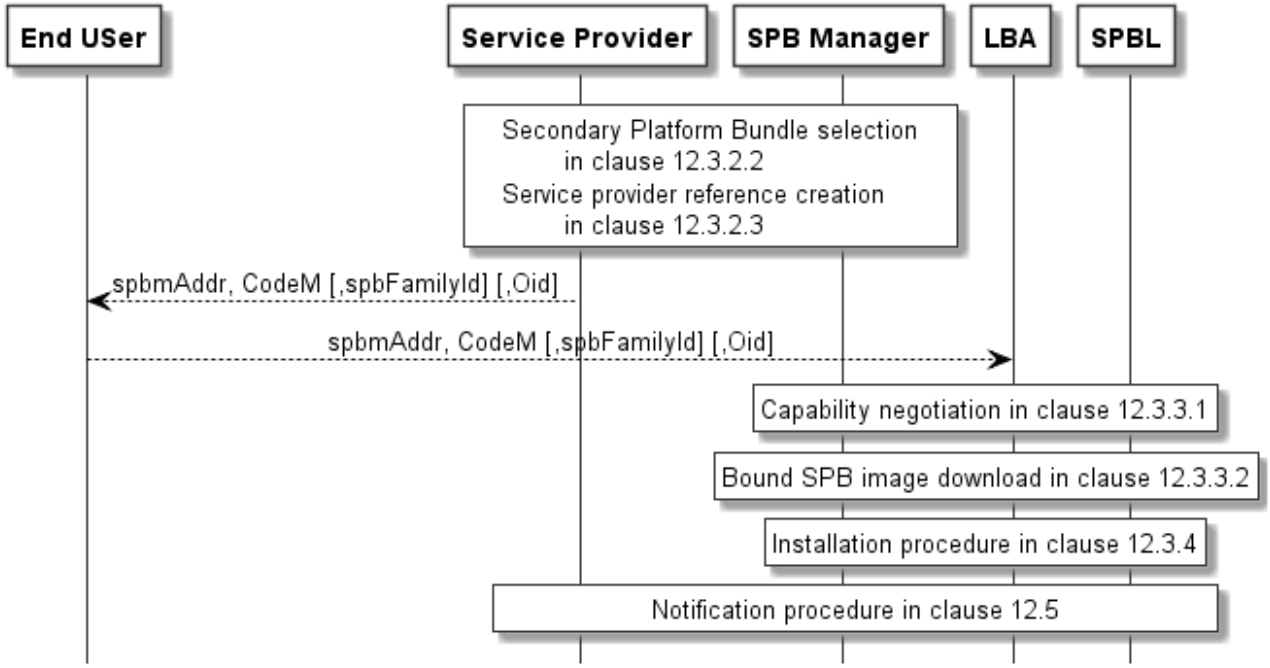

Figure 12.2: Secondary Platform Bundle provisioning procedure - case 1

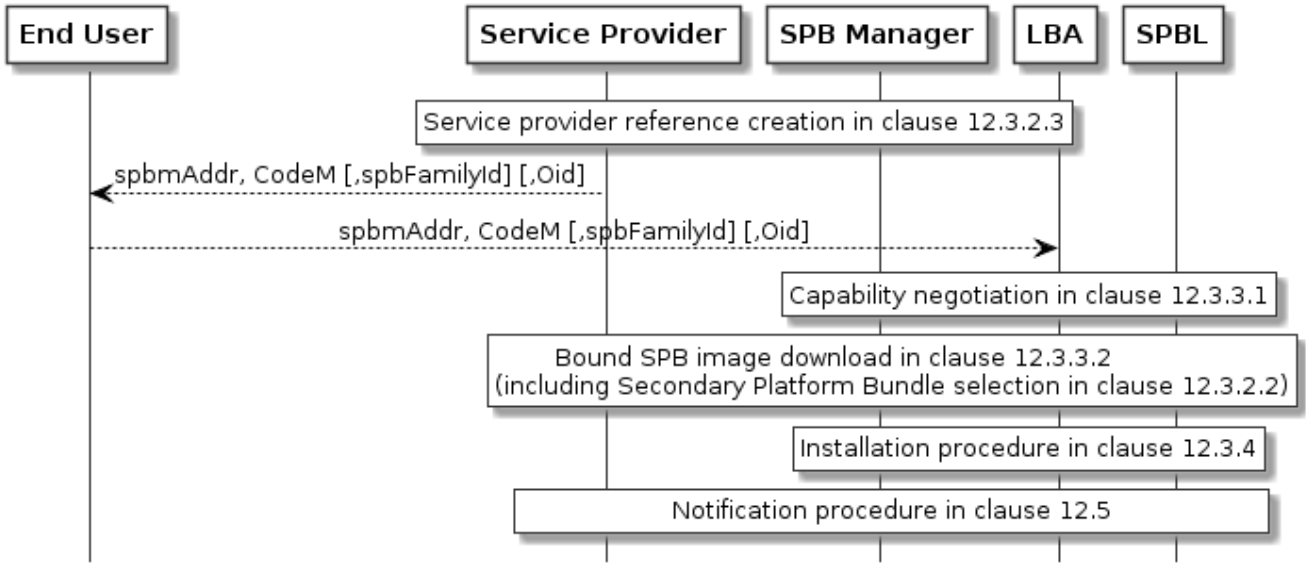

Figure 12.3: Secondary Platform Bundle provisioning procedure - case 2

12.3.2 Preparation procedure

12.3.2.1 Overview

This clause describes the procedures that take place between the service provider and the SPB Manager through the Si1 interface for preparing the download of a Secondary Platform Bundle.

The preparation procedure consists of the following two processes:

- Secondary Platform Bundle selection process: the selection of the Secondary Platform Bundle allowing the service provider to select a Secondary Platform Bundle that matches the terminal and the SSP capabilities.
- Service provider reference creation process: the creation of a reference shared between the service provider and the SPB Manager. This allows the end user to trigger the download procedure as defined in clause 12.3.3.

The Secondary Platform Bundle selection process and the service provider reference creation process may be executed in any order or in once, according to the service provider's implementation choices.

12.3.2.2 Secondary Platform Bundle selection process

The selection of the Secondary Platform Bundle allows the service provider to select a Secondary Platform Bundle that matches the terminal and the SSP capabilities. This is performed using the "Si1.SelectSpb" function.

This process may be executed regardless of the service provider reference creation process defined in clause 12.3.2.3.

If a CodeM is provided as input parameter of the "Si1.SelectSpb" function and is known by the SPB Manager and not already linked to another Secondary Platform Bundle, the service provider reference creation process is not needed.

If the Secondary Platform Bundle selection process is performed after the service provider reference creation process, the service provider may indicate to the SPB Manager that its internal procedures, e.g. the provisioning of its technical platforms or data bases, are completed using the "Si1.FinalizationPreparation" function.

If the service provider has set aFlagFinalize to TRUE in the "Si1.SelectSpb" function command, the SPB Manager shall wait for the completion of the Secondary Platform Bundle selection process (i.e. after it has sent the response to the "Si1.FinalizePreparation" function related to this Secondary Platform Bundle) to continue with the Bound SPB image download as defined in clause 12.3.3.2.

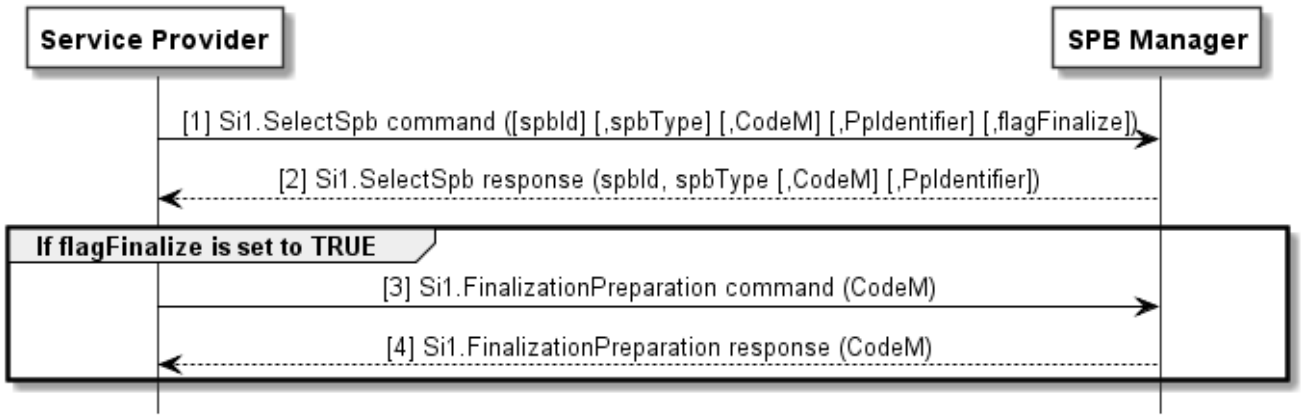

Figure 12.4: Secondary Platform Bundle selection process

12.3.2.3 Service provider reference creation process

This clause describes the process to create a reference shared between the service provider and the SPB Manager. This reference is the CodeM which is used by the end user to trigger the download procedure as defined in clause 12.3.3.

The service provider reference creation process shall be performed using the "Si1.CreateSPReference" function and may be executed regardless of the Secondary Platform Bundle selection process defined in clause 12.3.2.2.

The service provider may pass the CodeM value to be used as a parameter of the "Si1.CreateSPReference" function command. If this is not present, the SPB Manager shall generate the CodeM.

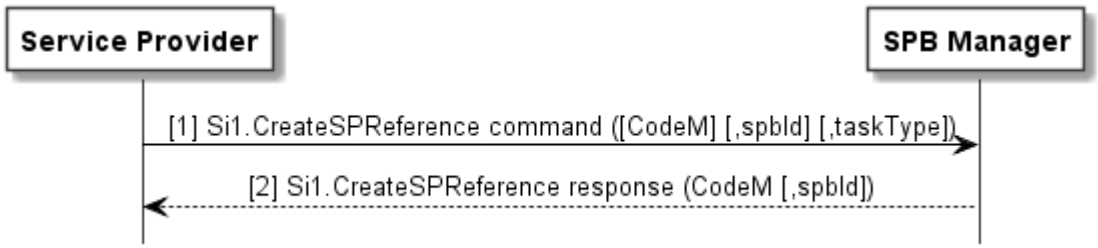

Figure 12.5: Service provider reference creation process

12.3.2.4 Cancellation of the preparation procedure

This clause describes the procedure to cancel the preparation procedure.

This procedure allows the service provider to cancel a pending preparation procedure.

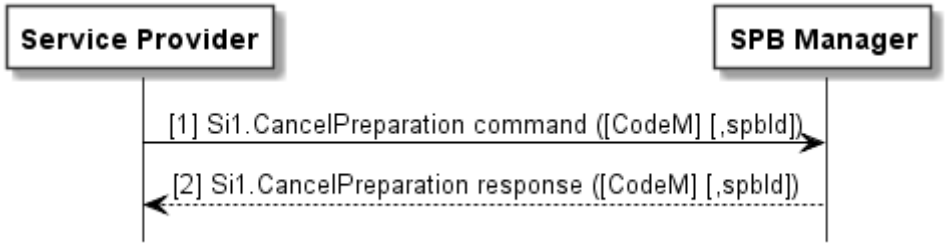

Figure 12.6: Cancellation of the preparation procedure

12.3.3 Download procedure

12.3.3.1 Capability negotiation

This clause describes the capability negotiation procedure between a Secondary Platform Bundle Loader and an SPB Manager to ensure the successful authentication and to determine appropriate cryptographic parameters as per the Secondary Platform Bundle family identifier.

The following shall be determined by the capability negotiation procedure:

- SPBM certificate for key agreement and its certification path.
- Public Key provisioned on the SPB Manager to verify the SPBL certificate and its certification path.
- Data encryption algorithm used by the SPB Manager and the Secondary Platform Bundle Loader.

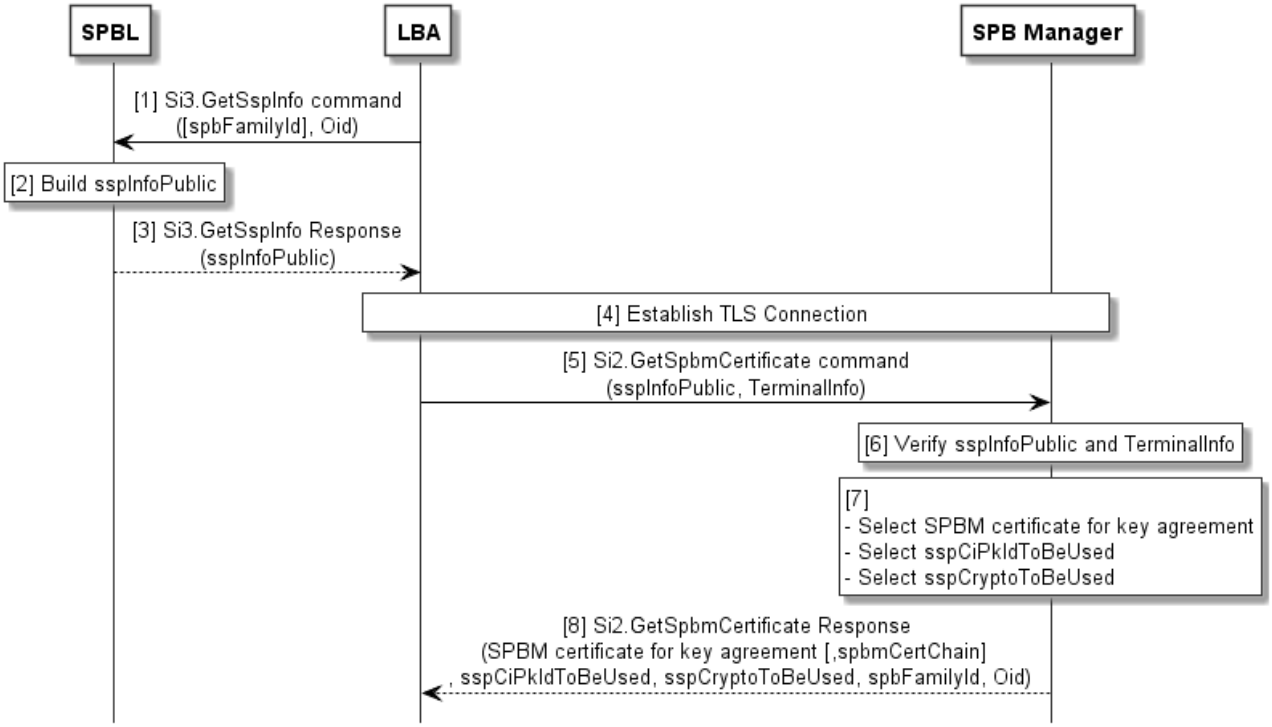

Figure 12.7: Capability negotiation procedure

The LBA shall obtain the address of the SPB Manager (spbmAddr) e.g. from the end user.

The LBA may obtain the family identifier of the Secondary Platform Bundle container (spbFamilyId) to load. If the family identifier is present, the LBA may also obtain the OID of a custodian of that family identifier.

The Secondary Platform Bundle Loader shall have the following:

- Private key(s) for creating the Secondary Platform Bundle Loader signature.
- Secondary Platform Bundle Loader certificate(s) for the digital signature used to verify the Secondary Platform Bundle Loader signature.
- Secondary Platform Bundle Loader certificate chain(s) to be used by an SPB Manager for verifying Secondary Platform Bundle Loader certificate for digital signature.
- Trusted public key(s) and algorithmIdentifier value(s) to be used to verify certificate(s) from an SPB Manager as per the family identifier(s) and/or custodian(s).
- List of supported algorithmIdentifier value(s) for key agreement and data encryption.

The SPB Manager shall have the following:

- Private key(s) for creating the SPB Manager signature.
- Private key(s) for key agreement.
- SPB Manager certificate(s) for digital signature used to verify the SPB Manager signature.
- SPB Manager certificate(s) for key agreement.
- SPB Manager certificate chain(s) to be used by a Secondary Platform Bundle Loader for verifying the SPB Manager certificates for key agreement and for digital signature.
- Trusted public key(s) and algorithmIdentifier value(s) to be used to verify the certificate(s) from the Secondary Platform Bundle Loader as per the family identifier(s) and/or custodian(s).

The capability negotiation procedure shall use the following steps:

1) The LBA shall call the "Si3.GetSspInfo" function. The function command may contain the spbFamilyId. If spbFamilyId is present, the function command may also contain the OID of the custodian for this spbFamilyId.

2) On reception of the "Si3.GetSspInfo" function command, the Secondary Platform Bundle Loader shall build aSspInfoPublic as defined in clause 12.6.2.2.2:

   a) aSpblSpecVerInfo;

   b) aSspGeneralCryptoInfo; and/or

   c) aSspFamilyCryptoInfoBlock, that may contain multiple SspFamilyCryptoInfoBlock data structures. Each aSspFamilyCryptoInfoBlock data structure shall contain a family identifier and may contain the aSspFamilyCryptoInfo and/or the set of SspOidCryptoInfoBlock data structures. Each SspOidCryptoInfoBlock data structure shall contain the aCustodianOid and aSspOidCryptoInfo. The aSspGeneralCryptoInfo, the aSspFamilyCryptoInfo and the aSspOidCryptoInfo shall comprise the following:

      - aSspPkIdForSpbmVerification: trusted public key identifier(s) available for the Secondary Platform Bundle Loader to verify SPB Manager certificate chain.
      - aSspPkIdForSpblVerification: trusted public key identifier(s). The SPB Manager shall use one of these trusted public key identifiers to verify Secondary Platform Bundle Loader certificate chain.
      - aKeyAgreementAlgorithmList: the list of algorithm identifiers for key agreement algorithms supported by the Secondary Platform Bundle Loader.
      - aCipherAlgorithmList: the list of algorithm identifiers of data encryption algorithms supported by the Secondary Platform Bundle Loader.

If the Secondary Platform Bundle Loader cannot build aSspInfoPublic, the Secondary Platform Bundle Loader shall return an error to the LBA as defined in clause 12.6.5.3. The LBA and the Secondary Platform Bundle Loader shall then terminate the procedure.

3) The Secondary Platform Bundle Loader shall return the aSspInfoPublic to the LBA.

4) The LBA shall establish a TLS connection with the SPB Manager in server authentication mode.

NOTE 1: The establishment and management of the TLS connection are outside the scope of the present document.

5) The LBA shall call the "Si2.GetSpbmCertificate" function with its input data comprising aSspInfoPublic and aTerminalInfo.

6) On reception of "Si2.GetSpbmCertificate" function command, the SPB Manager shall verify that it supports the contents in aSspInfoPublic and aTerminalInfo.

7) Using aSspInfoPublic, the SPB Manager shall choose:

   a) An SPB Manager certificate for key agreement that can be verified by the trusted public key indicated by one of the trusted public key identifiers in the aSspPkIdListForSpbmVerification. The algorithmIdentifier of the selected certificate shall be one of the algorithmIdentifier in aKeyAgreementAlgorithmList.

   b) An SPB Manager certificate for digital signature that can be verified by the trusted public key indicated by one of the trusted public key identifiers in the aSspPkIdListForSpbmVerification.

   c) One of trusted public key identifier(s) in the aSspPkIdListForSpblVerification that shall be used by the Secondary Platform Bundle Loader to select its certificate(s). The SPB Manager shall set the selected trusted public key identifier into aSspCiPkIdToBeUsed.

   d) One of algorithmIdentifiers in the aCipherAlgorithmList that shall be used by the Secondary Platform Bundle Loader and the SPB Manager for data encryption. The SPB Manager shall set the selected algorithmIdentifier into aSspCryptoToBeUsed.

   The SPB Manager shall generate a random challenge (ChallengeS) which is used in the authentication of the Secondary Platform Bundle Loader.

   If the SPB Manager cannot find the appropriate certificate(s), trusted public key identifier, or algorithmIdentifier value, the SPB Manager shall return an error to the LBA and shall terminate the procedure.

NOTE 2: The handling of aTerminalInfo is outside the scope of the present document and may be defined by other organizations.

8) The SPB Manager shall return the SPB Manager certificate for key agreement, aSspCiPkIdToBeUsed, aSspCryptoToBeUsed and aSpbFamilyId, and optionally the certificate chain for SPB Manager certificate for key agreement and the OID of a custodian of the aSpbFamilyId to the LBA.

12.3.3.2 Bound SPB image download

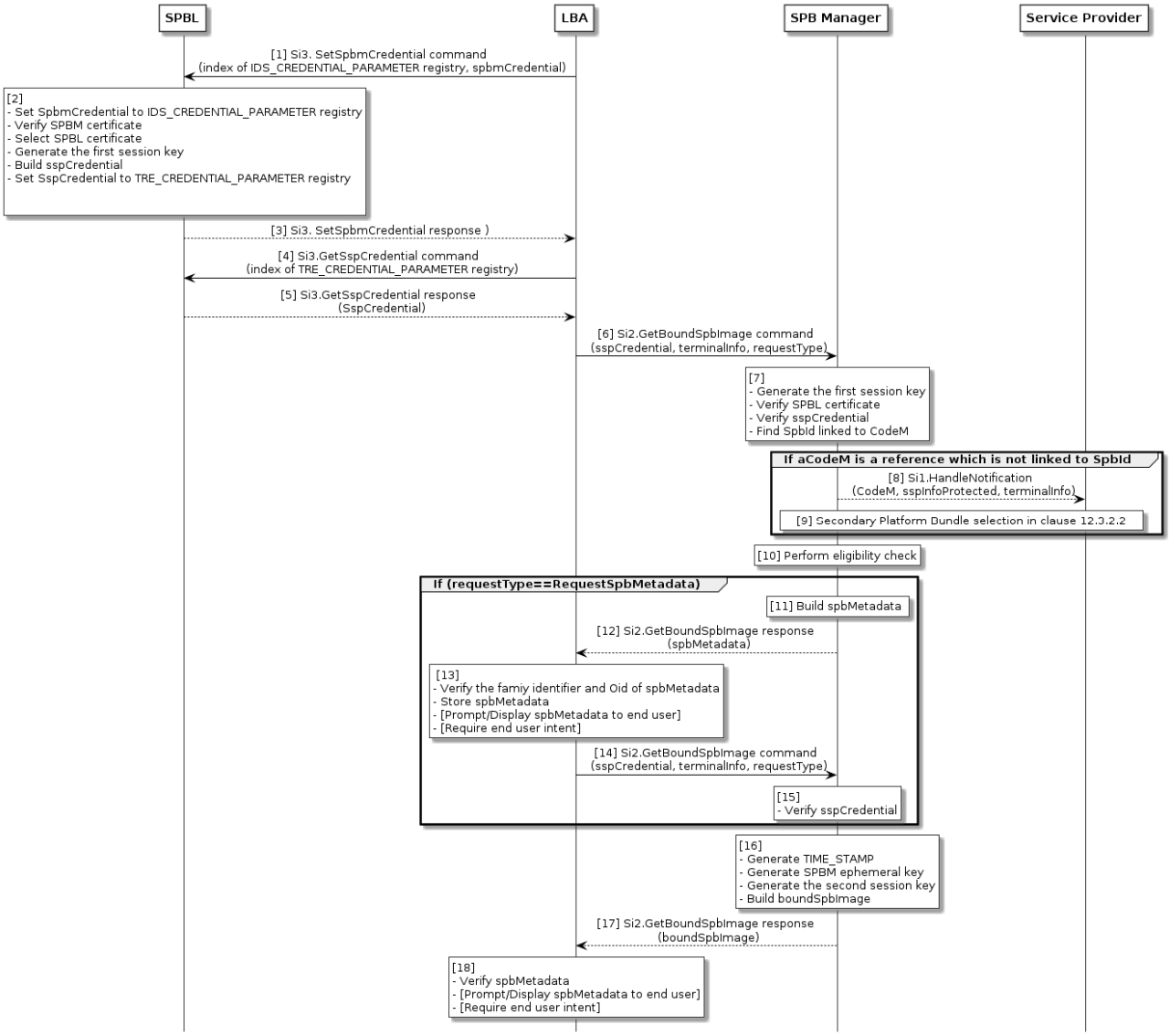

Figure 12.8: Bound SPB image download

The bound SPB image download procedure shall use the following steps:

1) The LBA shall call the "Si3.SetSpbmCredential" function. The function command shall contain the index of IDS_CREDENTIAL_PARAMETER registry and the aSpbmCredential which is defined in clause 12.6.2.3.

2) On reception of the "Si3.SetSpbmCredential" function command, the Secondary Platform Bundle Loader shall:

   a) Set the aSpbmCredential to IDS_CREDENTIAL_PARAMETER registry.

   b) Verify that the received SPB Manager's certificate for key agreement contained in the aSpbmCredential by using the certification path verification as defined in clause 12.2.1.1.4.

   c) Select the appropriate Secondary Platform Bundle Loader certificate that shall be verifiable by the trusted public key which is identified by the aSspPkIdForSpblVerification contained in the aSpbmCredential.

   d) Generate an ephemeral key pair and generate the first session key.

   e) Build the aSspCredential which is defined in clause 12.6.2.4.

f) Set the aSspCredential to the TRE_CREDENTIAL_PARAMETER registry.

3) The Secondary Platform Bundle Loader shall return ANY_OK to the LBA.

4) The LBA shall call "Si3.GetSspCredential" function. The function command shall contain the index of TRE_CREDENTIAL_PARAMETER registry.

5) The Secondary Platform Bundle Loader shall return ANY_OK with the aSspCredential.

6) The LBA shall call the "Si2.GetBoundSpbImage" function. The function command shall contain aSspCredential, aTerminalInfo, and aRequestType.

7) On reception of the "Si2.GetboundSpbImage" function command, the SPB Manager shall:

a) Generate the first session key.

b) Decrypt the encrypted data contained in the aSspCredential by using the first session key.

c) Verify the Secondary Platform Bundle Loader certificate by using the certification path verification as defined in clause 12.2.1.1.4 with the public key identified by the aSspPkIdForSpblVerification. The aSspPkIdForSpblVerification shall be determined in the capability negotiation as defined in clause 12.3.3.1.

d) Find the Secondary Platform Bundle identifier linked to the aCodeM contained in the aSspCredential.

If the aCodeM is a reference which is not linked to a Secondary Platform Bundle identifier, the steps 8 and 9 shall be performed. Otherwise the step 10 shall be performed after finishing the step 7:

8) The SPB Manager shall call the "Si1.HandleNotification" function. The function command shall contain the aSspInfoProtected, aTerminalInfo and aCodeM.

9) The service provider shall perform the Secondary Platform Bundle selection process as defined in clause 12.3.2.2.

10) The SPB Manager shall perform the eligibility check based on the aSspInfoProtected and the aTerminalInfo.

If the aRequestType is "RequestSpbMetadata", the steps from 11 to 15 shall be performed. Otherwise, the step 16 shall be performed after finishing the step 10:

11) The SPB Manager shall build aSpbMetadata and link the aSpbMetadata to the aIdTransac contained in aSspCredential.

12) The SPB Manager shall return the aSpbMetadata to the LBA.

13) The LBA:

a) Shall store the aSpbMetadata.

b) May display the aSpbMetadata to the end user and require the end user intent if configured.

14) The LBA shall call the "Si2.GetBoundSpbImage" function. The function command shall contain aSspCredential, aTerminalInfo and aBoundSpbImageByTransacId as aRequestType.

15) Upon reception of "Si2.GetBoundSpbImage" function command and if the received function command contains the aBoundImageByTransacId, the SPB Manager shall verify the step 7 of this procedure was performed with the same aSspCredential.

16) The SPB Manager shall:

a) Generate TIME_STAMP and ephemeral key pair.

b) Generate the second session key.

c) Build an aBoundSpbImage as defined in clause 12.6.2.5.

17) The SPB Manager shall return the aBoundSpbImage to the LBA.

18) On reception of the "Si2.GetBoundSpbImage" response, the LBA shall verify that the aSpbMetdatanreceived in step 13 and the aSpbMetadata contained in the aBoundSpbImage are the same. If the LBA did not request aSpbMetadata previously, the LBA shall display the aSpbMetadata to the end user and request the end user intent if configured.

12.3.4 Installation procedure

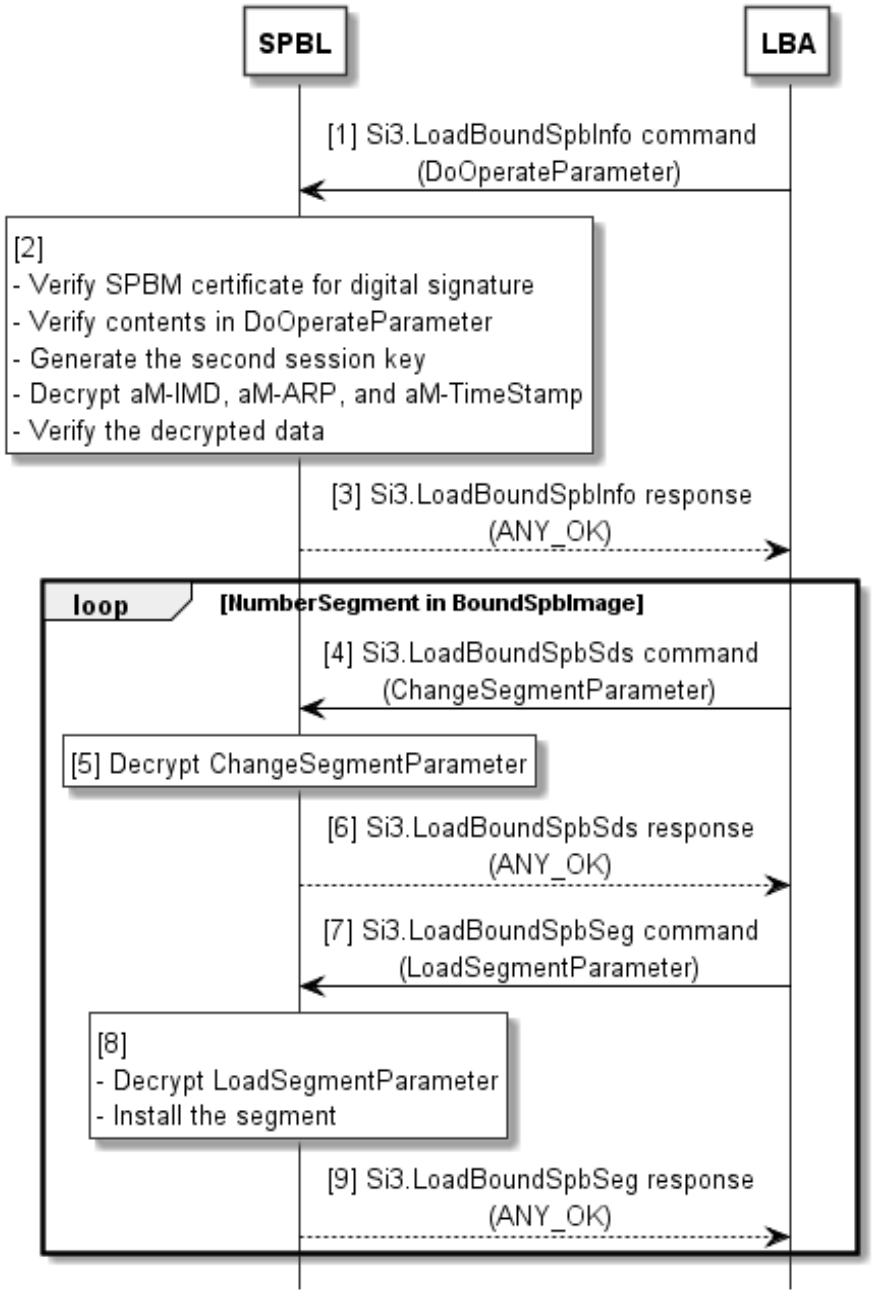

Figure 12.9: Installation procedure

The installation procedure shall use the following steps:

1) The LBA shall call the "Si3.LoadBoundSpbInfo" function. The function command shall contain aDoOperateParameter contained in the aBoundSpbImage received from the SPB Manager.

2) On reception of the "Si3.LoadBoundSpbInfo" function command, the Secondary Platform Bundle Loader shall:

   a) Verify SPBM certificate for digital signature.

b) Verify the content in the aDoOperateParameter.

c) Generate the second session key.

d) Decrypt the aM-TimeStamp contained in the aDoOperateParameter by using the first session key and decrypt the aM-IMD and aM-ARP contained in the aDoOperateParameter by using the second session key.

NOTE: The first session key and the second session key are generated during the download procedure as defined in clause 12.3.3.

e) Verify the content in the decrypted data.

3) The Secondary Platform Bundle Loader shall return ANY_OK to the LBA.

The LBA shall perform the following steps for aNumberSegment times. The aNumberSegment shall be in the aBoundSpbImage:

4) The LBA shall call the "Si3.LoadBoundSpbSds" function. The function command shall contain the aChangeSegmentParameter contained in the aBoundSpbImage.

5) On reception of the "Si3.LoadBoundSpbSds" function command, the Secondary Platform Bundle Loader shall decrypt the aChangeSegmentParameter by using the second session key.

6) If successful, the Secondary Platform Bundle Loader shall return ANY_OK to the LBA.

7) The LBA shall call the "Si3.LoadBoundSpbSeg" function. The function command shall contain the aLoadSegmentParameter contained in the aBoundSpbImage.

8) On reception of the "Si3.LoadBoundSpbSeg" function command, the Secondary Platform Bundle Loader shall decrypt the aLoadSegmentParameter by using the key obtained by decrypting the aChangeSegmentParameter.

9) If successful, the Secondary Platform Bundle Loader shall return ANY_OK to the LBA.

12.3.5 SSP activation code

The SSP activation code contains the information which is needed by the LBA to trigger the Secondary Platform Bundle provisioning procedure.

The SSP activation code is encoded using a URI, as defined in IETF RFC 3986 [42], using the following rules:

- The scheme shall have the value "lba".
- The authority shall have the value of the FQDN of the SPB Manager to which the terminal shall establish a connection to download a Secondary Platform Bundle.
- The path shall contain the action to be performed. The string "bundle" identifies the action to download a Secondary Platform Bundle.
- The query contains the other parameters, in the form of key/value pairs. The following keys are defined:

Key	Value	M/O/C
codem	It describes the CodeMatching identifier used to indicate the specific Secondary Platform Bundle which is linked to the CodeM during the bundle ordering process.	M
familyid	It describes the family identifier of the Secondary Platform Bundle.	O (see note)
oid	It indicates the OID of the custodian of the family identifier.	O
NOTE: If oid is provided, familyid shall be present.		

The LBA shall reject SSP activation codes containing an unknown path value.

Examples of SSP activation code are as follows:

- lba://spbm.carrier.com/bundle?codem=1111-2222-3333-4444
- lba://spbm.carrier.com/bundle?codem=1111-2222-3333-4444&familyId=69c51bb6-7470-5661-a8d7-169f8d869baf

NOTE: A custodian may define its own SSP activation code format to support its own use cases and requirements (e.g. SSP activation code for the Telecom Bundle).

12.4 Secondary Platform Bundle management procedure

12.4.1 Enable a Secondary Platform Bundle

This clause describes the procedure to enable a Secondary Platform Bundle installed on an iSSP.

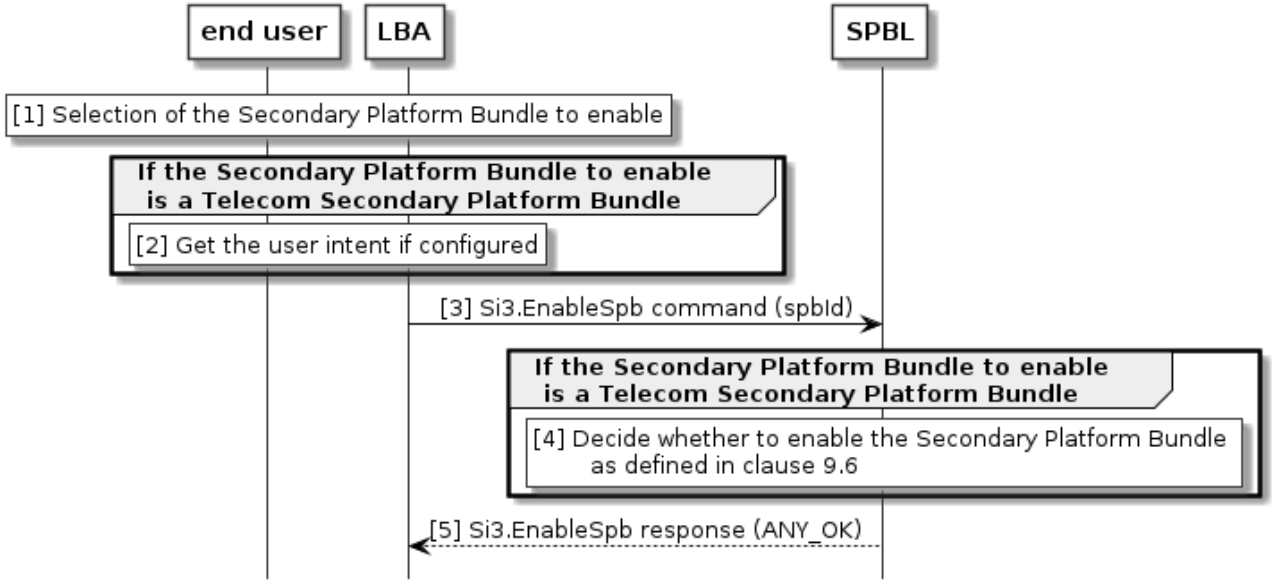

Figure 12.10: Enable Secondary Platform Bundle procedure

The procedure to enable a Secondary Platform Bundle installed on the iSSP shall use the following steps:

1) The end user selects the Secondary Platform Bundle to enable through the LBA (out of scope of the present document).

2) The LBA shall get the user intent if the Secondary Platform Bundle to enable is a Telecom Secondary Platform Bundle and if the user intent is configured in the Secondary Platform Bundle metadata.

3) The LBA shall send the Si3.EnableSpb command to the Secondary Platform Bundle Loader with the identifier of the Secondary Platform Bundle to enable.

4) If the Secondary Platform Bundle to enable is a Telecom Secondary Platform Bundle, the Secondary Platform Bundle Loader shall verify if it can be enabled as described in clause 12.6.5.5.7.

5) The Secondary Platform Bundle Loader shall use the Si3.EnableSpb response to indicate the execution status of the command.

12.4.2 Disable a Secondary Platform Bundle

This clause describes the procedure to disable a Secondary Platform Bundle installed on an iSSP.

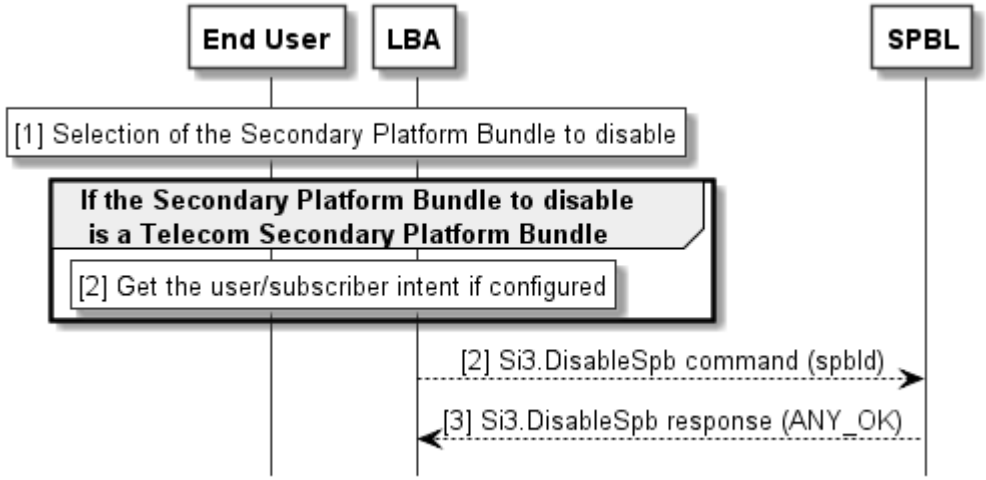

Figure 12.11: Disable Secondary Platform Bundle procedure

The procedure to disable a Secondary Platform Bundle installed on the iSSP shall use the following steps:

1) The end user selects the Secondary Platform Bundle to disable through the LBA (out of scope of the present document).

2) The LBA shall get the user intent if the Secondary Platform Bundle to disable is a Telecom Secondary Platform Bundle and if the user intent is configured in the Secondary Platform Bundle metadata.

   The LBA shall send the Si3.DisableSpb command to the Secondary Platform Bundle Loader with the identifier of the Secondary Platform Bundle to disable.

3) The Secondary Platform Bundle Loader shall use the Si3.DisableSpb response to indicate the execution status of the command.

12.4.3 Delete a Secondary Platform Bundle

This clause describes the procedure to delete a Secondary Platform Bundle installed on an iSSP.

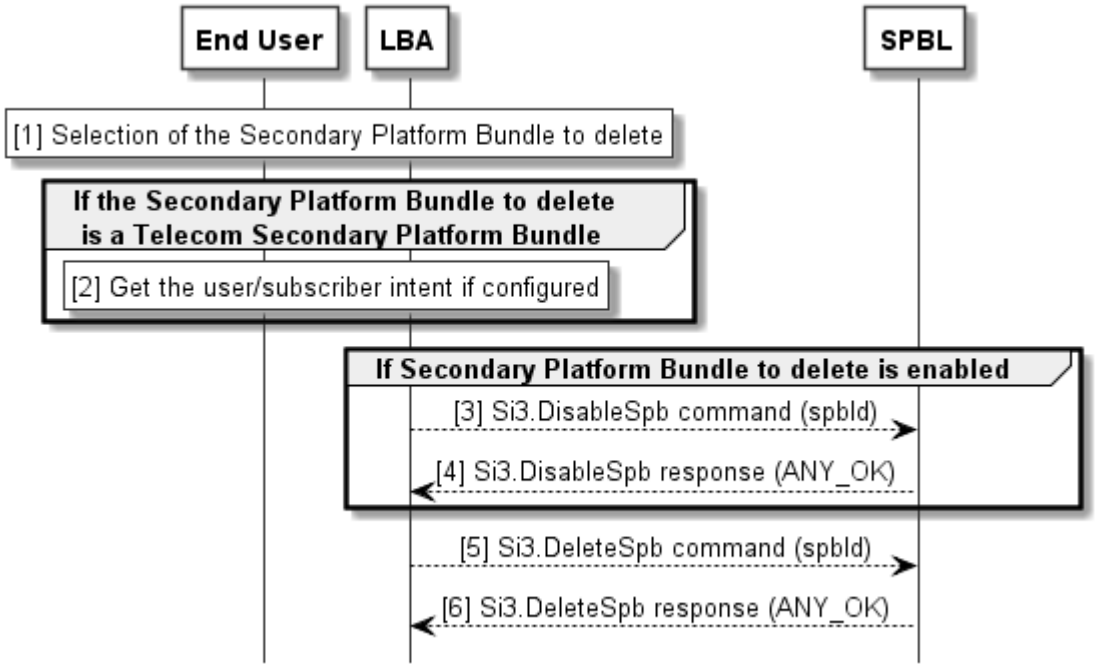

Figure 12.12: Delete Secondary Platform Bundle procedure

The procedure to delete a Secondary Platform Bundle installed on the iSSP shall use the following steps:

1) The end user selects the Secondary Platform Bundle to delete through the LBA (out of scope of the present document).

2) The LBA shall get the user intent if the Secondary Platform Bundle to delete is a Telecom Secondary Platform Bundle and if the user intent is configured in the Secondary Platform Bundle metadata.

If the Secondary Platform Bundle to delete is currently disabled, steps 3 and 4 should be skipped:

3) The LBA shall disable the Secondary Platform Bundle by sending the Si3.DisableSpb command to the Secondary Platform Bundle Loader with the identifier of the Secondary Platform Bundle to disable.

4) The Secondary Platform Bundle Loader shall use the Si3.DisableSpb response to indicate the execution status of the command.

5) The LBA shall send the Si3.DeleteSpb command to the Secondary Platform Bundle Loader with the identifier of the Secondary Platform Bundle to delete.

6) The Secondary Platform Bundle Loader shall use the Si3.DeleteSpb response to indicate the execution status of the command.

12.4.4 SPB metadata retrieving procedure

This clause describes the procedure for the LBA to retrieve the SPB metadata stored in the Secondary Platform Bundle Loader.

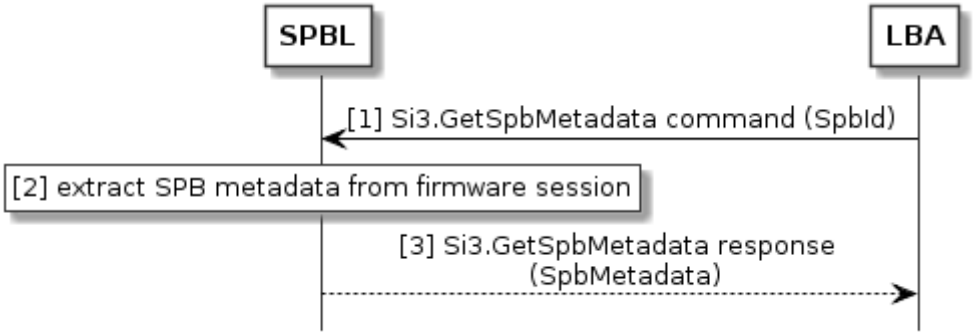

Figure 12.13: SPB metadata retrieving procedure

The SPB metadata retrieving procedure shall use the following steps:

1) The LBA shall call the "Si3.GetSpbMetadata" function. The function command shall contain the identifier of the Secondary Platform Bundle (aSpbId) corresponding to the SPB metadata that the LBA intends to retrieve.

2) The Secondary Platform Bundle Loader shall extract the SPB metadata from the firmware session of the Secondary Platform Bundle container identified by the aSpbId contained in the "Si3.GetSpbMetadata" function command.

3) The Secondary Platform Bundle Loader shall return ANY_OK with the SPB as the "Si3.GetSpbMetadata" function response.

12.4.5 SPB state retrieving procedure

This clause describes the procedure for the LBA to retrieve the state of the Secondary Platform Bundle installed in the iSSP.

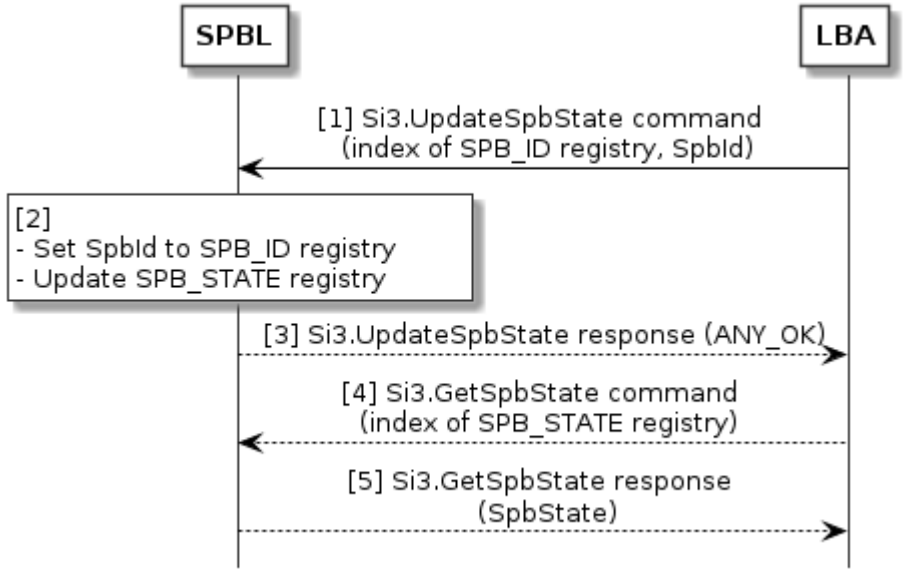

Figure 12.14: SPB state retrieving procedure

The SPB state retrieving procedure shall use the following steps:

1) The LBA shall call the "Si3.UpdateSpbState" function. The function command shall contain the index of SPB_ID registry and the Secondary Platform Bundle identifier (SpbId) corresponding to the SPB of which the LBA intends to retrieve the state.

2) On reception of the "Si3.UpdateSpbState" function command, the Secondary Platform Bundle Loader shall:

   a) Set the SpbId to SPB_ID registry.

   b) Update the value of SPB_STATE registry with the current state of the Secondary Platform Bundle identified by the SpbId.

3) The Secondary Platform Bundle Loader shall return ANY_OK to the LBA.

4) The LBA shall call "Si3.GetSpbState" function. The function command shall contain the index of SPB_STATE registry.

5) The Secondary Platform Bundle Loader shall return ANY_OK with the value of SPB_STATE registry to the LBA.

12.5 Notification procedure

12.5.1 Overview

This clause describes the procedure to provide a remote entity with a report about the progress or the execution status of a Secondary Platform Bundle provisioning or a Secondary Platform Bundle management operation.

12.5.2 Notification of the service provider

This clause describes the procedure to send a notification to the service provider.

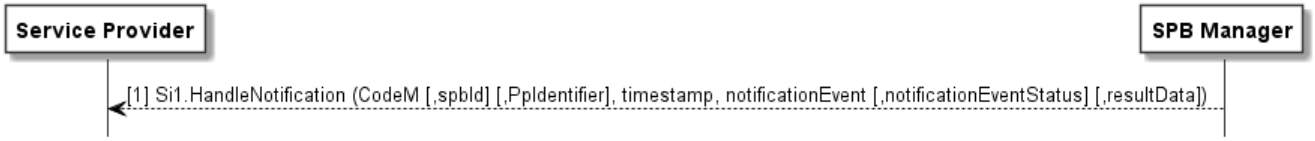

Figure 12.15: Handle notification from SPB Manager

If any of the following steps has to be notified to the service provider according to the configuration of the SPB metadata, the SPB Manager shall call the "Si1.HandleNotification" function after the execution of this step:

- The eligibility check procedure, as defined in Annex C, has been executed.
- The user rejected the download of a Secondary Platform Bundle.
- The download of a bound Secondary Platform Bundle image.
- The maximum retry attempts to download a Secondary Platform Bundle image has been reached.
- The installation of a Secondary Platform Bundle.
- The enablement, disablement or deletion of a Secondary Platform Bundle.

If, for this step, a notification containing a notification token was previously received from the LBA, as defined in clause 12.5.3, the notification event contained in the "Si1.HandleNotification" function command shall be the notification event retrieved from this notification token.

12.5.3 Notification from the LBA

This clause describes the procedure to send a notification to the SPB Manager from the LBA.

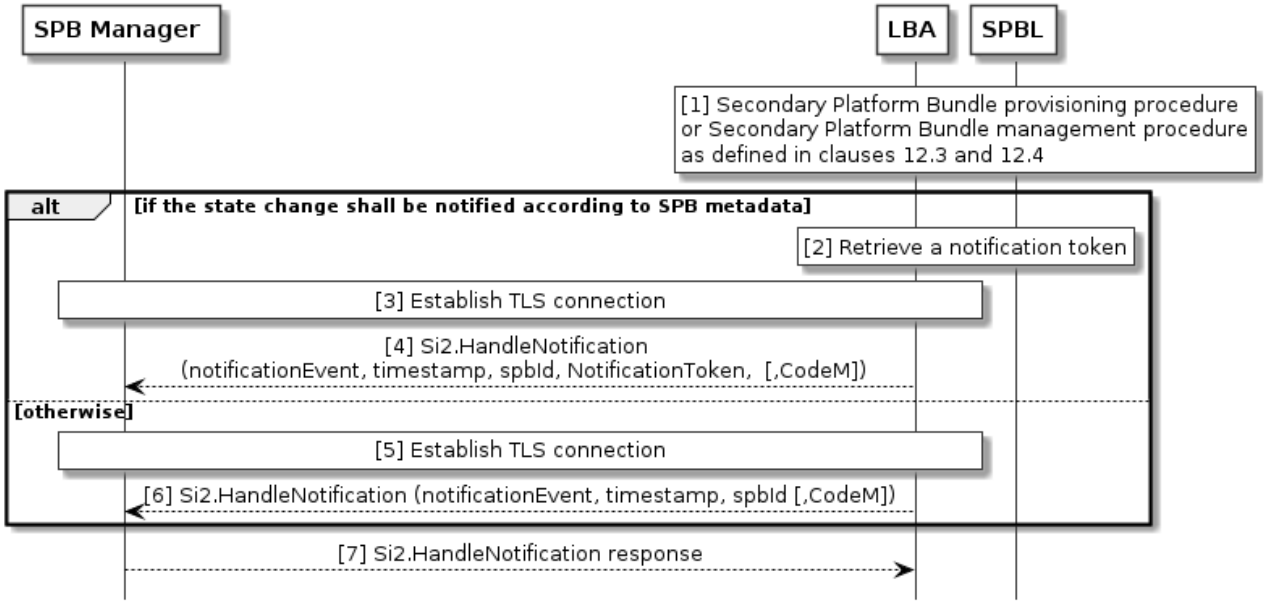

Figure 12.16: Handle notification from LBA

The notification procedure consists of the following steps:

1) A Secondary Platform Bundle container is installed or the state of the Secondary Platform Bundle container is changed as defined in clauses 12.3 and 12.4.

If the state of the Secondary Platform Bundle container shall be notified to the SPB Manager according to the configuration of the SPB metadata, the steps 2, 3, 4 and 7 shall be performed.

2) The LBA shall retrieve a notification token by using ANY_GET_PARAMETER with the index of OPERATION_TOKEN registry.

3) The LBA shall establish a TLS connection with the SPB Manager in server authentication mode.

4) The LBA shall call the "Si2.HandleNotification" function. The function command shall include a notification token.

The LBA shall store the notification token retrieved at step 2 until that notification token is successfully delivered to the SPB Manager. Once the notification is successfully delivered to the SPB Manager, the LBA shall delete that notification token.

Otherwise, the steps 5, 6 and 7 shall be performed.

5) The LBA shall establish a TLS connection with the SPB Manager in server authentication mode.

6) The LBA shall call the "Si2.HandleNotification" function without a notification token.

7) The SPB Manager shall respond to the LBA to notify a successful reception of the notification.

12.6 Interfaces and functions

12.6.1 Overview

This clause describes the interfaces and the functions used for the Secondary Platform Bundle provisioning and the Secondary Platform Bundle management operations.

Table 12.4 provides the list of functions over the interfaces associated with these operations.

Table 12.4: List of functions

Interface	Function	Function requester	Function provider
Si1	Si1.SelectSpb Si1.CreateSPReference Si1.FinalizePreparation Si1.CancelPreparation	Service Provider	Secondary Platform Bundle Manager
	Si1.HandleNotification	Secondary Platform Bundle Manager	Service Provider
Si2	Si2.GetSpbmCertificate Si2.GetBoundSpbImage	Local Bundle Assistant	Secondary Platform Bundle Manager
Si3	Si3.GetSspInfo Si3.SetSpbmCredential Si3.LoadBoundSpbInfo Si3.LoadBoundSpbSds Si3.LoadBoundSpbSeg Si3.GetSspCredential Si3.EnableSpb Si3.DisableSpb Si3.DeleteSpb	Local Bundle Assistant	Secondary Platform Bundle Loader

12.6.2 Common features

12.6.2.1 Common data types

The following ASN.1 types and objects are used for clause 12.6.

```
-- ASN1START

UUID ::= OCTET STRING (SIZE(16))

CodeM ::= OCTET STRING (SIZE(4..16))  -- CodeMatching to download the SPB container from SPB Manager

ChallengeS ::= OCTET STRING (SIZE(4..16))  -- CHALLENGE_S as defined in GlobalPlatform Open Firmware
-- Loader for Tamper Resistant Element [6]

VersionType ::= OCTET STRING (SIZE(2))  -- Same as VersionType in ETSI TS 103 666-1 [3]

IdTransac ::= OCTET STRING (SIZE(16))  -- Image session identifier (ID_TRANSAC) as defined in
-- GlobalPlatform Open Firmware Loader for Tamper Resistant Element [6]

OidSpecificInfoBlock ::= SEQUENCE
{
    aOid OBJECT IDENTIFIER, -- OID to identify the aOidSpecificInfo
```

```
    aOidSpecificInfo ANY DEFINED BY aOid OPTIONAL
}

-- ASN1STOP
```

12.6.2.2 SSP information

12.6.2.2.1 Introduction

The SSP information comprises sspInfoPublic and sspInfoProtected.

The SPB Manager shall perform eligibility check based on Annex C by using the received sspInfoPublic and sspInfoProtected.

12.6.2.2.2 Public SSP information

The sspInfoPublic is used during the capability negotiation procedure defined in clause 12.3.3 to provide the SPB Manager with the trusted public key identifiers and cryptographic algorithms supported by the Secondary Platform Bundle Loader allowing the SPB Manager to select an appropriate certificate and cryptographic algorithm.

The ASN.1 data structure of the public SSP information is described below.

```
-- ASN1START

SspInfoPublic ::= SEQUENCE
{
    aSpblSpecVerInfo VersionType,
    aSspGeneralCryptoInfo  SspCryptoInfo OPTIONAL,
    aSspFamilyCryptoInfoBlock SET OF SspFamilyCryptoInfoBlock OPTIONAL
}

SspCryptoInfo ::= SEQUENCE
{
    aSspPkIdListForSpbmVerification SET OF SubjectKeyIdentifier, -- List of Public Key identifiers
-- allowed for SPBM certificate chain verification
    aSspPkIdListForSpblVerification SET OF SubjectKeyIdentifier, -- List of Public Key identifiers
-- allowed for SPBL certificate chain verification
    aKeyAgreementAlgorithmList SET OF AlgorithmIdentifier, -- List of algorithm identifiers of key
-- agreement
    aCipherAlgorithmList SET OF EncryptionType -- List of data encryption algorithms
}

SspFamilyCryptoInfoBlock ::= SEQUENCE
{
    aSpbFamilyId UUID, -- SPB family identifier
    aSspFamilyCryptoInfo SspCryptoInfo OPTIONAL,

aSspOidCryptoInfoBlock SET OF SspOidCryptoInfoBlock OPTIONAL
}

SspOidCryptoInfoBlock ::= SEQUENCE
{
    aCustodianOid OBJECT IDENTIFIER, -- The OID of a custodian of the aSpbFamilyId
    aSspOidCryptoInfo SspCryptoInfo
}

-- ASN1STOP
```

aSpblSpecVerInfo: the release of the specification that is implemented by the Secondary Platform Bundle Loader. The first byte indicates the major version of the specification. The second byte indicates the minor version of the specification.

aSspPkIdListForSpbmVerification:

- For aSspGeneralCryptoInfo, the list indicates the Public Key identifiers supported by the Secondary Platform Bundle Loader that allows for the Secondary Platform Bundle Loader to verify the SPBM certificate chain.
- For aSspFamilyCryptoInfo, this list indicates the Public Key identifiers only allowed for the loading Secondary Platform Bundles with the aSpbFamilyId contained in the same SspFamilyCryptoInfoBlock.

- For aSspOidCryptoInfo, this list indicates the Public Key identifiers only allowed for loading Secondary Platform Bundles with the aSpbFamilyId contained in the same SspFamilyCryptoInfo and the OID contained in the same SspOidCryptoInfoBlock.

aSspPkIdListForSpblVerification:

- For aSspGeneralCryptoInfo, the list indicates the Public Key identifiers supported by the Secondary Platform Bundle Loader that allows for the SPB Manager to verify the SPBL certificate chain.
- For aSspFamilyCryptoInfo, the list indicates the Public Key identifiers only allowed for the loading of Secondary Platform Bundles with the aSpbFamilyId contained in the same SspFamilyCryptoInfoBlock.
- For aSspOidCryptoInfo, the list indicates the Public Key identifiers only allowed for the loading of Secondary Platform Bundles with the aSpbFamilyId contained in the same SspFamilyCryptoInfo and the OID contained in the same SspOidCryptoInfoBlock.

aKeyAgreementAlgorithmList:

- For aSspGeneralCryptoInfo, the list indicates the algorithm identifiers for key agreement algorithms supported by the Secondary Platform Bundle Loader.
- For aSspFamilyCryptoInfo, the list indicates the key agreement algorithms only allowed for the loading of Secondary Platform Bundles with the aSpbFamilyId contained in the same SspFamilyCryptoInfoBlock.
- For aSspOidCryptoInfo, the list indicates the key agreement algorithms only allowed for the loading of Secondary Platform Bundles with the aSpbFamilyId in the same SspFamilyCryptoInfoBlock and the OID contained in the same SspOidCryptoInfoBlock.

aCipherAlgorithmList:

- For aSspGeneralCryptoInfo, the list indicates data encryption algorithms supported by the Secondary Platform Bundle Loader.
- For aSspFamilyCryptoInfo, the list indicates the data encryption algorithms only allowed for the loading of Secondary Platform Bundles with the aSpbFamilyId contained in the same SspFamilyCryptoInfoBlock.
- For aSspOidCryptoInfo, the list indicates the data encryption algorithms only allowed for the loading of Secondary Platform Bundles with the aSpbFamilyId in the same aSspFamilyCryptoInfoBlock and the OID contained in the same aSspOidCryptoInfoBlock.

aSpbFamilyId: a family identifier supported by the Secondary Platform Bundle Loader.

aOid: the OID of a custodian of the family identifier aSpbFamilyId.

12.6.2.2.3 Protected SSP information

The Secondary Platform Bundle Loader shall provide the SPB Manager with sspInfoProtected containing the primary platform identifier and family identifier-specific SSP information.

The ASN.1 data structure of the protected SSP information is described below.

```
-- ASN1START

SspInfoProtected ::= SEQUENCE
{
    aPpIdentifier OCTET STRING, -- Primary Platform Identifier as defined in clause 7.5
    aMaxSpbSizeSupported INTEGER (0..MAX) OPTIONAL,
    aFamilySpecificSspInfoBlock FamilySpecificSspInfoBlock OPTIONAL -- Family Identifier-specific
-- SSP information
}

FamilySpecificSspInfoBlock ::= SEQUENCE
{
    aSpbFamilyId UUID, -- Family Identifier
    aFamilySpecificSspInfo ANY OPTIONAL, -- DEFINED BY aSpbFamilyId
    aOidSpecificSspInfoBlock SET OF OidSpecificInfoBlock -- OID specific SSP information
}

-- ASN1STOP
```

aPpIdentifier: the Primary Platform identifier as defined in clause 7.5.

aMaxSpbSizeSupported: it indicates the maximum size, in bytes, of the Secondary Platform Bundle container that the iSSP supports. The value of the aMaxSpbSizeSupported shall be the same as the value of MK_MEMORY_PARTITION_SIZE as defined in GlobalPlatform VPP - Concepts and Interfaces [7] clause 7.2. This information is optional and can be retrieved from the part number identifier.

aFamilySpecificSspInfo: it shall include the family identifier-specific SSP information which may be defined for that family identifier.

aSpbFamilyId: the family identifier of the Secondary Platform Bundle.

aOidSpecificSspInfoBlock: it shall include the family identifier-specific SSP information which may be defined by an organization that is responsible for that family identifier and referenced by aOID.

12.6.2.3 SPBM credential

The SPBM credential shall be delivered to the Secondary Platform Bundle Loader during the Secondary Platform Bundle provisioning procedure in clause 12.3.

The LBA shall provide the SPBM credential to the Secondary Platform Bundle Loader by calling the "Si3.GetSspCredential" function and obtain SSP credential as the response.

The ASN.1 data structure of SPBM credential is described below.

```
-- ASN1START

SpbmCredential ::= SEQUENCE
{
    aCodeM CodeM,
    aChallengeS ChallengeS OPTIONAL,
    aSpbFamilyId UUID,
    aCustodianOid OBJECT IDENTIFIER OPTIONAL,
    aSpbmKaCertificates SEQUENCE OF Certificate, -- Certificates in the SPBM certificate chain for
-- key agreement
    aSspCiPkIdToBeUsed SubjectKeyIdentifier,
    aSspCryptoToBeUsed EncryptionType
}

-- ASN1STOP
```

aCodeM: the value of the CODE_M as defined in GlobalPlatform Open Firmware Loader for Tamper Resistant Element [6]. It indicates the code matching identifier for a Secondary Platform Bundle image within a SPB Manager.

aChallengeS: the value of CHALLENGE_S as defined in GlobalPlatform Open Firmware Loader for Tamper Resistant Element [6]. The aChallengeS is generated by the SPB Manager and used in authentication of the Secondary Platform Bundle Loader.

aSpbFamilyId: the family identifier of the Secondary Platform Bundle.

aCustodianOid: the OID of a custodian of the aSpbFamilyId. The custodian shall be associated with a certification path.

aSpbmKaCertificates: SPBM Certificates for key agreement.

aSspCiPkIdToBeUsed: CI Public Key identifier for SPBL Certificate which shall be used by the Secondary Platform Bundle Loader for signature generation.

aSspCryptoToBeUsed: Selected data encryption algorithm which shall be used by the Secondary Platform Bundle Loader and the SPB Manager.

12.6.2.4 SSP credential

The SSP credential is delivered from the Secondary Platform Bundle Loader to the SPB Manager for authentication, key agreement, and for the binding of a Secondary Platform Bundle container.

The ASN.1 data structure of SSP credential is described below.

```
-- ASN1START

SspCredential ::= SEQUENCE
{
    aTbsSspImageSessionToken TbsSspImageSessionToken, -- SSP Token delivered to the SPB Manager in
-- clear text
    aSspImageSessionTokenSignature Signature, -- the signature of aTbsSspImageSessionToken
    aM-SSP EncryptedBlock, -- the encrypted data and integrity check of PrivateBlock
    aEncryptionType EncryptionType,
    aSpblCertChain SEQUENCE OF Certificate -- Certificates in the SPBL certificate chain to be used
-- to verify SPBL Certificate for digital signature
}

TbsSspImageSessionToken ::= SEQUENCE
{
    aIdTransac [0] IdTransac, -- ID_TRANSAC
    aEPkSpblKa [1] OCTET STRING,
    aSpbmKaPkIdToBeUsed [2] SubjectKeyIdentifier, -- SubjectKeyIdentifier of SPBM certificate for
-- key agreement
    aUuidL [3] UUID, -- UUID of OFL authority as defined in [6]
    aPartNumberId [4] UUID, -- Part Number identifier as defined in clause 7.7
    aChallengeS [5] ChallengeS OPTIONAL
}

PrivateBlock ::= SEQUENCE
{
    aCodeM [0] CodeM,
    aSspInfoProtected [1] SspInfoProtected,
    aSPBLCertificate [2] Certificate,
    aSPBLCodeM [3] BOOLEAN OPTIONAL
}

EncryptionType ::= ENUMERATED
{
    eGCM-AES-128 (0),
    eGCM-SM4-128 (1),
    eGCM-AES-256 (2)
}

EncryptedBlock ::= SEQUENCE
{
    aM OCTET STRING, -- Encrypted message
    aH OCTET STRING -- Integrity check
}

-- ASN1STOP
```

aTbsSspImageSessionToken: it contains:

- aIdTransac: the ID_TRANSAC as defined in GlobalPlatform Open Firmware Loader for Tamper Resistant Element [6].
- aEPkSpblKa: the SPBL ephemeral public key.
- aSpbmKaPkIdToBeUsed: the subject key identifier of the SPBM certificate for key agreement which shall be used to generate the first session key.
- aUuidL: the $UUID_L$ as defined in GlobalPlatform Open Firmware Loader for Tamper Resistant Element [6].
- **aPartNumberId:** the Part Number identifier as defined in clause 7.7 (the identifier of the Part Number in the format of UUID as defined in GlobalPlatform Open Firmware Loader for Tamper Resistant Element [6].) The aPartNumberId shall be used by the SPB Manager to identify the Primary Platform manufacturer and the model of the Primary Platform.
- aChallengeS: the value of CHALLENGE_S as defined in GlobalPlatform Open Firmware Loader for Tamper Resistant Element [6].

aSspImageSessionTokenSignature: the signature of aTbsSspImageSessionToken which can be verified by SPBL Certificate.

aM-SSP: EncryptedBlock of data containing an encrypted PrivateBlock in aM and its integrity check in aH. The content of the PrivateBlock and the generation of aM-SSP by the Secondary Platform Bundle Loader as defined in clause 12.6.5.5.2.

aEncryptionType: it indicates the data encryption algorithm used to generate the items of type EncryptedBlock.

EncryptedBlock: data structure containing the encrypted message and the integrity check.

aSpblCertChain: it contains Certificates used for the SPB Manager to verify SPBL Certificate.

PrivateBlock: it contains:

- aCodeM: the value of the CODE_M as defined in GlobalPlatform Open Firmware Loader for Tamper Resistant Element [6].
- aSspInfoProtected: the protected SSP information.
- aSPBLCertificate: the SPBL certificate that can be verified with aSpblCertChain.
- aSPBLCodeM: the boolean is true if CodeM is issued by the SPBL. If not present, the value shall be considered as false.

12.6.2.5 Bound SPB image

The Secondary Platform Bundle container shall be bound to the iSSP as defined in clause 12.6.4.3 and delivered to the LBA as the bound SPB image.

The ASN.1 data structure of the bound SPB image is described below.

```
-- ASN1START

BoundSpbImage ::= SEQUENCE
{
    aImageOwnerId UUID OPTIONAL, -- Identifier of the Secondary Platform Bundle owner
    aNumberSegment INTEGER, -- Number of segments in the bound SPB image
    aDoOperateParameter DoOperateParameter,
    aIdTransac IdTransac OPTIONAL,
    aServerNotifyBaseUrls SET OF UTF8String OPTIONAL,
    aImageMakerId UUID OPTIONAL, -- Identifier of the Secondary Platform Bundle maker
    aMetaDataImage MetaData OPTIONAL,
    aChangeSegmentParameters SEQUENCE OF ChangeSegmentParameter, -- List of Segment Descriptor
-- Structures
    aLoadSegmentParameters SEQUENCE OF LoadSegmentParameter -- List of encrypted segments
}

DoOperateParameter ::= SEQUENCE
{
    aSpbMetadata SpbMetadata,
    aEncryptionType EncryptionType,
    aM-IMD EncryptedBlock, --  the encrypted data and integrity check of Image Descriptor (IMD)
    aM-ARP EncryptedBlock ,--  the encrypted data and integrity check of ATK.ARP.ECDSA
    aM-TimeStamp EncryptedBlock, --  the encrypted data and integrity check of TIME_STAMP
    aSpbmToken SpbmToken,
    aSpbmCerts SEQUENCE OF Certificate -- List of the Certificates of the certification path from a
-- trusted certificate to the SPBM certificate
}

SpbmToken ::= SEQUENCE
{
    aTbsSpbmToken TbsSpbmToken,
    aSpbmTokenSignature Signature
}

TbsSpbmToken ::= SEQUENCE
{
    aEPkSbpmKa OCTET STRING,
    aIdTransac IdTransac
}

ChangeSegmentParameter ::= OCTET STRING -- Segment Descriptor Structure

LoadSegmentParameter ::= OCTET STRING -- Segment Structure

-- ASN1STOP
```

aImageOwnerId: Owner Identifier of the Secondary Platform Bundle container.

aNumberSegment: Number of Segment Structures in the bound SPB image.

aServerNotifyBaseUrls: URLs of the servers for notifications.

aImageMakerId: Identifier of the Secondary Platform Bundle maker as defined in GlobalPlatform Open Firmware Loader for Tamper Resistant Element [6].

aMetaDataImage: Metadata of the image from the Image Maker.

aM-IMD: EncryptedBlock of Image Descriptor as defined in GlobalPlatform Open Firmware Loader for Tamper Resistant Element [6].

aM-ARP: EncryptedBlock of ATK.ARP.ECDSA as defined in GlobalPlatform Open Firmware Loader for Tamper Resistant Element [6].

aM-TimeStamp: EncryptedBlock of TIME_STAMP as defined in GlobalPlatform Open Firmware Loader for Tamper Resistant Element [6].

aSpbmToken: Data structure containing TbsSpbmToken and signature of the TbsSpbmToken.

aTbsSpbmToken: Data structure containing the ephemeral public key of the SPB Manager and the image session identifier (ID_TRANSAC).

aSpbmCerts: List of the Certificate of the certification path from a trusted certificate to the SPBM certificate.

aDoOperateParameter: The parameter for the OFL_DO_OPERATE command as defined in GlobalPlatform Open Firmware Loader for Tamper Resistant Element [6] including the SPB metadata aSpbMetadata.

aChangeSegmentParameter: The parameter for the OFL_CHANGE_SEGMENT command as defined in GlobalPlatform Open Firmware Loader for Tamper Resistant Element [6].

aLoadSegmentParameter: The parameter for the OFL_LOAD_SEGMENT command as defined in GlobalPlatform Open Firmware Loader for Tamper Resistant Element [6].

12.6.2.6 SPB metadata

The SPB metadata contains specific information of the Secondary Platform Bundle. During the download procedure described in clause 12.3.3, the SPB metadata shall be provided to the LBA in a plaintext. After the Secondary Platform Bundle is successfully installed, the SPB metadata shall be accessible by the LBA via Si3 interface irrespective of the state of that Secondary Platform Bundle. The SPBL shall accept aSpbMetadata with a size of at least 76 bytes.

The ASN.1 data structure of the SPB metadata is described below.

```
-- ASN1START

SpbMetadata ::= SEQUENCE
{
    aSpbId UUID, -- Identifier of the Secondary Platform Bundle
    aSpbFamilyId UUID, --Family identifier of the Secondary Platform Bundle
    aCustodianOid OBJECT IDENTIFIER, -- OID of the custodian defining the specific requirements
-- applied to this Secondary Platform Bundle
    aSupportedCustodianList SEQUENCE OF OBJECT IDENTIFIER OPTIONAL, -- List of custodian identifiers
-- associated with supported certification path to download the Secondary Platform Bundle
    aSpbNotificationConfig SEQUENCE OF SpbNotificationConfig OPTIONAL, -- notifications to be sent
-- to the service provider
  aFamilySpecificData ANY OPTIONAL, -- DEFINED BY aSpbFamilyId Metadata defined in the present
-- document for the family identifier indicated by aSpbFamilyId
    aOidSpecificData  SET OF OidSpecificInfoBlock OPTIONAL -- a set of metadata defined by a
-- custodian(s) for the family identifier indicated by a SpbFamilyId
}
SpbNotificationConfig ::= SEQUENCE
{
    aSpbNotificationEvent SpbNotificationEvent,
    aRecipientAddress UTF8String
}

SpbNotificationEvent ::= BIT STRING
{
    eNotificationStatus-Eligibility (0),
    eNotificationStatus-MaxRetryAttempt (1),
```

```
    eNotificationStatus-UserRejection (2),
    eNotificationStatus-SPBDownload (3),
    eNotificationStatus-SPBInstallation (4),
    eNotificationStatus-LocalSPBEnable (5),
    eNotificationStatus-LocalSPBDisable (6),
    eNotificationStatus-LocalSPBDelete (7),
    eNotificationStatus-RemoteSPBEnable (8),
    eNotificationStatus-RemoteSPBDisable (9),
    eNotificationStatus-RemoteSPBDelete (10)
}
-- ASN1STOP
```

The SPB metadata shall include the followings:

- aSpbId: identifier of the Secondary Platform Bundle.
- aSpbFamilyId: family identifier of the Secondary Platform Bundle as defined in clause 9.7.
- aCustodianOid: OID of one of the custodians associated with a SpbFamilyId which defines specific requirement (e.g. trusted CIs, product certification, operational modes of the terminal) applied to this Secondary Platform Bundle.

The SPB metadata may include the followings:

- aSupportedCustodianList: list of OIDs of custodians associated with supported certification path used to load the Secondary Platform Bundle. If the aSupportedCustodianList contains multiple OIDs, the first OBJECT IDENTIFIER denotes the most preferred custodian to select a certification path. The aSupportedCustodianList may contain the aCustodianOid.
- aSpbNotificationConfig: it includes the configuration set by a service provider as per a notification recipient. The configuration shall include the address of a notification recipient and the list of events which shall be notified.

NOTE: Eligibility check and maximum retry attempts notification status are only intended to be used for Si1 notifications.

- aFamilySpecificData: family identifier-specific metadata defined as per the family identifier.
- aOidSpecificData: family identifier-specific metadata defined by custodian(s) of the family identifier. The aOidSpecificData may consists of multiple OidSpecificInfoBlock data structures. Each OidSpecificInfoBlock shall have a custodian-defined metadata and the OID identifying that custodian-defined metadata.

12.6.2.7 Terminal information

The terminal information contains details about the capabilities of the terminal. It is delivered by the LBA to the SPB Manager.

The ASN.1 data structure of terminal information is described below.

```
-- ASN1START

TerminalInfo ::= SEQUENCE
{
    aLbaSpecVerInfo VersionType,
    aFamilySpecificTerminalInfoBlock SET OF FamilySpecificTerminalInfoBlock OPTIONAL
}

FamilySpecificTerminalInfoBlock ::= SEQUENCE
{
    aSpbFamilyId UUID, -- Family Identifier
    aFamilySpecificTerminalInfo ANY OPTIONAL, -- DEFINED BY aSpbFamilyId
    aOidSpecificTerminalInfoBlock SET OF OidSpecificInfoBlock -- OID specific terminal information
}

-- ASN1STOP
```

aLbaSpecVerInfo: the release of the specification that is implemented by the LBA. The first byte indicates the major version of the specification. The second byte indicates the minor version of the specification.

aSpbFamilyId: a family identifier of the Secondary Platform Bundle.

a**FamilySpecificTerminalInfoBlock:** it shall include the family identifier-specific terminal information which may be defined for that family identifier.

aOidSpecificTerminalInfoBlock: it shall include family identifier-specific terminal information which may be defined by an organization that is responsible for that family identifier and referenced by the OID.

12.6.2.8 Notification token

The notification token contains the information about the state change of the Secondary Platform Bundle container.

The Secondary Platform Bundle Loader shall generate the notification token after installation, enabling, disabling or deleting of the Secondary Platform Bundle container if it is configured in the SPB metadata. Prior to the generation of the notification token, the notification counter of the firmware session of the Secondary Platform Bundle container as described in clause 7.3.1.5 shall be incremented by one. If the maximum value is reached, the counter shall return to the initial value. After generating the notification token, the Secondary Platform Bundle Loader shall set the value of the notification token data object into OPERATION_TOKEN registry.

The LBA shall use ANY_GET_PARAMETER command with the index of OPERATION_TOKEN registry to retrieve the most recently generated notification token.

The ASN.1 data structure of the notification token is described below.

```
-- ASN1START

NotificationToken ::= SEQUENCE
{
    aTbhNotificationToken TbhNotificationToken,
    aNotificationTokenHash OCTET STRING (SIZE(32))
}

TbhNotificationToken ::= SEQUENCE
{
    aSpbId UUID, -- the public identifier of the Secondary Platform Bundle
    aNotificationEvent Si3NotificationEvent,
    aCounter Integer -- the notification counter value managed in the firmware session of the
Secondary Platform Bundle identified by aSpbId
}

Si3NotificationEvent ::= ENUMERATED
{
    eNotificationStatus-SPBInstallation (4),
    eNotificationStatus-LocalSPBEnable (5),
    eNotificationStatus-LocalSPBDisable (6),
    eNotificationStatus-LocalSPBDelete (7),
    eNotificationStatus-RemoteSPBEnable (8),
    eNotificationStatus-RemoteSPBDisable (9),
    eNotificationStatus-RemoteSPBDelete (10)
}

-- ASN1STOP
```

aSpbId: the identifier of the Secondary Platform Bundle. The aSpbId shall be the Public image UUID.

aNotificationEvent: it indicates the procedure related to this notification.

aCounter: the notification counter value managed in the firmware session of the Secondary Platform Bundle identified by the aSpbId.

aNotificationTokenHash: the hashed value generated by an HMAC-SHA-256 [32] of the message being the string concatenating the aSpbId, the aNotificationEvent, the aCounter, and the Primary Platform identifier (of 32 bytes, as defined in clause 7.5), and with the secondary platform private identifier described in clause 9.4.5 as the key. aNotificationTokenHash is therefore computed with following parameters:

aNotificationTokenHash = HMAC-SHA-256 (message, key) where:

message = aSpbId | aNotificationEvent | aCounter | aPpId, and key = aPrivateSpbId

NOTE: The Primary Platform Identifier is used to compute the aNotificationTokenHash but is not included in the aTbhNotificationToken.

12.6.3 Si1 interface

12.6.3.1 Overview

The Si1 interface is used between the service provider and the SPB Manager to prepare the download of a Secondary Platform Bundle.

The binding of the Si1 interface shall be performed over Hypertext Transfer Protocol version 2 (HTTP/2) as defined in IETF RFC 7540 [25] and the Transport Layer Security (TLS) version 1.3 or higher in mutual authentication mode as defined in IETF RFC 8446 [26].

The service provider shall be in charge of managing the connection establishment to the SPB Manager for the Si1 interface.

The service provider shall use HTTP POST request message with HTTP path 'etsi/issp/si1/asn1' to deliver any function command over the Si1 interface.

12.6.3.2 Si1 common headers

12.6.3.2.1 Si1 command header

The ASN1 structure of all Si1 commands contains the Si1CommandHeader defined as follows:

```
-- ASN1START

Si1CommandHeader ::= SEQUENCE
{
    aFunctionRequesterId OCTET STRING (SIZE(4   ..64)), -- Identification of the function requester
    aFunctionCallId OCTET STRING (SIZE(4..64)) -- Identification of the function call
}

-- ASN1STOP
```

aFunctionRequesterId: identifier of the function requester.

aFunctionCallId: identifier of the function call. This identifier is used to manage function call retries.

12.6.3.2.2 Si1 response header

The ASN1 structure of all Si1 responses contains the Si1ResponseHeader defined as follows:

```
-- ASN1START

Si1ExecutionStatus ::= ENUMERATED
{
    eSi1ExecutionStatus-Executed-Success (0),
    eSi1ExecutionStatus-Failed (1),
    eSi1ExecutionStatus-Executed-WithWarning (2)
}

Si1ResponseHeader ::= SEQUENCE
{
    aFunctionExecutionStatus Si1ExecutionStatus -- Function execution status
}

-- ASN1STOP
```

aFunctionExecutionStatus: indicates the status after the execution of the function.

12.6.3.3 Si1 error codes

This clause describes the error codes used to indicate an error over the Si1 interface.

Table 12.5 gives the applicability matrix according to the Si1 function.

```
-- ASN1START

Si1ErrorCode::= ENUMERATED
{
    eSpbIdUnknown (0), -- the Secondary Platform Bundle identifier is unknown to the SPB Manager
    eSpbIdNotAllowed (1), -- the function caller is not allowed to perform this function on this
-- SpbId
    eSpbIdNotAvailable (2), -- the Secondary Platform Bundle identifier is not available
    eSpbIdAlreadyLinked (3), -- the Secondary Platform Bundle identifier is already in use
    eSpbTypeUnknown (4), -- the type of Secondary Platform Bundle is unknown to the SPB Manager
    eSpbTypeNotAllowed (5), -- the function caller is not allowed to perform this function on this
-- SpbType
    eSpbTypeNotAvailable (6), -- there is no more Secondary Platform Bundle identifier available for
-- this SpbType
    eSpbTypeMismatch (7), -- the Secondary Platform Bundle identifier has a different SpbType
    eCodeMUnknown (8), -- the CodeM is unknown to the SPB Manager
    eCodeMNotAllowed (9), -- the function caller is not allowed to perform this function for this
-- CodeM
    eTaskTypeUnknown (10), -- the type of task is unknown to the SPB Manager
    eTaskNotAllowed (11) -- the function caller is not allowed to handle this type of task
}

-- ASN1STOP
```

Table 12.5: Si1 command/error codes

Si1 function	eSpbIdUnknown	eSpbIdNotAllowed	eSpbIdNotAvailable	eSpbIdAlreadyLinked	eSpbTypeUnknown	eSpbTypeNotAllowed	eSpbTypeNotAvailable	eSpbTypeMismatch	eCodeMUnknown	eCodeMNotAllowed	eTaskTypeUnknown	eTaskNotAllowed
Si1.SelectSpb	•	•	•		•	•	•	•	•	•		
Si1.CreateSPReference	•	•	•	•					•	•	•	•
Si1.FinalizePreparation	•	•							•	•		
Si1.CancelPreparation	•	•							•			

12.6.3.4 Si1.SelectSpb

12.6.3.4.1 Command

The "Si1.SelectSpb" function shall be used by the service provider during the Secondary Platform Bundle selection as defined in clause 12.3.2.2.

The service provider shall use the "Si1.SelectSpbm" function to select a Secondary Platform Bundle that matches the terminal and the SSP capabilities.

The body part of the HTTP POST request for the "Si1.SelectSpbm" function command shall contain Si1SelectSpbCommand defined as follows:

```
-- ASN1START

Si1SelectSpbCommand ::= SEQUENCE
{
    aSi1CommandHeader Si1CommandHeader, -- Header of the command
    aSpbId UUID OPTIONAL, -- Identifier of the Secondary Platform Bundle
    aSpbType OCTET STRING (SIZE(1..128)) OPTIONAL, -- Free information defined by the service
-- provider
    aPpIdentifier OCTET STRING OPTIONAL, -- Identifier of the Primary Platform as defined in clause
-- 7.5
    aCodeM CodeM OPTIONAL, -- CodeMatching to download the SPB container from SPB Manager
    aFlagFinalize BOOLEAN DEFAULT FALSE, -- Indicates if the Finalize Preparation procedure will be
-- executed
    aCustodianSpecificInfoBlock OidSpecificInfoBlock OPTIONAL, -- Custodian-specific information
    aServiceProviderSpecificInfoBlock OidSpecificInfoBlock OPTIONAL -- Service provider specific
-- information
}
```

```
-- ASN1STOP
```

aSi1CommandHeader: header of the command as defined in clause 12.6.3.2.1. It may be used by aNotificationReceiverId in subsequent Si1.HandleNotification calls related to this selection.

aSpbId: identifier of the Secondary Platform Bundle to reserve.

aSpbType: type of Secondary Platform Bundle in which the SPB Manager shall select an available Secondary Platform Bundle identifier.

aPpIdentifier: the primary platform identifier to link with the Secondary Platform Bundle reserved by the Si1.SelectSpb function.

aCodeM: CodeM to be linked with the Secondary Platform Bundle reserved by the Si1.SelectSpb function. This parameter shall be present if the creation of the service provider reference defined in clause 12.3.2.3 has been previously executed.

aFlagFinalize: Boolean that indicates whether the "Si1.FinalizePreparation" function will be called later.

aCustodianSpecificInfoBlock: specific parameter which may be defined by the custodian of the family identifier issuing the command. How this parameter is handled by the SPB Manager is out of scope of the present document.

aServiceProviderSpecificInfoBlock: specific parameter which may be defined by the service provider. How this parameter is handled by the SPB Manager is out of scope of the present document.

12.6.3.4.2 Procedure

Upon reception of the "Si1.SelectSpb" function command, the SPB Manager shall:

- Store the value of aSi1CommandHeader.
- If the Secondary Platform Bundle identifier (spbId) was provided as input data:
  - return an error with the code eSpbIdNotAvailable if the spbId is not available;
  - return an error with the code eSpbIdUnknown if the spbId does not exist;
  - return an error with the code eSpbTypeMismatch if a Secondary Platform Bundle type (spbType) was provided as input data and does not match the type of the spbId.
- If a Secondary Platform Bundle type (spbType) was provided as input data:
  - return an error with the code eSpbTypeUnknown if the spbType is unknown to the SPB Manager;
  - return an error with the code eSpbTypeNotAvailable if the SPM Manager cannot find an available spbId corresponding to the requested sbpType.
- Return an error with the code eCodeMNotAllowed if a CodeM was provided as input data and is already linked to another Secondary Platform Bundle identifier.
- Store the CodeM if provided as input data and is not known to the SPB Manager.
- Reserve a Secondary Platform Bundle among those available in its inventory and that corresponds to the requested spbId and/or sbpType.
- Link the CodeM with the reserved spbId.
- Link the reserved spbId to the primary platform identifier if the function command contains the primary platform identifier (aPpIdentifier).
- Memorize whether the "Si1.FinalizePreparation" function will be called later. If aFlagFinalize is not present, it is considered as set to FALSE.

- Build Si1SelectSpbResponse containing either an error code if one of the above step has failed or the selected Secondary Platform identifier (spbId) together with its type (sbpType) and, optionally the Primary Platform Identifier (aPpIdentifier) if it was provided in the command and the CodeM (aCodeM) if it was provided in the incoming command. Si1SelectSpbResponse may also contain family identifier and/or service provider specific information. Their content is not in the scope of the present document.
- Send the response to the service provider.

12.6.3.4.3 Response

The body part of the HTTP POST response for the "Si1.SelectSpb" function shall contain Si1SelectSpbResponse defined as follows:

```
-- ASN1START

Si1SelectSpbResponse ::= SEQUENCE
{
    aSi1ResponseHeader Si1ResponseHeader, -- Header of the response
    aSi1SelectSpbResult CHOICE
    {
        aSi1SelectSpbOk Si1SelectSpbOk,
        aSi1SelectSpbError Si1ErrorCode
    },
    aCustodianSpecificInfoBlock OidSpecificInfoBlock OPTIONAL, -- Custodian-specific information
    aServiceProviderSpecificInfoBlock OidSpecificInfoBlock OPTIONAL -- Service provider specific
-- information

}

Si1SelectSpbOk ::= SEQUENCE
{
    aSpbId UUID, -- Identifier of the selected Secondary Platform Bundle
    aSpbType OCTET STRING (SIZE(1..128)), -- Free information defined by the service provider
    aPpIdentifier OCTET STRING OPTIONAL, -- Identifier of the Primary Platform linked with the
-- selected Secondary Platform Bundle
    aCodeM CodeM OPTIONAL -- CodeMatching linked with the selected Secondary Platform Bundle
}

-- ASN1STOP
```

aSi1ResponseHeader: header of the response as defined in clause 12.6.3.2.2.

aSpbId: identifier of the Secondary Platform Bundle reserved by the SPB Manager.

aSpbType: type of Secondary Platform Bundle tied to aSpbId.

aPpIdentifier: identifier of the primary platform linked with aSpbId, if present in the incoming command.

aCodeM: CodeM to linked with the aSpbId, if present in the incoming command.

aCustodianSpecificInfoBlock: specific parameter which may be defined by the custodian of the family identifier issuing the response. How this parameter is handled by the SPB Manager is out of scope of the present document.

aServiceProviderSpecificInfoBlock: specific parameter which may be defined by the service provider. How this parameter is handled by the SPB Manager is out of scope of the present document.

aFamilySpecificSelectSpbmResponse: family identifier-specific parameter which may be defined for that family identifier. How this parameter is handled by the SPB Manager is out of scope of the present document.

12.6.3.5 Si1.CreateSPReference

12.6.3.5.1 Command

The "Si1.CreateSPReference" function shall be used by the service provider during the procedure of creation of a service provider reference as defined in clause 12.3.2.3.

The service provider may use the "Si1.CreateSPReference" function to create a reference shared between the service provider and the SPB Manager. This reference, i.e. CodeM shall be provided to the End User by the service provider as part of the activation code, allowing the End User to trigger the download procedure as defined in clause 12.3.3.

The body part of the HTTP POST request for the "Si1.CreateSPReference" function command shall contain Si1.CreateSPReferenceCommand defined as follows:

```
-- ASN1START

Si1TaskType ::= ENUMERATED
{
    eSi1TaskType-DownloadSPB (0), -- Download of a Secondary Platform Bundle
    eSi1TaskType-EgibilityInfo (1) -- Retrieval of eligibility information
}

Si1CreateSPReferenceCommand ::= SEQUENCE
{
    aSi1CommandHeader Si1CommandHeader, -- Header of the command
    aCodeM CodeM OPTIONAL, -- CodeMatching to download the SPB container from SPB Manager
    aSpbId UUID OPTIONAL, -- Identifier of the Secondary Platform Bundle
    aTaskType Si1TaskType DEFAULT eSi1TaskType-DownloadSPB, -- Type of action expected from the SPB
-- Manager
    aCustodianSpecificInfoBlock OidSpecificInfoBlock OPTIONAL, -- Custodian-specific information
    aServiceProviderSpecificInfoBlock OidSpecificInfoBlock OPTIONAL -- Service provider specific
-- information

}

-- ASN1STOP
```

aSi1CommandHeader: header of the command as defined in clause 12.6.3.2.1. It may be used by aNotificationReceiverId in subsequent Si1.HandleNotification calls related to the CodeM provided as input parameter or generated by the SPB Manager.

aSpbId: identifier of the Secondary Platform Bundle. This parameter shall be present if the Secondary Platform Bundle selection procedure has been executed first, else it shall be ignored.

aCodeM: CodeM generated by the service provider.

aTaskType: type of task associated with the reference.

aCustodianSpecificInfoBlock: specific parameter which may be defined by the custodian of the family identifier issuing the command. How this parameter is handled by the SPB Manager is out of scope of the present document.

aServiceProviderSpecificInfoBlock: specific parameter which may be defined by the service provider. How this parameter is handled by the SPB Manager is out of scope of the present document.

12.6.3.5.2 Procedure

Upon reception of the "Si1.CreateSPReference" function command, the SPB Manager shall:

- Store the value of aSi1CommandHeader.
- Return an error with the code eTaskTypeUnknown if the Si1TaskType is not eSi1TaskType-DownloadSPB.
- Return an error with the code eTaskNotAllowed if the function caller is not allowed to use the Si1TaskType.

NOTE: How the function caller is allowed to use is not in the scope of the present document.

- Generate a CodeM if it was not provided as input data and ensure that it is unique on its own context.
- Return an error with the code eCodeMNotAllowed if the CodeM was provided as input data and is already linked to another Secondary Platform Bundle identifier.
- Store the CodeM.
- Build Si1CreateSPReferenceResponse containing either an error code if one of the above step has failed or the CodeM (aCodeM) provided as input data or generated by the SPB Manager and the Secondary Platform identifier (spbId) if it was provided as input data. Si1CreateSPReferenceResponse may also contain family identifier and/or service provider specific information. Their content is not in the scope of the present document.
- Send the response to the service provider.

12.6.3.5.3 Response

The body part of the HTTP POST response for the "Si1.CreateSPReference" function shall contain Si1CreateSPReferenceResponse defined as follows:

```
-- ASN1START

Si1CreateSPReferenceResponse ::= SEQUENCE
{
    aSi1ResponseHeader Si1ResponseHeader, -- Header of the response
    aSi1CreateSPReferenceResult CHOICE
    {
        aSi1CreateSPReferenceOk Si1CreateSpReferenceOk,
        aSi1CreateSPReferenceError Si1ErrorCode
    },
    aCustodianSpecificInfoBlock OidSpecificInfoBlock OPTIONAL, -- Custodian-specific information
    aServiceProviderSpecificInfoBlock OidSpecificInfoBlock OPTIONAL -- Service provider specific
-- information

}

Si1CreateSPReferenceOk ::= SEQUENCE
{
    aCodeM CodeM, -- CodeMatching linked with the selected Secondary Platform Bundle
    aSpbId UUID OPTIONAL -- Identifier of the selected Secondary Platform Bundle
}

-- ASN1STOP
```

aSi1ResponseHeader: header of the response as defined in clause 12.6.3.2.2.

aCodeM: CodeM generated by the SPB manager if not present in the incoming command or CodeM as it was in the incoming command.

aSpbId: identifier of the Secondary Platform Bundle as if was in the incoming command.

aCustodianSpecificInfoBlock: specific parameter which may be defined by the custodian of the family identifier issuing the response. How this parameter is handled by the SPB Manager is out of scope of the present document.

aServiceProviderSpecificInfoBlock: specific parameter which may be defined by the service provider. How this parameter is handled by the SPB Manager is out of scope of the present document.

12.6.3.6 Si1.FinalizePreparation

12.6.3.6.1 Command

The "Si1.FinalizePreparation" function shall be used by the service provider to finalize the preparing procedure as defined in clause 12.3.2.2.

If the selection of the Secondary Platform Bundle procedure, as defined in clause 12.3.2.2 has been executed after the creation of the CodeM procedure, as defined in clause 12.3.2.3, the service provider may use the "Si1.FinalizePreparation" function to indicate that its internal procedures are completed, e.g. the provisioning of its technical platforms or data bases.

If the service provider has set aFlagFinalize to TRUE in the "Si1.SelectSpb" function command, the SPB Manager shall wait for the completion of the Secondary Platform Bundle selection process as described in clause 12.3.2.2 (i.e. after it has sent the response to the "Si1.FinalizePreparation" function related to this Secondary Platform Bundle) to continue with the Bound SPB image download as defined in clause 12.3.3.2.

The body part of the HTTP POST request for the "Si1.FinalizePreparation" function command shall contain Si1.FinalizePreparationCommand defined as follows:

```
-- ASN1START

Si1FinalizePreparationCommand ::= SEQUENCE
{
    aSi1CommandHeader Si1CommandHeader, -- Header of the command
    aCodeM CodeM, -- TBC
    aCustodianSpecificInfoBlock OidSpecificInfoBlock OPTIONAL, -- Custodian-specific information
    aServiceProviderSpecificInfoBlock OidSpecificInfoBlock OPTIONAL -- Service provider specific
```

```
-- information

}

-- ASN1STOP
```

aSi1CommandHeader: header of the command as defined in clause 12.6.3.2.1. It may be used by aNotificationReceiverId in subsequent Si1.HandleNotification calls related to aCodeM.

aCodeM: reference to the preparing procedure to finalize.

aCustodianSpecificInfoBlock: specific parameter which may be defined by the custodian of the family identifier issuing the command. How this parameter is handled by the SPB Manager is out of scope of the present document.

aServiceProviderSpecificInfoBlock: specific parameter which may be defined by the service provider. How this parameter is handled by the SPB Manager is out of scope of the present document.

12.6.3.6.2 Procedure

Upon reception of the "Si1.FinalizePreparation" function command, the SPB Manager shall:

- Store the value of aSi1CommandHeader.
- Verify the CodeM provided as input data.
- Return an error with the code eCodeMUnknown if the CodeM is unknown to the SPB Manager.
- Return an error with the code eCodeMNotAllowed if the CodeM is not linked to a Secondary Platform Bundle identifier.
- Build Si1FinalizePreparationResponse containing either an error code if the above step has failed or the CodeM (aCodeM) provided as input data. Si1FinalizePreparationResponse may also contain family identifier and/or service provider specific information. Their content is not in the scope of the present document.
- Send the response to the service provider.
- Allow the bound SPB image download procedure as defined in clause 12.3.3.2.

12.6.3.6.3 Response

The body part of the HTTP POST response for the "Si1.finalizePreparation" function shall contain Si1FinalizePreparationResponse defined as follows:

```
-- ASN1START

Si1FinalizePreparationResponse ::= SEQUENCE
{
    aSi1ResponseHeader Si1ResponseHeader, -- Header of the response
    aSi1FinalizePreparationResult CHOICE
    {
        aSi1FinalizePreparationOk Si1FinalizePreparationOk,
        aSi1FinalizePreparationError Si1ErrorCode
    },
    aCustodianSpecificInfoBlock OidSpecificInfoBlock OPTIONAL, -- Custodian-specific information
    aServiceProviderSpecificInfoBlock OidSpecificInfoBlock OPTIONAL -- Service provider specific
-- information
}

Si1FinalizePreparationOk ::= SEQUENCE
{
    aCodeM CodeM -- CodeMatching linked with the selected Secondary Platform Bundle
}

-- ASN1STOP
```

aSi1ResponseHeader: header of the response as defined in clause 12.6.3.2.2.

aCodeM: CodeM as it was in the incoming command.

aCustodianSpecificInfoBlock: specific parameter which may be defined by the custodian of the family identifier issuing the response. How this parameter is handled by the SPB Manager is out of scope of the present document.

aServiceProviderSpecificInfoBlock: specific parameter which may be defined by the service provider. How this parameter is handled by the SPB Manager is out of scope of the present document.

12.6.3.7 Si1.CancelPreparation

12.6.3.7.1 Command

The "Si1.CancelPreparation" function shall be used by the service provider to cancel a pending preparation procedure as defined in clause 12.3.2.

The body part of the HTTP POST request for the "Si1.CancelPreparation" function command shall contain Si1CancelPreparationCommand defined as follows:

```
-- ASN1START

Si1CancelPreparationCommand ::= SEQUENCE
{
    aSi1CommandHeader Si1CommandHeader, -- Header of the command
    aCodeM CodeM OPTIONAL, -- CodeMatching to download the SPB container from SPB Manager
    aSpbId UUID OPTIONAL, -- Identifier of the Secondary Platform Bundle
    aCustodianSpecificInfoBlock OidSpecificInfoBlock OPTIONAL, -- Custodian-specific information
    aServiceProviderSpecificInfoBlock OidSpecificInfoBlock OPTIONAL -- Service provider specific
-- information

}


-- ASN1STOP
```

aSi1CommandHeader: header of the command as defined in clause 12.6.3.2.1. It may be used by aNotificationReceiverId in subsequent Si1.HandleNotification calls related to aCodeM or aSpbId.

aCodeM: task's reference to cancel. This parameter shall be present if aSpbId is not provided as input parameters.

aSpbId: identifier of the Secondary Platform Bundle associated to the procedure to cancel. This parameter shall be present if aCodeM is not provided as input parameters.

aCustodianSpecificInfoBlock: specific parameter which may be defined by the custodian of the family identifier issuing the command. How this parameter is handled by the SPB Manager is out of scope of the present document.

aServiceProviderSpecificInfoBlock: specific parameter which may be defined by the service provider. How this parameter is handled by the SPB Manager is out of scope of the present document.

12.6.3.7.2 Procedure

Upon reception of the "Si1.CancelPreparation" function command, the SPB Manager shall:

- Store the value of aSi1CommandHeader.
- Return an error with the code eCodeMUnknown if a CodeM is provided as input data and is unknown to the SPB Manager.
- Return an error with the code eCodeMNotAllowed if the bound SPB image download procedure as defined in clause 12.3.3.2 associated with the Secondary Platform Bundle identifier linked to the CodeM provided as input data is completed.
- Return an error with the code eSpbIdUnknown if a aSpbId is provided as input data and is unknown to the SPB Manager.
- Return an error with the code eSpbIdNotAllowed if a aSpbId is provided as input data and is not linked with the CodeM provided as input data.
- Cancel any pending procedure associated with the CodeM and/or the SpbId provided as input parameter(s), e.g. download procedure.

- Unreserved the Secondary Platform Bundle identifier provided as input data and/or linked to the CodeM provided as input data.
- Remove any reference to the CodeM if provided as input data.
- Build Si1CancelPreparationResponse containing either an error code if one of the above step has failed or either the CodeM (aCodeM) or the Secondary Platform Bundle identifier (aSpbId) if provided as input data and the linked Secondary Platform identifier if any. Si1CancelPreparationResponse may also contain family identifier and/or service provider specific information. Their content is not in the scope of the present document.
- Send the response to the service provider.

12.6.3.7.3 Response

The body part of the HTTP POST response for the "Si1.CancelPreparation" function shall contain Si1CancelPreparationResponse defined as follows:

```
-- ASN1START

Si1CancelPreparationResponse ::= SEQUENCE
{
    aSi1ResponseHeader Si1ResponseHeader, -- Header of the response
    aSi1CancelPreparationResult CHOICE
    {
        aSi1CancelPreparationOk Si1CancelPreparationOk,
        aSi1CancelPreparationError Si1ErrorCode
    },
    aCustodianSpecificInfoBlock OidSpecificInfoBlock OPTIONAL, -- Custodian-specific information
    aServiceProviderSpecificInfoBlock OidSpecificInfoBlock OPTIONAL -- Service provider specific
-- information
}

Si1CancelPreparationOk ::= SEQUENCE
{
    aCodeM CodeM OPTIONAL, -- CodeMatching linked with the selected Secondary Platform Bundle
    aSpbId UUID OPTIONAL -- Identifier of the selected Secondary Platform Bundle
}

-- ASN1STOP
```

aSi1ResponseHeader: header of the response as defined in clause 12.6.3.2.2.

aCodeM: CodeM as it was in the incoming command.

aSpbId: identifier of the Secondary Platform Bundle linked to aCodeM if aCodeM was provided as input data or aSpbId as if was in the incoming command.

aCustodianSpecificInfoBlock: specific parameter which may be defined by the custodian of the family identifier issuing the response. How this parameter is handled by the SPB Manager is out of scope of the present document.

aServiceProviderSpecificInfoBlock: specific parameter which may be defined by the service provider. How this parameter is handled by the SPB Manager is out of scope of the present document.

12.6.3.8 Si1.HandleNotification

12.6.3.8.1 Command

The "Si1.HandleNotification" function shall be used by the SPB Manager to send any notifications as agreed with the service provider owning the pending related task. The agreement of the notifications to send is outside the scope of the present document.

The body part of the HTTP POST request for the "Si1.HandleNotification" function shall contain Si1HandleNotificationBlock defined as follows:

```
-- ASN1START

Si1HandleNotificationBlock ::= SEQUENCE
```

```
{
    aHandleNotificationHeader HandleNotificationHeader, -- header of the notification
    aCodeM CodeM,
    aSpbId UUID OPTIONAL, -- Identifier of the Secondary Platform Bundle
    aSpbType OCTET STRING (SIZE(1..128)) OPTIONAL, -- Free information defined by the service
-- provider
    aPpIdentifier OCTET STRING OPTIONAL, -- Identifier of the Primary Platform
    aTimeStamp GeneralizedTime,
    aNotificationEvent Si1NotificationEvent,
    aNotificationEventStatus Si1ExecutionStatus OPTIONAL,
    aCustodianSpecificInfoBlock OidSpecificInfoBlock OPTIONAL, -- Custodian-specific information
    aServiceProviderSpecificInfoBlock OidSpecificInfoBlock OPTIONAL -- Service provider specific
-- information
}

HandleNotificationHeader ::= SEQUENCE
{
    aNotificationReceiverId OCTET STRING (SIZE(4    ..64)), -- Identification of the notification
-- recipient
    aNotificationCallId OCTET STRING (SIZE(4..64)) -- Identification of the function caller in the
-- context of the notification recipient
}

Si1NotificationEvent ::= ENUMERATED
{
    eNotificationStatus-Eligibility (0),
    eNotificationStatus-MaxRetryAttempt (1),
    eNotificationStatus-UserRejection (2),
    eNotificationStatus-SPBDownload (3),
    eNotificationStatus-SPBInstallation (4),
    eNotificationStatus-LocalSPBEnable (5),
    eNotificationStatus-LocalSPBDisable (6),
    eNotificationStatus-LocalSPBDelete (7),
    eNotificationStatus-RemoteSPBEnable (8),
    eNotificationStatus-RemoteSPBDisable (9),
    eNotificationStatus-RemoteSPBDelete (10)
}

-- ASN1STOP
```

aNotificationReceiverId: identifier of the recipient of the notification. It may equal to the function requester identity extracted from the last request-response function related to the same pending task, e.g. to the same download procedure.

aNotificationCallId: identifier of the function caller in the context of the recipient of the notification. It may be equal to the function caller identity extracted from the last request-response function related to the same pending task, e.g. to the same download procedure.

aCodeM: task's reference to cancel. This parameter shall be present if aSpbId is not provided as input parameters.

aSpbId: identifier of the Secondary Platform Bundle associated to the procedure to cancel. This parameter shall be present if aCodeM is not provided as input parameters.

aSpbType: type of Secondary Platform Bundle in which the SPB Manager shall select an available Secondary Platform Bundle identifier.

aPpIdentifier: identifier of the primary platform to link with the Secondary Platform Bundle reserved by the Si1.SelectSpb function.

aTimeStamp: indicates the date/time when the operation has been performed or when the notification has been received by the SPB Manager.

aNotificationEvent: indicates the step reached by the procedure that was executed.

aNotificationEventStatus: indicates the status after the execution of the notification.

aCustodianSpecificInfoBlock: specific parameter which may be defined by the custodian of the family identifier issuing the command. How this parameter is handled by the SPB Manager is out of scope of the present document.

aServiceProviderSpecificInfoBlock: specific parameter which may be defined by the service provider. How this parameter is handled by the SPB Manager is out of scope of the present document.

12.6.3.8.2 Procedure

Table 12.6 indicates which parameters shall be present depending on aNotificationEvent.

Table 12.6: Si1 notification parameters

aNotificationEvent	aNotificationReceiverId	aNotificationCallId	aCodeM	aSpbId	aSpbType	aPpIdentifier	aTimeStamp	aNotificationEvent	aNotificationEventStatus
eNotificationStatus-Eligibility	•	•	•				•	•	
eNotificationStatus-MaxRetryAttempt	•	•	•	•	•	•	•	•	•
eNotificationStatus-UserRejection	•	•	•	•	•	•	•	•	•
eNotificationStatus-SPBDownload	•	•	•	•	•	•	•	•	•
eNotificationStatus-SPBInstallation	•	•	•	•	•	•	•	•	•
eNotificationStatus-LocalSPBEnable	•	•		•	•	•	•	•	
eNotificationStatus-LocalSPBDisable	•	•		•	•	•	•	•	
eNotificationStatus-LocalSPBDelete	•	•		•	•	•	•	•	
eNotificationStatus-RemoteSPBEnable	•	•		•	•	•	•	•	
eNotificationStatus-RemoteSPBDisable	•	•		•	•	•	•	•	
eNotificationStatus-RemoteSPBDelete	•	•		•	•	•	•	•	

12.6.4 Si2 interface

12.6.4.1 Overview

The Si2 interface is used between the LBA and SPB Manager to provide a transport of the bound Secondary Platform Bundle image and the management commands on the Secondary Platform Bundles installed in the iSSP.

The binding of the Si2 interface shall be performed over Hypertext Transfer Protocol version 2 (HTTP/2) as defined in IETF RFC 7540 [25] and the Transport Layer Security (TLS) version 1.3 in server authentication mode as defined in IETF RFC 8446 [26].

The LBA shall be in charge of managing the connection establishment to the SPB Manager for the Si2 interface.

The LBA shall use HTTP POST request message with HTTP path 'etsi/issp/si2/asn1' to deliver any function command over the Si2 interface.

12.6.4.2 Si2.GetSpbmCertificate

12.6.4.2.1 Command

The "Si2.GetSpbmCertificate" function shall be used by the LBA during the capability negotiation procedure as defined in clause 12.3.3.1.

The LBA shall use the "Si2.GetSpbmCertificate" function to provide the SPB Manager with the public SSP information (SspInfoPublic) as defined in clause 12.6.2.2.2 and terminal information (TerminalInfo) as defined in clause 12.6.2.7.

The body part of the HTTP POST request for the "Si2.GetSpbmCertificate" function command shall contain Si2GetSpbmCertificateCommand defined as follows:

```
-- ASN1START

Si2GetSpbmCertificateCommand ::= SEQUENCE
{
    aSspInfoPublic SspInfoPublic, -- Public SSP information as defined in clause 12.6.2.2.2
```

```
    aTerminalInfo TerminalInfo -- Terminal information as defined in clause 12.6.2.7
}

-- ASN1STOP
```

12.6.4.2.2 Procedure

On reception of "Si2.GetSpbmCertificate" function command, the SPB Manager shall:

1) Perform eligibility check based on Annex C as follows:

   a) The SPB Manager shall verify that the aSpblSpecVerInfo contained in the aSspInfoPublic and aLbaSpecVerInfo contained in aTerminalInfo are supported by itself. If a version is not supported, the SPB Manager shall return eNotSupportedLbaVersion or eNotSupportedSpblVersion (the error indicating that the version of the Secondary Platform Bundle Loader or the LBA is not supported).

2) Determine the family identifier of the Secondary Platform Bundle container to be provisioned as follows:

   a) If the SPB Manager supports only one family identifier, the SPB Manager shall select that family identifier. If there is a aSspFamilyCryptoInfoBlock and no aSspGeneralCryptoInfo inside the aSspInfoPublic, the SPB Manager shall check whether one of the family identifiers contained in the aSspFamilyCryptoInfoBlock is supported. If supported, the SPB Manager shall select that family identifier. If not supported, the SPB Manager shall return eNotSupportedFamilyId (the error indicating that the family identifier is not supported).

   b) If the SPB Manager supports multiple family identifiers:

      - If there is only one SspFamilyCryptoInfoBlock data structure containing a family identifier supported by the SPB Manager, the SPB Manager shall select that family identifier. If there is no SspFamilyCryptoInfoBlock data structure containing a family identifier supported by the SPB Manager, the SPB Manager shall return eNotSupportedFamiyId (the error indicating that the family identifier is not supported).

      - If there are multiple aSspFamilyCryptoInfoBlock data structure containing the family identifier supported by the SPB Manager inside the aSspInfoPublic or there is only aSspCryptoInfo inside the aSspInfoPublic, the SPB Manager shall return eSpblSelectOneFamilyId (the error indicating that one family identifier shall be selected by the Secondary Platform Bundle Loader).

3) Set the selected family identifier into the aSpbFamilyId.

4) Using the selected family identifier, select one of aSspCryptoInfo, aSspFamilyCryptoInfo and aSspOidCryptoInfo inside the aSspInfoPublic as follows:

   a) If there is a SspFamilyCryptoInfoBlock data structure containing the selected family identifier, the SPB Manager shall select that SspFamilyCryptoInfoBlock data structure. Using the selected SspFamilyCryptoInfoBlock data structure:

      - If the SPB Manager supports only one custodian for the selected family identifier and there is a SspOidCryptoInfoBlock data structure containing the OID of that custodian, the SPB Manager shall select the aSspOidCyrptoInfo data structure contained in that SspOidCryptoInfoBlock data structure. If there is no SspOidCryptoInfoBlock data structure containing the OID of that custodian, the SPB Manager shall select the aSspFamilyCryptoInfo inside the SspFamilyCryptoInfoBlock data structure.

      - If the SPB Manager supports multiple custodians for the selected family identifier and there is only one SspOidCryptoInfoBlock data structure containing the Oid of one of the custodians supported by the SPB Manager, the SPB Manager shall select the aSspOidCryptoInfo contained in that SspOidCryptoInfoBlock data structure.

      - If the SPB Manager supports multiple custodians for the selected family identifier and there are multiple SspOidCryptoInfoBlock data structures containing the OIDs of custodians supported by the SPB Manager, the SPB Manager shall return eSpblSelectOneOid (the error indicating that one custodian shall be selected by the Secondary Platform Bundle Loader).

- If there is no SspOidCryptoInfo data structure containing the OID of a custodian supported by the SPB Manager, the SPB Manager shall select the aSspFamilyCryptoInfo.

b) If there is no SspFamilyCryptoInfoBlock data structure containing the selected family identifier, the SPB Manager shall select the aSspGeneralCryptoInfo inside the aSspInfoPublic.

5) Using the selected SspCryptoInfo data structure, choose the following:

a) An SPB Manager certificate for key agreement that can be verified by the trusted public key indicated by one of the trusted public key identifiers in the aSspPkIdListForSpbmVerification. If none of the trusted public key identifiers in the aSspPkIdListForSpbmVerification is supported, the SPB Manager shall return eNotSupportedPkIdSpbmVerification. The algorithmIdentifier of the selected certificate shall be one of the algorithmIdentifier in aKeyAgreementAlgorithmList. If the algorithmIdentifier of the selected certificate is not supported, the SPB Manager shall return eNotSupportedKeyAgreementAlgorithm.

b) An SPB Manager certificate for digital signature that can be verified by the trusted public key indicated by one of the trusted public key identifiers in the aSspPkIdListForSpbmVerification. If any trusted public key identifiers in the aSspPkIdListForSpbmVerification is not supported, the SPB Manager shall return eNotSupportedPkIdSpbmVerification.

c) One of trusted public key identifier(s) in the aSspPkIdListForSpblVerification that shall be used by the Secondary Platform Bundle Loader to select its certificate(s). The SPB Manager shall set the selected trusted public key identifier into aSspCiPkIdToBeUsed. If any trusted public key identifiers in the aSspPkIdListForSpblVerification is not supported, the SPB Manager shall return eNotSupportedPkIdSpblVerification.

d) One of algorithmIdentifiers in the aCipherAlgorithmList that shall be used by the Secondary Platform Bundle Loader and the SPB Manager for data encryption. The SPB Manager shall set the selected algorithmIdentifier into aSspCryptoToBeUsed. If none of the algorithmIdentifier in the aCipherAlgorithmList is supported, the SPB Manager shall return eNotSupportedEncryptionAlgorithm.

6) Generate a new random octet string for aChallengeS which shall be used to authenticate the Secondary Platform Bundle Loader.

7) Build Si2GetSpbmCertificateResponse containing the SPB Manager certificate for key agreement, the aSspCiPkIdToBeUsed, the aSspCryptoToBeUsed, aChallengeS, and the aSpbFamilyId and optionally the certificate chain for SPB Manager certificate for key agreement.

12.6.4.2.3 Response

The body part of the HTTP POST response for the "Si2.GetSpbmCertificate" function shall contain Si2GetSpbmCertificateResponse defined as follows:

```
-- ASN1START

Si2GetSpbmCertificateResponse ::= CHOICE
{
    aSi2GetSpbmCertificateOk Si2GetSpbmCertificateOk,
    aSi2GetSpbmCertificateError Si2GetSpbmCertificateErrorCode
}

Si2GetSpbmCertificateOk ::= SEQUENCE
{
    aSspPkIdForSpblVerification SubjectKeyIdentifier, -- Public Key identifier for trust anchor that
-- is used to verify Secondary Platform Bundle Loader certification path
    aSspCryptoToBeUsed EncryptionType, -- data encryption algorithm to be used
    aSpbmKaCert Certificate, -- SPB Manager certificate for key agreement
    aSpbFamilyId UUID,
    aCustodianOid OBJECT IDENTIFIER OPTIONAL,
    aChallengeS ChallengeS, -- the random challenge for SPBL authentication
    aSpbmCertChain SEQUENCE OF Certificate OPTIONAL -- certificates to be used to construct
-- certification path for verification of aSpbmKaCert
}

Si2GetSpbmCertificateErrorCode ::= ENUMERATED
{
    eNotSupportedLbaVersion (0), -- the version of the Secondary Platform Bundle Loader is not
-- supported
    eNotSupportedSpblVersion (1), -- the version of the LBA is not supported
```

```
    eNotSupportedFamilyId (2), -- any family identifier is not supported
    eNotSupportedOid (3), -- any Oid for the selected family identifier is not supported
    eNotSupportedKeyAgreementAlgorithm (4), -- any algorithmIdentifier in the selected
-- aKeyAgreementAlgorithmList is not supported
    eNotSupportedEncryptionAlgorithm (5), -- any data encryption algorithm in the selected
-- aCipherAlgorithmList is not supported
    eNotSupportedPkIdSpbmVerification (6), -- any trusted public key identifiers in the selected
-- aSspPkIdListForSpbmVerification is not supported
    eNotSupportedPkIdSpblVerification (7), -- any trusted public key identifiers in the selected
-- aSspPkIdListForSpblVerification is not supported
    eSpblSelectOneFamilyId (8), -- one family identifier shall be selected by the Secondary Platform
-- Bundle Loader
    eSpblSelectOneOid (9) -- one Oid shall be selected by the Secondary Platform Bundle Loader
}

-- ASN1STOP
```

aSspPkIdForSpblVerification: CI Public Key identifier for SPBL Certificate which shall be used by the Secondary Platform Bundle Loader for signature generation.

aSspCryptoToBeUsed: data encryption algorithm selected for data encryption which shall be used by the Secondary Platform Bundle Loader and the SPB Manager.

aSpbmKaCert: SPBM certificate for key agreement.

aSpbFamilyId: the family identifier of the Secondary Platform Bundle.

aCustodianOid: the OID of a custodian for the aSpbFamilyId.

aChallengeS: the value of CHALLENGE_S as defined in GlobalPlatform Open Firmware Loader for Tamper Resistant Element [6]. The aChallengeS is generated by the SPB Manager and used in authentication of the Secondary Platform Bundle Loader.

aSpbmCertChain: the certificates to be used to construct certification path for verification of SPBM certificate for key agreement.

12.6.4.3 Si2.GetBoundSpbImage

12.6.4.3.1 Command

The "Si2.GetBoundSpbImage" function shall be used by the LBA during the download procedure as defined in clause 12.3.3.

The LBA shall use "Si2.GetBoundSpbImage" function to provide the SPB Manager with SSP credential (SspCredential) as defined in clause 12.6.2.4 and terminal information (TerminalInfo) as defined in clause 12.6.2.7.

The LBA shall provide RequestType to the SPB Manager to indicate the request type. The LBA shall set:

- "RequestBoundSpbImage" to the RequestType if the LBA requests a bound Secondary Platform Bundle image.
- "RequestSpbMetadata" to the RequestType if the LBA requests only SPB metadata before requesting a bound Secondary Platform Bundle image to check the SPB metadata and, if configured, require the user intent.
- "BoundSpbImageByTransacId" to the RequestType if the LBA requests a bound Secondary Platform Bundle image after receiving the SPB metadata via "Si2.GetBoundSpbImage" function with "RequestSpbMetadata" as the requestType.

The body part of the HTTP POST request for the "Si2.GetBoundSpbImage" function command shall contain Si2GetBoundSpbImageCommand defined as follows:

```
-- ASN1START

Si2GetBoundSpbImageCommand ::= SEQUENCE
{
    aSspCredential SspCredential, -- SSP credential as defined in clause 12.6.2.4
    aTerminalInfo TerminalInfo, -- Terminal Information as defined in clause 12.6.2.7
    aRequestType ENUMERATED
{
```

```
        eRequestBoundSpbImage (0),
        eRequestSpbMetadata (1),
        eBoundSpbImageByTransacId (2)
    }
}

-- ASN1STOP
```

12.6.4.3.2 Procedure

On reception of the Si2.GetBoundSpbImage function command, the SPB Manager shall check the value of the aRequestType contained in the Si2GetBoundSpbImage and perform the procedure as described below:

1) If the value of aRequestType is "eRequestBoundSpbImage (0)" or "eRequestSpbMetadata (1)", the SPB Manager shall:

   a) Extract the aSpbmKaPkIdToBeUsed contained in the aSspImageSessionToken in the aSspCredential.

   b) Selecta aSpbmKaCertificate which can be verified by the CI certificate indicated in aSpbmKaPkIdToBeUsed.

   c) Generate the first session key as defined in GlobalPlatform Open Firmware Loader for Tamper Resistant Element [6]. The first session key shall be generated by using the private key corresponding to the SPB Manager certificate for key agreement and the aEPkSpblKa contained in the aTbsSspImageSessionToken in the aSspCredential.

NOTE 1: The first session key is the same as 'KS1' in GlobalPlatform Open Firmware Loader for Tamper Resistant Element [6].

   d) Decrypt the aM-SSP contained in the aSspCredential by using the first session key. The SPB Manager shall use the algorithm identified by the aEncryptionType. The SPB Manager shall obtain the Secondary Platform Bundle Loader certificate aSPBLCertificate in the PrivateBlock by decrypting the aM-SSP.

   e) Verify the Secondary Platform Bundle Loader certificate by using the trust public key which is identified by the aSspPkIdForSpblVerification. The Secondary Platform Bundle Loader certificate shall be verified based on the certification path verification as defined in clause 12.2.1.1.4. If the verification fails, the SPB Manager shall return eInvalidSpblCertificate.

   f) Verify the aSspImageSessionTokenSignature by using the Secondary Platform Bundle Loader certificate.

   g) Store the aIdTransac contained in the aTbsSspImageSessionToken and attach aChallengeS to the aIdTransac to manage this on-going image session.

   h) Find the Secondary Platform Bundle identifier corresponding to the aCodeM contained in the PrivateBlock. If there is not the Secondary Platform Bundle identifier corresponding to aCodeM, the SPB Manager shall call the "Si1.HandleNotification" function. The function command shall contain the aNotificationEvent, the aCodeM, the aSspInfoProtected and aTerminalInfo. The aNotificationEvent shall be set to eNotificationStatus_Eligibility. The SPB Manager shall suspend the bound SPB image download procedure until the service provider has completed the Secondary Platform Bundle selection process as defined in clause 12.3.2.2.

   i) Perform the eligibility check based on Annex C by using the aSspInfoProtected contained in PrivateBlock and aTerminalInfo. The SPB Manager shall:

      - Verify the aPpIdentifier contained in the aSspInfoProtected.
      - Verify the aFamilySpecificSspInfoBlock contained in the aSspInfoProtected (out of scope of the present document).
      - Verify the aFamilySpecificTerminalInfo and aOidSpecificInfo contained in the aTerminalInfo (out of scope of the present document).
      - Check whether the selected Secondary Platform Bundle image is supported by the iSSP based on aPpIdentifier, aSspInfoProtected, and aTerminalInfo. If the selected Secondary Platform Bundle image is not supported, the SPB Manager shall return eInvalidSpbImage.

j) Build aSpbMetadata corresponding to the selected Secondary Platform Bundle image (out of scope of the present document). The SPB Manager may construct aFamilySpecificData and aOidSpecificData in aSpbMetadata based on aFamilySpecificSspInfoBlock and aFamilySpecificTerminalInfoBlock contained in Si2GetBoundSpbImageCommand.

k) If the aRequestType is "eRequestSpbMetadata (1)", the SPB Manager shall:

- Bind the aSspCredential to the aIdTransac.
- Return aSpbMetadata to the LBA as the response of the "Si2.GetBoundSpbImage" function.

2) If the aRequestType is " eBoundSpbImageByTransacId (2)", the SPB Manager shall verify that the aSspCredential is verified in step 1 with aRequestType set to "eRequestSpbMetadata (1)". If the verification fails, the SPB Manager shall return eInvalidBoundSpbImageByTransacId.

3) After successfully finishing the above steps, the SPB Manager shall:

a) Generate TIME_STAMP and generate aM-TimeStamp by encrypting the TIME_STAMP by using the first session key and the encryption algorithm identified by the aSspCyrptoToBeUsed determined in the capability negotiation procedure as defined in clause 12.3.3.1.

b) Generate an SPB Manager's ephemeral key pair. The domain parameter used to generate the ephemeral key pair shall be the same as the one used by the SPB Manager certificate for key agreement.

c) Generate the second session key as defined in GlobalPlatform Open Firmware Loader for Tamper Resistant Element [6]. The second session key shall be generated with the SPB Manager's ephemeral private key and the aEPkSpblKa contained in aTbsSspImagesessionToken.

NOTE 2: The second session key is the same as 'KS2' in GlobalPlatform Open Firmware Loader for Tamper Resistant Element [6].

d) Generate aSpbmToken data structure containing the SPB Manager's ephemeral public key and the aIdTransac as defined in clause 12.6.2.5.

e) Select an SPB Manager certificate for digital signature which can be verified by the trusted public key indicated by one of the trusted public key identifiers in the aSspPkIdListForSpbmVerification. The selected SPB Manager certificate for digital signature shall be verified by the same trusted public key as the one used to verify the SPB Manager certificate for key agreement determined by the "Si2.GetSpbmCertificate" function.

f) Compute the aSpbmTokenSignature over the aTbsSpbmToken using the private key corresponding to the SPB Manager certificate for digital signature.

g) Obtain aM-IMD by encrypting the Image Descriptor (IMD) by using the second session key as defined in GlobalPlatform Open Firmware Loader for Tamper Resistant Element [6].

h) Obtain aM-ARP by encrypting the ATK.ARP.ECDSA by using the second session key as defined in GlobalPlatform Open Firmware Loader for Tamper Resistant Element [6].

i) Build aDoOperateParameter data structure as defined in clause 12.6.2.5.

j) Build aChangeSegmentParameters data structure as defined in clause 12.6.2.5. The aChangeSegmentParameters shall be the list of ChangeSegmentParameters. Each ChangeSegmentParameter shall be generated by encrypting the Segment Descriptor Structure by using the second session key as defined in GlobalPlatform Open Firmware Loader for Tamper Resistant Element [6].

k) Build aBoundSpbImage data structure as defined in clause 12.6.2.5.

l) Return the aBoundSpbImage data structure to the LBA as the response of the "Si2.GetBoundSpbImage" function.

12.6.4.3.3 Response

The body part of the HTTP POST response of the "Si2.GetBoundSpbImage" shall contain Si2GetBoundSpbImageResponse defined as follows:

```
-- ASN1START

Si2GetBoundSpbImageResponse ::= CHOICE
{
    aSi2GetBoundSpbImageOk Si2GetBoundSpbImageOk,
    aSi2GetBoundSpbImageError Si2GetBoundSpbImageError
}

Si2GetBoundSpbImageOk ::= CHOICE
{
    aBoundSpbImage BoundSpbImage, -- aBoundSpbImage as defined in clause 12.6.2.5
    aSpbMetadata SpbMetadata -- SPB metadata as defined in clause 12.6.2.6
}

Si2GetBoundSpbImageError ::= SEQUENCE
{
    aSi2GetBoundSpbImageErrorCode Si2GetBoundSpbImageErrorCode,
    aFamilySpecificErrorContainer FamilySpecificErrorContainer OPTIONAL
}

Si2GetBoundSpbImageErrorCode ::= ENUMERATED
{
    eInvalidSpblCertificate (0), -- the Secondary Platform Bundle Loader certificate is not verified
    eInvalidCodeM (1), -- the aCodeM has not been reserved by the Service Provider
    eInvalidChallengeS (2), -- the ChallengeS is not valid
    eInvalidSpbImage (3), -- the selected Secondary Platform Bundle image is not supported by the
-- iSSP
    eInvalidBoundSpbImageByTransacId (4), -- the transacId is not valid
    eInvalidCommand (5) -- the command is not valid
}

FamilySpecificErrorContainer ::= SEQUENCE
{
    aSpbFamilyId UUID,
    aFamilySpecificError ANY OPTIONAL, -- DEFINED BY aSpbFamilyId
    aOidSpecificError OidSpecificInfoBlock  OPTIONAL
}

-- ASN1STOP
```

aBoundSpbImage: Secondary Platform Bundle image bound to the Secondary Platform Bundle Loader.

aSpbMetadata: the SPB metadata of the Secondary Platform Bundle corresponding to the aCodeM.

aSi2GetBoundSpbImageErrorCode:

- **eInvalidSpblCertificate:** the error indicating that the SPBL certification path could not be verified.
- **eInvalidCodeM:** the error indicating that the aCodeM has not been reserved by the Service Provider.
- **eInvalidChallengeS:** the error indicating that aChallengeS is not valid in this image session.
- **eInvalidSpbImage:** the error indicating that the Secondary Platform Bundle corresponding to the aCodeM is not compatible with the SSP.
- **eInvalidBoundSpbImageByTransacId:** the error indicating that the aSspCredential containing this aIdTransac has not been verified with aRequestType set to "eRequestSpbMetadata (1)".
- **eInvalidCommand:** the error to be used in all cases not stated in the above list.

aSpbFamilyId: the family identifier of the Secondary Platform Bundle referenced by the aCodeM.

aFamilySpecificError: a family identifier-specific error container which may be defined for the aSpbFamilyId.

aOidSpecificError: a family identifier-specific error container which may be defined by the custodian indicated in aOid contained in aOidSpecificError.

12.6.4.4 Si2.HandleNotification

12.6.4.4.1 Command

The "Si2.HandleNotification" function shall be used by the LBA to send any notification about the result of the Secondary Platform Bundle management to the SPB Manager.

The body part of the HTTP POST request for the "Si2.HandleNotification" function command shall contain Si2HandleNotificationCommand defined as follows:

```
-- ASN1START

Si2HandleNotificationCommand ::= SEQUENCE
{
    aNotificationEvent Si2NotificationEvent,
    aTimeStamp GeneralizedTime,
    aSpbId UUID, -- Identifier of the Secondary Platform Bundle related to this notification
    aNotificationToken NotificationToken OPTIONAL, -- notification token as defined in clause
-- 12.6.2.8
    aCodeM CodeM OPTIONAL, -- CodeMatching linked with the Secondary Platform Bundle to download
    aFamilySpecificNotificationCommandContainer FamilySpecificNotificationCommand OPTIONAL

}

Si2NotificationEvent ::= ENUMERATED
{
    eNotificationStatus-SPBdownload (3),
    eNotificationStatus-SPBInstallation (4),
    eNotificationStatus-LocalSPBEnable (5),
    eNotificationStatus-LocalSPBDisable (6),
    eNotificationStatus-LocalSPBDelete (7),
    eNotificationStatus-RemoteSPBEnable (8),
    eNotificationStatus-RemoteSPBDisable (9),
    eNotificationStatus-RemoteSPBDelete (10)
}

FamilySpecificNotificationCommand ::= SEQUENCE
{
    aSpbFamilyId UUID,
    aFamilySpecificNotificationCommand ANY OPTIONAL, -- DEFINED BY aSpbFamilyId
    aCustodianSpecificNotificationCommand OidSpecificInfoBlock OPTIONAL
}

-- ASN1STOP
```

aNotificationEvent: it indicates the procedure related to this notification.

aTimeStamp: it indicates the time when this notification message is constructed by the LBA.

aSpbId: identifier of the Secondary Platform Bundle related to aNotificationEvent.

aNotificationToken: notification token which contains the information about the state change of the Secondary Platform Bundle container in the iSSP as defined in clause 12.6.2.8.

aCodeM: the CodeMatching identifier linked with the Secondary Platform Bundle to download. If the Si2NotificationEvent is 'eNotificationStatus_SPBInstallationError', this parameter shall be present.

aFamilySpecificNotificationCommand: family identifier-specific Si2HandleNotificationCommand which may be defined for that family identifier. How this parameter is handled by the SPB Manager is out of scope of the present document.

aCustodianSpecificNotificationCommand: Custodian-specific Si2HandleNotificationCommand which may be defined by a custodian identified by aOid inside the aCustodianSpecificNotificationCommand.

12.6.4.4.2 Procedure

On reception of Si2HandleNotificationCommand, the SPB Manager shall respond to the LBA to notify a successful reception of the notification. The response may contain a family identifier-specific notification response or a custodian-specific notification response.

12.6.4.4.3 Response

The body part of the HTTP POST response for the "Si2.HandleNotification" function shall contain Si2HandleNotificationResponse defined as follows:

```
-- ASN1START

Si2HandleNotificationResponse ::= SEQUENCE
{
    aFamilySpecificNotificationResponseContainer FamilySpecificNotificationResponse OPTIONAL
}

FamilySpecificNotificationResponse ::= SEQUENCE
{
    aSpbFamilyId UUID,
    aFamilySpecificNotificationResponse ANY OPTIONAL, -- DEFINED BY aSpbFamilyId
    aCustodianSpecificNotificationResponse OidSpecificInfoBlock OPTIONAL
}
-- ASN1STOP
```

aFamilySpecificNotificationResponse: a family identifier-specific Si2HandleNotificationResponse which may be defined for that family identifier.

aCustodianSpecificNotificationResponse: a custodian-specific Si2HandleNotificationResponse which may be defined by the custodian identified by aOid inside the aCustodianSpecificNotificationResponse.

12.6.5 Si3 interface

12.6.5.1 Overview

The Si3 interface is used between the LBA and the Secondary Platform Bundle Loader.

The LBA shall use the Si3 interface to transfer a bound Secondary Platform Bundle image and management commands to the Secondary Platform Bundle Loader.

The OFL agent host in the LBA and the OFL service hosted in the Secondary Platform Bundle Loader shall exchange commands, responses, and events over the Si3 interface as defined in as defined in GlobalPlatform VPP - OFL VNP Extension [9] with the additional commands, responses and registry defined in clauses 12.6.5.2, 12.6.5.3 and 12.6.5.4.

12.6.5.2 Registries

The OFL service Gate in the Secondary Platform Bundle Loader shall support the registry entries defined in clause 7.3.1.2.

12.6.5.3 Commands

The OFL service Gate in the Secondary Platform Bundle Loader shall support the commands defined in clause 7.3.1.3.

12.6.5.4 Responses

The OFL service Gate in the Secondary Platform Bundle Loader shall support the responses defined in clause 7.3.1.4.

12.6.5.5 Functions

12.6.5.5.1 Si3.GetSspInfo

The "Si3.GetSspInfo" function shall be used by the LBA during the capability negotiation procedure as defined in clause 12.3.3.1.

The LBA shall use the "Si3.GetSspInfo" function to retrieve aSspInfoPublic from the Secondary Platform Bundle Loader.

The "Si3.GetSspInfo" function command shall be GET_SSP_INFO.

The parameter of GET_SSP_INFO command is defined as follows:

```
-- ASN1START

GetSspInfoParameter ::= SEQUENCE
{
    aSpbFamilyId UUID OPTIONAL, -- the family identifier of the Secondary Platform Bundle container
-- to be loaded
    aCustodianOid OBJECT IDENTIFIER OPTIONAL -- The OID of a custodian of the aSpbFamilyId
}

-- ASN1STOP
```

On reception of the "Si3.GetSspInfo" function command, the Secondary Platform Bundle Loader shall:

1) Set the GET_SSP_INFO command parameter into the GET_SSP_INFO_PARAMETER registry.

2) Build aSspInfoPublic data structure defined in clause 12.6.2.2.2 as follows:

   a) The Secondary Platform Bundle Loader shall set the release of the specification that is implemented by the Secondary Platform Bundle Loader into the aSpblSpecVerInfo.

   b) If GET_SSP_INFO contains both the aSpbFamilyId and the aCustodianOid, the Secondary Platform Bundle Loader shall build aSspInfoPublic containing:

      - One aSspFamilyCryptoInfoBlock which shall contain the aSpbFamilyId and only one aSspOidCryptoInfoBlock if there is a configuration for both of the aSpbFamilyId and the aOid. The aSspOidCryptoInfoBlock shall have aCustodianOid and aSspOidCryptoInfo which contains the list of trusted public key identifiers and the list of algorithm identifiers which are allowed to be used for loading of the Secondary Platform Bundles with that aSpbFamilyId and that aCustodianOid.

      - One aSspFamilyCryptoInfoBlock which shall contain the aSpbFamilyId and aSspFamilyCryptoInfo if there is a configuration for the aSpbFamilyId but not for the aCustodianOid. The aSspFamilyCryptoInfo shall contain the list of trusted public key identifiers and the list of algorithm identifiers which are allowed to be used for loading of the Secondary Platform Bundles with that aSpbFamilyId.

      - aSspGeneralCryptoInfo if there is no configuration for the aSpbFamilyId. The aSspGeneralCryptoInfo shall contain the list of trusted public key identifiers and the list of algorithm identifiers which are not associated with any family identifier and any custodian.

   c) If GET_SSP_INFO command parameter contains only aSpbFamilyId, the Secondary Platform Bundle Loader shall build aSspInfoPublic containing:

      - One aSspFamilyCryptoInfoBlock which shall contain the aSpbFamilyId if there is a configuration for the aSpbFamilyId. The aSspFamilyCryptoInfoBlock may contain the set of aSspOidCryptoInfoBlocks as many as the configurations for the custodians of that aSpbFamilyId. Each aSspOidCryptoInfoBlock shall have aCustodianOid and aSspOidCryptoInfo which contains the list of trusted public key identifiers and the list of algorithm identifiers which are allowed to be used for loading of the Secondary Platform Bundles with that aSpbFamilyId and that aCustodianOid. The aSspFamilyCryptoInfoBlock may also contain aSspFamilyCryptoInfo.

      - aSspGeneralCryptoInfo if there is no configuration for the aSpbFamilyId. The aSspGeneralCryptoInfo shall contain the list of trusted public key identifiers and the list of algorithm identifiers which are not associated with any family identifier and any custodian.

   d) If GET_SSP_INFO command parameter is empty, the Secondary Platform Bundle Loader shall build aSspInfoPublic containing:

      - SspFamilyCryptoInfoBlock data structures as many as the number of family identifiers supported by the Secondary Platform Bundle Loader. Each SspFamilyCryptoInfoBlock data structure may contain aSspFamilyCryptoInfo. Each SspFamilyCryptoInfoBlock data structure may contain the set of SspOidCryptoInfoBlock data structures as many as custodians supported by the Secondary Platform Bundle Loader for the family identifier contained in the SspFamilyCryptoInfoBlock data structure. Each SspOidCryptoInfoBlock data structure shall contain the aCustodianOid and aSspOidCryptoInfo. The Secondary Platform Bundle Loader may include aSspGeneralCryptoInfo.

3) Return ANY_OK with the aSspInfoPublic.

12.6.5.5.2 Si3.SetSpbmCredential

The "Si3.SetSpbmCredential" function shall be used by the LBA during the bound SPB image download procedure as defined in clause 12.3.3.2.

The LBA shall use "Si3.SetSpbmCredential" function to deliver aSpbmCredential to the Secondary Platform Bundle Loader.

The "Si3.SetSpbmCredential" function command shall be ANY_SET_PARAMETER command defined in ETSI TS 103 666-1 [3], clause 8.5.4 which allows the LBA to update the registry.

The parameter of ANY_SET_PARAMETER command shall contain the index of IDS_CREDENTIAL_PARAMETER registry and the aSpbmCredential data structure defined in clause 12.6.2.3.

The LBA shall build the aSpbmCredential containing the aSpbFamilyId, the aSpbmKaCertificates, the aSspCiPkIdToBeUsed, and the aSspCryptoToBeUsed contained in the aSi2GetSpbmCertificateResponse.

On reception of the "Si3.SetSpbmCredential" command, the Secondary Platform Bundle Loader shall:

1) Set the received SpbmCredential data structure to the IDS_CREDENTIAL_PARAMETER registry.

2) Verify the received elements as follows:

   a) Verify that the aSpbmKaCertificates contained in the aSpbmCredential based on the certification path verification as defined in clause 12.2.1.1.4. The trusted public key used to verify the aSpbmKaCertificates shall be allowed to be used for loading the Secondary Platform Bundles with the aSpbFamilyId and aCustodianOid contained in the aSpbmCredential.

   b) Verify that the aSspCiPkIdToBeUsed is supported by itself for the loading of the Secondary Platform Bundles with the aSpbFamilyId and the aCustodianOid contained in the aSpbmCredential.

   c) Verify that the aSspCryptoToBeUsed is supported by itself for the loading of the Secondary Platform Bundles with the aSpbFamilyId and the aCustodianOid contained in the aSpbmCredential.

3) Select the appropriate Secondary Platform Bundle Loader certificate that shall be verifiable by the trusted public key which is indicated by the aSspCiPkIdToBeUsed contained in the aSpbmCredential.

4) Generate the following:

   a) A Secondary Platform Bundle Loader's ephemeral key pair. The domain parameter used to generate the ephemeral key pair shall be the same as the one indicated by the SubjectPublicKeyInfo in the SPB Manager certificate for key agreement contained in aSpbmCredential.

   b) ID_TRANSAC as defined in GlobalPlatform Open Firmware Loader for Tamper Resistant Element [6].

   c) aTbsSspImageSessionToken as defined in clause 12.6.2.4.

   d) aSspImageSessionTokenSignature by signing the aTbsSspImageSessionToken with the private key corresponding to the Secondary Platform Bundle Loader certificate for digital signature.

   e) The first session key as defined in Global Platform Open Firmware Loader for Tamper Resistant Element [6]. The first session key shall be generated with the Secondary Platform Bundle Loader's ephemeral private key and the public key contained in the SPB Manager certificate for key agreement.

NOTE: The first session key is the same as 'KS1' in GlobalPlatform Open Firmware Loader for Tamper Resistant Element [6].

   f) aSspInfoProtected as defined in clause 12.6.2.2.3.

   g) PrivateBlock containing aCodeM, aChallengeS, aSspInfoProtected and SPBL certificate which shall be protected. The aCodeM and aChallengeS shall be the same as those in the aSpbmCredential contained in GET_SSP_CREDENTIAL command parameter.

h) aM-SSP as defined in clause 12.6.2.4. The aM-SSP shall be generated by encrypting the PrivateBlock. The encryption algorithm indicated by the aEncryptionType and the first session key shall be used to generate the aM-SSP.

i) Verify the aPartNumberId as valid for processing in the SBP Manager.

5) Generate aSspCredential as defined in clause 12.6.2.4 and set the aSspCredential into the TRE_CREDENTIAL_PARAMETER registry.

6) Return ANY_OK with the GetSspCredentialResponse data structure to the LBA.

12.6.5.5.3 Si3.LoadBoundSpbInfo

The "Si3.LoadBoundSpbInfo" function shall be used by the LBA during the installation procedure as defined in clause 12.3.4.

The LBA shall use the "Si3.LoadBoundSpbInfo" function to provide the Secondary Platform Bundle Loader with the aDoOperateParameter contained in the bound SPB image received from the SPB Manager as the response of the "Si2.GetBoundSpbImage" function.

The "Si3.LoadBoundSpbInfo" function command shall be OFL_DO_OPERATE as defined in GlobalPlatform VPP - OFL VNP Extension [9].

The parameter of OFL_DO_OPERATE command shall be aDoOperateParameter defined in clause 12.6.2.5.

On reception of the "Si3.LoadBoundSpbInfo" function command, the Secondary Platform Bundle Loader shall:

1) Verify the aSpbmCerts contained in the aDoOperateParameter based on the certification path verification as defined in clause 12.2.1.1.4. The trusted public key used to verify the aSpbmCerts shall be the same as the one used to verify the aSpbmKaCertificates.

2) Verify the aSpbmTokenSignature contained in the aSpbmToken by using the SPB Manager certificate for digital signature. The SPB Manager certificate for digital signature shall be the last certificate in the aSpbmCerts.

3) Verify that the aIdTransac contained in the aTbsSpbmToken matches to the previously generated ID_TRANSAC.

4) Generate the second session key as defined in GlobalPlatform Open Firmware Loader for Tamper Resistant Element [6]. The second session key shall be generated with the aEPkSpbmKa contained in the aTbsSpbmToken and the Secondary Platform Bundle Loader's ephemeral private key.

NOTE: The second session key is the same as 'KS2' in GlobalPlatform Open Firmware Loader for Tamper Resistant Element [6].

5) Obtain the TIME_STAMP by decrypting the aM-TimeStamp by using the first session key and the encryption algorithm indicated by the aEncryptionType as defined in GlobalPlatform Open Firmware Loader for Tamper Resistant Element [6].

6) Obtain the ATK.ARP.DS by decrypting the aM-ARP by using the second session key and the encryption algorithm indicated by the aEncryptionType as defined in GlobalPlatform Open Firmware Loader for Tamper Resistant Element [6].

7) Obtain Image Descriptor by decrypting the aM-IMD by using the second session key and the encryption algorithm indicated by the aEncryptionType as defined in GlobalPlatform Open Firmware Loader for Tamper Resistant Element [6].

8) Verify the Image Descriptor as follows:

a) Verify that the family identifier contained in the Image Descriptor matches to the value of the family identifier in SSP_INFO_PUBLIC registry.

9) Verify the aSpbMetadata as follows:

a) Verify that the family identifier contained in the aSpbMetadata matches to the value of the family identifier in SSP_INFO_PUBLIC registry.

b) Verify that the aSpbId contained in the aSpbMetadata matches to the value of the public UUID of the image contained in the Image Descriptor.

c) Verify the aFamilySpecificData and aOidSpecificMetadata (out of scope of the present document).

10) Verify that the trusted public key used to verify the SPB Manager certificate is one of the trusted public keys supported by the Secondary Platform Bundle Loader used to load the Secondary Platform Bundle according to the rules below:

a) if the Secondary Platform Bundle family identifier is not part of any SspFamilyCryptoInfoBlock:

- the keys in aSspGeneralCryptoInfo.

b) else:

- the keys in aSspFamilyCryptoInfo, if none of the custodian OIDs in aSpbMetadata (either aCustodianOid or in aSupportedCustodianList) is part of any aSspOidCryptoInfoBlock;
- else, the keys in aSspOidCryptoInfo of the SspOidCryptoInfoBlock data structure which the custodian OID has been found in aSpbMetadata.

11) Return ANY_OK without any parameters to the LBA.

12.6.5.5.4 Si3.LoadBoundSpbSds

The "Si3.LoadBoundSpbSds" function shall be used by the LBA during the installation procedure as defined in clause 12.3.4.

The LBA shall use the "Si3.LoadBoundSpbSds" function to provide the Secondary Platform Bundle Loader with an element of aChangeSegmentParameter contained in the bound SPB image received from the SPB Manager as the response of the "Si2.GetBoundSpbImage" function.

The "Si3.LoadBoundSpbSds" function command shall be OFL_CHANGE_SEGMENT as defined in GlobalPlatform VPP - OFL VNP Extension [9].

The parameter of OFL_CHANGE_SEGMENT command shall be aChangeSegmentParameter defined in clause 12.6.2.5.

On reception of the "Si3.LoadBoundSpbSds" function command, the Secondary Platform Bundle Loader shall decrypt aChangeSegmentParameter to obtain Segment Descriptor as defined in GlobalPlatform Open Firmware Loader for Tamper Resistant Element [6]. The Secondary Platform Bundle Loader shall return ANY_OK to the LBA after successful decryption of the aChangeSegmentParameter.

NOTE: The aChangeSegmentParameter is the same as the Segment Descriptor Structure defined in GlobalPlatform Open Firmware Loader for Tamper Resistant Element [6].

12.6.5.5.5 Si3.LoadBoundSpbSeg

The "Si3.LoadBoundSpbSeg" function shall be used by the LBA during the installation procedure as defined in clause 12.3.4.

The LBA shall use the "Si3.LoadBoundSpbSeg" function to provide the Secondary Platform Bundle Loader with an element of aLoadSegmentParameter contained in the bound SPB image received from the SPB Manager as the response of the "Si2.GetBoundSpbImage" function.

The "Si3.LoadBoundSpbSeg" function command shall be OFL_LOAD_SEGMENT as defined in GlobalPlatform VPP - OFL VNP Extension [9].

The parameter of OFL_LOAD_SEGMENT command shall be aLoadSegmentParameter defined in clause 12.6.2.5.

On reception of the "Si3.LoadBoundSpbSeg" function command, the Secondary Platform Bundle Loader shall decrypt the aLoadSegmentParameter and install the decrypted segment into the iSSP as defined in GlobalPlatform Open Firmware Loader for Tamper Resistant Element [6]. The Secondary Platform Bundle Loader shall return ANY_OK to the LBA after the successful installation of the segment.

NOTE: The aLoadSegmentParameter is the same as the Segment Structure defined in GlobalPlatform Open Firmware Loader for Tamper Resistant Element [6].

12.6.5.5.6 Si3.GetSspCredential

The "Si3.GetSspCredential" function shall be used by the LBA during the bound SPB image download procedure as defined in clause 12.3.3.2.

The LBA shall use the "Si3.GetSspCredential" function to retrieve aSspCredential from the Secondary Platform Bundle Loader.

The "Si3.GetSspCredential" function command shall be ANY_GET_PARAMETER command defined in ETSI TS 103 666-1 [3], clause 8.5.4 which allows the LBA to retrieve the value of the registry.

The parameter of ANY_GET_PARAMETER command shall contain the index of TRE_CREDENTIAL_PARAMETER registry.

On reception of the "Si3.GetSspCredential" command, the Secondary Platform Bundle Loader shall return ANY_OK with the value of TRE_CREDENTIAL_PARAMETER registry which contains aSspCredential.

12.6.5.5.7 Si3.EnableSpb

The "Si3.EnableSpb" function shall be used by the LBA for the procedure to enable a Secondary Platform Bundle as defined in clause 12.4.1.

The LBA shall use the "Si3.EnableSpb" function to provide the Secondary Platform Bundle Loader with the Secondary Platform Bundle identifier to enable.

The "Si3.EnableSpb" function command shall be OFL_ENABLE_FIRMWARE as defined in GlobalPlatform VPP - OFL VNP Extension [9].

The Secondary Platform Bundle identifier to enable shall be the Public Image UUID as defined in GlobalPlatform VPP - OFL VNP Extension [9].

If the Secondary Platform Bundle to enable is a Telecom Secondary Platform Bundle, the Secondary Platform Bundle Loader shall check the number of currently enabled Telecom Secondary Platform Bundles and operate depending on the value of TELECOM_CAPABILITY as follows:

- If the number of currently enabled Telecom Secondary Platform Bundles is smaller than the value of TELECOM_CAPABILITY, the Secondary Platform Bundle Loader shall enable the Telecom Secondary Platform Bundle to enable.
- Otherwise, the Secondary Platform Bundle Loader shall reject the "Si3.EnableSpb" command with an error indicating that enabling the Telecom Secondary Platform Bundle is limited by TELECOM_CAPABILITY.

After successfully enabling the Secondary Platform Bundle, the Secondary Platform Bundle Loader shall update the value of the state to 'Enabled' in the Firmware session of that Secondary Platform Bundle.

12.6.5.5.8 Si3.DisableSpb

The "Si3.DisableSpb" function shall be used by the LBA for the procedure to disable a Secondary Platform Bundle as defined in clause 12.4.2.

The LBA shall use the "Si3.DisableSpb" function to provide the Secondary Platform Bundle Loader with the Secondary Platform Bundle identifier to disable.

The "Si3.DisableSpb" function command shall be OFL_DISABLE_FIRMWARE as defined in GlobalPlatform VPP - OFL VNP Extension [9].

The Secondary Platform Bundle identifier to disable shall be the Public Image UUID as defined in GlobalPlatform VPP - OFL VNP Extension [9].

After successfully disabling the Secondary Platform Bundle, the Secondary Platform Bundle Loader shall update the value of the state to 'Disabled' in the Firmware session of that Secondary Platform Bundle.

12.6.5.5.9 Si3.DeleteSpb

The "Si3.DeleteSpb" function shall be used by the LBA for the procedure to delete a Secondary Platform Bundle as defined in clause 12.4.3.

The LBA shall use the "Si3.DeleteSpb" function to provide the Secondary Platform Bundle Loader with the Secondary Platform Bundle identifier to delete.

The "Si3.DeleteSpb" function command shall be OFL_DELETE_SESSION as defined in GlobalPlatform VPP - OFL VNP Extension [9].

The Secondary Platform Bundle identifier to disable shall be the Public Image UUID as defined in GlobalPlatform VPP - OFL VNP Extension [9].

12.6.5.5.10 Si3.GetSpbMetadata

The "Si3.GetSpbMetadata" function shall be used by the LBA to retrieve the SPB metadata of a Secondary Platform Bundle container installed in the iSSP.

The "Si3.GetSpbMetadata" function command shall be GET_SPB_METADATA.

The parameter of GET_SPB_METADATA command is a Secondary Platform Bundle identifier.

The Secondary Platform Bundle identifier shall be the Public Image UUID as defined in GlobalPlatform VPP - OFL VNP Extension [9].

On reception of the "Si3.GetSpbMetadata" function command, the Secondary Platform Bundle Loader shall:

1) find the firmware session which contains the Public Image UUID same as the received Secondary Platform Bundle identifier;

2) extract the SPB metadata contained in that firmware session;

3) return ANY_OK with the SPB metadata as the "Si3.GetSpbMetadata" function response.

12.6.5.5.11 Si3.UpdateSpbState

The "Si3.UpdateSpbState" function shall be used by the LBA during the SPB state retrieving procedure as defined in clause 12.4.5.

The LBA shall use "Si3.UpdatetSpbState" function to request the Secondary Platform Bundle Loader to update the value of SPB_ID registry.

The "Si3.UpdateSpbState" function command shall be ANY_SET_PARAMETER command defined in ETSI TS 103 666-1 [3], clause 8.5.4 which allows the LBA to update the registry.

The parameter of ANY_SET_PARAMETER command shall contain the index of SPB_ID registry and the Secondary Platform Bundle identifier.

On reception of the "Si3.UpdateSpbState" command, the Secondary Platform Bundle Loader shall:

1) Set the received Secondary Platform Bundle identifier (SpbId) to the SPB_ID registry.

2) Extract the SPB state from the firmware session which contains the Public Image UUID same as the received SpbId.

3) Update the SPB_STATE registry with the value of the extracted SPB state.

4) Return ANY_OK to the LBA as "Si3.UpdateSpbState" function response to the LBA.

12.6.5.5.12 Si3.GetSpbState

The "Si3.GetSpbState" function shall be used by the LBA during the SPB state retrieving procedure as defined in clause 12.4.5.

The "Si3.GetSpbState" function command shall be ANY_GET_PARAMETER command defined in ETSI TS 103 666-1 [3], clause 8.5.4 which allows the LBA to retrieve the value of the registry.

The parameter of ANY_GET_PARAMETER command shall contain the index of SPB_STATE registry.

On reception of the "Si3.GetSpbState" command, the Secondary Platform Bundle Loader shall return ANY_OK with the value of SPB_STATE registry to the LBA.

Annex A (normative): Additions for Telecom Secondary Platform Bundles

A.1 Telecom family identifier

The telecom family identifier identifies Telecom Secondary Platform Bundles, i.e. Secondary Platform Bundles which contain or are intended to contain at least one 3GPP NAA. Its value is defined in ETSI TS 101 220 [27].

There may be more than one custodian that defines specific requirements (e.g. trusted CIs, product certification, operational modes of the terminal) for the telecom family identifier.

A.2 Data types for telecom family identifier

A.2.1 Introduction

The two following clauses describe the data types that are applicable to the telecom family identifier.

A.2.2 SSP information

The ASN.1 data structure of aFamilySpecificSspInfo in aFamilySpecificSspInfoBlock:

```
-- ASN1START

TelecomSpecificSspInfo ::= SEQUENCE
{
    aTelecomBundleConcurrencyCapability INTEGER (1..MAX)
}

-- ASN1STOP
```

aTelecomBundleConcurrencyCapability: it indicates the number which shall be set on the iSSP at the time of manufacturing to limit the maximum number of the Telecom Secondary Platform Bundles in the Enabled or Active state.

A.2.3 Terminal information

The ASN.1 data structure of aFamilySpecificTerminalInfo in aFamilySpecificTerminalInfoBlock:

```
-- ASN1START

TelecomSpecificTerminalInfo ::= SEQUENCE
{
    aTAC OCTET STRING (SIZE(4))
}

-- ASN1STOP
```

aTAC: it indicates the type of allocation code defined in ETSI TS 123 003 [40]. The value shall be encoded as a Telephony Binary Coded Decimal String as defined in ETSI TS 129 002 [41].

A.3 SPB metadata for the telecom family identifier

The SPB metadata for the Secondary Platform Bundle associated with the telecom family identifier shall satisfy the followings:

- The SPB metadata shall contain the family identifier-specific metadata.
- The Family Identifier-specific metadata in the SPB metadata shall contain the telecom family identifier.
- The SPB metadata shall use TelecomFamilyData data type as aFamilySpecificData contained in the family identifier-specific metadata.

The ASN.1 data structure of the TelecomFamilyData is described below.

```
-- ASN1START

TelecomFamilyData ::= SEQUENCE
{
    aBundleClass     BundleClass OPTIONAL,
    aRequireUserIntent  IntentType OPTIONAL  -- The bundle management operation requiring the user
-- intent before being performed
}

BundleClass ::= ENUMERATED
{
    eTestTelecomBundle (0)  -- To indicate that this Telecom Bundle is a Test Telecom Bundle
}

IntentType ::= BIT STRING
{
    eInstallation (0), -- The user intent is required prior to installing the Telecom Secondary
-- Platform Bundle
    eEnable (1), -- The user intent is required prior to enabling the Telecom Secondary Platform
-- Bundle
    eDisable (2), -- The user intent is required prior to disabling the Telecom Secondary Platform
-- Bundle
    eDelete (3) -- The user intent is required prior to deletion of the Telecom Secondary Platform
-- Bundle
}

-- ASN1STOP
```

A.4 Terminal behaviour

When at least two Telecom Secondary Platform Bundles are in the Enabled or Active state, the operational modes of the terminal are described by the custodians of the telecom family identifier associated to the Enabled or Active Telecom Secondary Platform Bundles.

If the terminal uses an iSSP to access a network according to Recommendation ITU-T E.212 [43], it shall use a Telecom Secondary Platform Bundle from that iSSP to access to that network.

A.5 Telecom Secondary Platform Bundle management

A.5.1 Introduction

The following clauses describe the Secondary Platform Bundle management that is applicable to the telecom family identifier.

A.5.2 Switch Telecom Secondary Platform Bundles

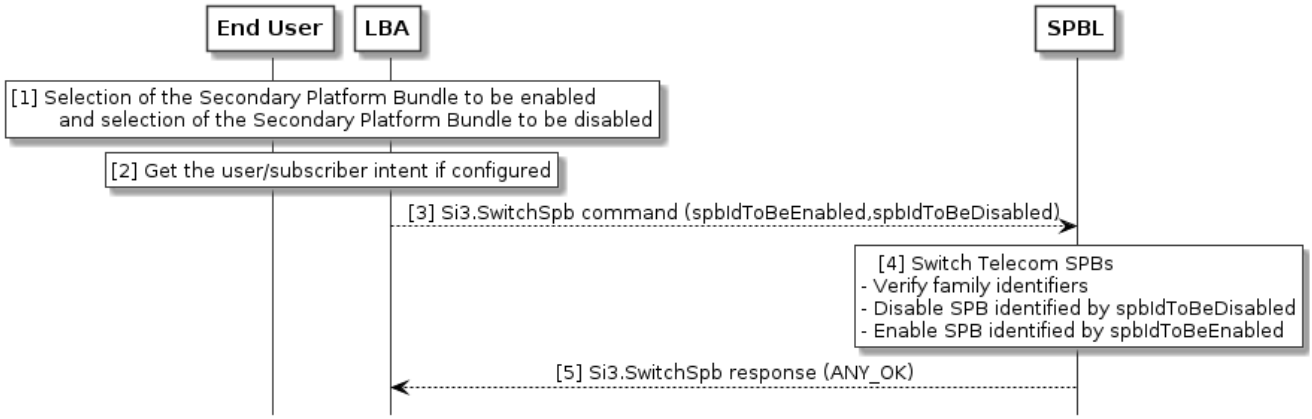

Figure A.1: Switch Telecom Secondary Platform bundle

This clause describes the procedure to disable one of the currently enabled Telecom Secondary Platform Bundles, and then enable another Telecom Secondary Platform Bundle installed on the iSSP.

The procedure to switch Telecom Secondary Platform Bundles installed on the iSSP shall use the following steps:

NOTE: The term "switching" means disabling an enabled Telecom Secondary Platform Bundle and then enabling another Telecom Secondary Platform Bundle.

1) The end user selects the Telecom Secondary Platform Bundle to be enabled and the Telecom Secondary Platform Bundle to be disabled through the LBA (out of scope of the present document).

2) The LBA shall get the user intent if the user intent required is configured in the SPB metadata of Telecom Secondary Platform Bundle to be disabled. In addition, the LBA shall get the user intent if the user intent required is configured in the SPB metadata of Telecom Secondary Platform Bundle to be enabled.

3) The LBA shall call the "Si3.SwitchSpb" function. The function command shall contain the identifier of the Telecom Secondary Platform Bundle to be enabled ("spbIdToBeEnabled") and the identifier of the Telecom Secondary Platform Bundle to be disabled ("spbIdToBeDisabled").

4) On the reception of the "Si3.SwitchSpb" function command, the Secondary Platform Bundle Loader shall disable the Telecom Secondary Platform Bundle identified by "spbIdToBeDisabled" and then enable the Telecom Secondary Platform Bundle identified by "spbIdToBeEnabled" as defined in clause A.6.2.

5) The Secondary Platform Bundle Loader shall use the "Si3.SwitchSpb" response to indicate the execution status of the command.

A.6 Si3 interface function for telecom family identifier

A.6.1 Introduction

The following clauses describe the Si3 functions that are applicable to the telecom family identifier.

A.6.2 Si3.SwitchSpb

The "Si3.SwtichSpb" function shall be used by the LBA to switch Telecom Secondary Platform Bundles as defined in clause A.5.2.

The LBA shall use the "Si3.SwitchSpb" function to provide the Secondary Platform Bundle Loader with the identifier of Telecom Secondary Platform Bundle to be enabled and the identifier of an enabled Telecom Secondary Platform Bundle to be disabled.

The "Si3.SwitchSpb" function command shall be SWITCH_TELECOM_SPB.

The parameter of SWITCH_TELECOM_SPB is defined as follows:

```
-- ASN1START

SwitchTelecomSpb ::= SEQUENCE
{
    aSpbIdToBeEnabled UUID, -- the identifier of Telecom Secondary Platform Bundle to be enabled
    aSpbIdToBeDisabled UUID -- the identifier of Telecom Secondary Platform Bundle to be disabled
}

-- ASN1STOP
```

The identifiers of Telecom Secondary Platform Bundles to be enabled and disabled shall be the Secondary Platform Bundle identifiers as defined in clause 9.4.5.

On reception of SWITCH_TELECOM_SPB, the Secondary Platform Bundle Loader shall:

1) Verify that the family identifiers of the Secondary Platform Bundles identified by aSpbIdToBeEnabled and aSpbIdToBeDisabled are the telecom family identifiers. If either of the family identifiers is not the telecom family identifier, the Secondary Platform Bundle Loader shall response with an error indicating that the family identifier is not valid.

2) Verify that the Telecom Secondary Platform Bundle identified by aSpbIdToBeDisabled can be disabled. If the Telecom Secondary Platform Bundle identified by aSpbIdToBeDisabled is not currently enabled or does not exist, the Secondary Platform Bundle Loader shall response with an error indicating that aSpbIdToBeDisabled is not valid.

3) Verify that the Telecom Secondary Platform Bundle identified by aSpbIdToBeEnabled can be enabled. If the Telecom Secondary Platform Bundle identified by aSpbIdToBeEnabled is not currently disabled or does not exist, the Secondary Platform Bundle Loader shall response with an error indicating that aSpbIdToBeEnabled is not valid.

4) Disable the Telecom Secondary Platform Bundle identified by aSpbIdToBeDisabled.

5) Enable the Telecom Secondary Platform Bundle identified by aSpbIdToBeEnabled.

6) Return ANY_OK without any parameters to the LBA if switching Telecom Secondary Platform Bundles is successfully performed.

Annex B (normative): ASN.1 definitions

B.1 End of ASN.1

```
-- ASN1START

END

-- ASN1STOP
```

B.2 Complete ASN.1 file

The complete ASN.1 definition, as generated merging all the ASN.1 snippets present in the present document is available at the following URL:

- https://forge.etsi.org/rep/set/ts_103666-2_issp/raw/v15.4.0/asn.1/asn1_syntax.asn1.

Annex C (normative): Bundle eligibility check

C.1 Introduction

The SPB Manager verifies that the Secondary Platform Bundle to be downloaded is compatible with the target SSP device when the SPB Manager receives the request of the Secondary Platform Bundle download from the target SSP device.

NOTE: The bundle eligibility check does not include the check on the certification path to download the Secondary Platform Bundle.

The bundle eligibility check procedure is comprised of the basic eligibility check and family identifier-specific eligibility check.

The basic eligibility check shall be performed regardless of the family identifier.

The family identifier-specific eligibility check shall be performed according to the rules defined by a/the custodian(s) for that family identifier.

C.2 Basic eligibility check

C.2.1 Summary

The basic eligibility check consists of the following:

- Version compatibility check.
- Bundle compatibility check.
- Primary platform identifier check.

C.2.2 Version compatibility check

The SPB Manager shall check whether the release of the specification that is implemented by the Secondary Platform Bundle Loader and the LBA is supported or not.

C.2.3 Bundle compatibility check

The SPB Manager shall check whether the Secondary Platform Bundle image which is referenced by the CodeM is compatible with the terminal containing the SSP. The SPB Manager shall check the compatibility of the Secondary Platform Bundle image referenced by the CodeM based on the Part Number of the SSP.

C.2.4 Primary platform identifier check

If the Secondary Platform Bundle image referenced by the CodeM is linked to a primary platform identifier, the SPB Manager shall verify that it matches the primary platform identifier of the SSP.

C.3 Family identifier-specific eligibility check

The SPB manager may perform the family identifier-specific eligibility check based on the family identifier-specific SSP information and/or the family identifier-specific terminal information as defined in clauses 12.6.2.2.3 and 12.6.2.7, respectively.

The processes of family identifier-specific eligibility check are out of scope of the present document.

Annex D (informative): UML code of figures

Table D.1 contains the link to the UML code used to generate some of the figures in the present document.

Table D.1: Link to UML code

Figure	Link to UML code
Figure 12.2	https://forge.etsi.org/rep/set/ts_103666-2_issp/raw/v15.0.0/figures/Figure_12_2.plantuml
Figure 12.3	https://forge.etsi.org/rep/set/ts_103666-2_issp/raw/v15.0.0/figures/Figure_12_3.plantuml
Figure 12.4	https://forge.etsi.org/rep/set/ts_103666-2_issp/raw/v15.0.0/figures/Figure_12_4.plantuml
Figure 12.5	https://forge.etsi.org/rep/set/ts_103666-2_issp/raw/v15.0.0/figures/Figure_12_5.plantuml
Figure 12.6	https://forge.etsi.org/rep/set/ts_103666-2_issp/raw/v15.0.0/figures/Figure_12_6.plantuml
Figure 12.7	https://forge.etsi.org/rep/set/ts_103666-2_issp/raw/v15.0.0/figures/Figure_12_7.plantuml
Figure 12.8	https://forge.etsi.org/rep/set/ts_103666-2_issp/raw/v15.3.0/figures/Figure_12_8.plantuml
Figure 12.9	https://forge.etsi.org/rep/set/ts_103666-2_issp/raw/v15.0.0/figures/Figure_12_9.plantuml
Figure 12.10	https://forge.etsi.org/rep/set/ts_103666-2_issp/raw/v15.0.0/figures/Figure_12.10.plantuml
Figure 12.11	https://forge.etsi.org/rep/set/ts_103666-2_issp/raw/v15.0.0/figures/Figure_12_11.plantuml
Figure 12.12	https://forge.etsi.org/rep/set/ts_103666-2_issp/raw/v15.0.0/figures/Figure_12_12.plantuml
Figure 12.13	https://forge.etsi.org/rep/set/ts_103666-2_issp/raw/v15.1.0/figures/Figure_12_13.plantuml
Figure 12.14	https://forge.etsi.org/rep/set/ts_103666-2_issp/raw/v15.3.0/figures/Figure_12_14.plantuml
Figure 12.15	https://forge.etsi.org/rep/set/ts_103666-2_issp/raw/v15.3.0/figures/Figure_12_15.plantuml
Figure 12.16	https://forge.etsi.org/rep/set/ts_103666-2_issp/raw/v15.3.0/figures/Figure_12_16.plantuml
Figure A.1	https://forge.etsi.org/rep/set/ts_103666-2_issp/raw/v15.2.0/figures/Figure_A_1.plantuml

Annex E (informative): Change history

The table below indicates all changes that have been incorporated into the present document since it was placed under change control.

Change history								
Date	**Meeting**	**Plenary Doc**	**CR**	**Rev**	**Cat**	**Subject/Comment**	**Old**	**New**
2019-09-27	SCP#89	SCP(19)000213				Version 15.0.0 first publication	-	15.0.0
2019-12-20	SCP#90	SCP(19)000242	002		F	Rule for the use of Telecom SPB to access a public network	15.0.0	15.1.0
		SCP(19)000243	003		B	SPB metadata retrieving procedure		
		SCP(19)000244r1	004	1	B	Addition on SSP and terminal information		
		SCP(19)000246	006		F	Editorial improvement to clause 12		
		SCP(19)000256	008		B	notification token		
		SCP(19)000257r1	009	1	B	Primary Platform Identifier registry		
		SCP(19)000287	001	1	F	Definition of Primary Platform Identifier		
2020-04-02	SCP#91	SCP(19)000255r2	007	2	B	SSP activation code	15.1.0	15.2.0
		SCP(20)091018	010		B	Clarification of the procedure to notify the service provider		
		SCP(20)091019	011		F	Clarification of the Primary Platform Identifier		
		SCP(20)091020	012		F	Update Common Criteria reference		
		SCP(20)091021	013		F	HMAC reference and ASN.1 correction		
		SCP(20)091022	014		F	Si3 command for Telecom SPB		
2020-06-25	SCP#94	SCP(20)000055	005	1	B	Assurance levels of the PP and the SPBL	15.2.0	15.3.0
		SCP(20)000045r1	015	1	D	Clarification of the use of "user intent" / "user consent"		
		SCP(20)000043r1	016	1	F	CR 103 666-2 Rel-15 Clarification on additional extension fields for certificate		
		SCP(20)000044r1	017	1	F	Capability exchange clarification		
		SCP(20)000056r1	018	1	B	SPB state retrieval		
		SCP(20)000057r1	019	1	F	Clarification on the certification path verification		
2021-12-10	SCP#103	SCP(21)000213	021		F	Corrections of spelling problems	15.3.0	15.4.0
		SCP(21)000214	022		F	Refinement in Table 7.4		
		SCP(21)000215	023		F	Correction of ASN.1 code in clause 12.6.3.8.1 additional extension fields for certificate		
		SCP(21)000217	025		F	Mismatches in public and private keys between SPBL and SPBM		
		SCP(21)000218r1	024	1	F	Addition of a generic error code		
2022-03-25	SCP#104	SET(22)000051	20	1	F	Clarification of data encryption algorithms	15.3.0	15.4.0
		SET (22)000052	26		F	Minimum_size_of_the_metadata		
		SET (22)000053	27		F	ChallengeS size clarification		
		SET (22)000054	28		F	Container for SSP encrypted data		
		SET (22)000055	29		F	CodeM_originator		
2024-10	SET#115	SET(24)000128	31		F	References Corrections	15.4.0	15.5.0

History

Document history		
V15.0.0	November 2019	Publication
V15.1.0	February 2020	Publication
V15.2.0	June 2020	Publication
V15.3.0	September 2020	Publication
V15.4.0	August 2022	Publication
V15.5.0	December 2024	Publication